中国人民大学
RENMIN UNIVERSITY OF CHINA

WAMDM
Innovative Data Systems Research

# 2009
# ANNUAL REPORT

网络与移动数据管理实验室

Lab of Web&Mobile Data Management

2010 年 1 月

中国人民大学
RENMIN UNIVERSITY OF CHINA

WAMDM
Innovative Data Systems Research

# 2009
# ANNUAL REPORT

网络与移动数据管理实验室

Lab of Web&Mobile Data Management

2010 年 1 月

# 序

在过去的十年间，随着互联网的迅速发展，整个 Web 的数据量已经超过了200,000TB，并仍在快速地增长，这使其成为人们获取有用信息的最重要的途径之一。另一方面，随着 3G 时代的到来，大量的手机、移动设备需要频繁访问互联网，以从互联网上获取丰富的信息，这是一个必然的趋势。而 3G 所来带的高带宽，使得未来手机将不再是一个简单的通话工具，人们从互联网上获取信息将越来越依赖于手机和以及其它移动设备。

目前，虽然用户已经能够通过手机及其它移动设备访问互联网，进行信息查询。但是无论从互联网上的信息集成系统而言，还是从手机上的查询服务而言，都远远不能满足用户的需求。因此如何解决面向**移动用户的 Web 数据集成问题**，成为实验室今后关注的一个新的研究领域，目前研究界还缺乏有关的研究成果，我们认为这是一个创新的机遇。

此外，云计算是当今信息产业最受关注的一种计算模式，在这种模式下，企业和个人可以根据自己的需要购买存储设备和计算能力，而不是花费大量资金购买大规模高性能计算机。作为云计算的一项关键技术，云数据存储和云数据管理为业界带来巨大的潜在商用价值。随着信息产业的发展，企业和公司产生的数据量快速增长，通常数据规模可以达到 TB 甚至 PB 级别。如何管理和分析海量数据是目前很多领域所面临的问题，例如在医疗、通信和互联网领域。云环境是由大量的性能普通、价格便宜的计算节点组成的一种无共享大规模并行处理环境，所以从成本和性能两方面考虑，越来越多的企业更愿意把自己的数据中心从昂贵的高性能计算机转移到共有或私有云环境中。对此实验室的提出的新的研究课题是**云计算环境下数据库技术**，实现一种具有高可用性、高容错性、可扩展性和高性能的云数据库系统。为此我们创办了首个云数据管理的研讨会 CouldDB2009（First International Workshop on Cloud Data Management, conjunction with CIKM2009, Hong Kong）,并与工业界建议了密切的合作关系，开设了“移动&云计算系列学术报告”。

一年即将过去，在继过去三年有关实验室科研情况的年度报告的基础上，再次整理了 2009 年的年度进展报告，感觉即是对我们的鞭策与鼓励，也是对同行的一份心意。

这一年实验室还是喜事连连。我们历经九年的研究成果“纯 XML 数据库系统技术”获得本年度中国计算机学会**“王选奖”一等奖**。其次硕士生王仲远同学获得中国人民大学最高奖励“**吴玉章奖学金**”，刘伟博士获得中国人民大学**校级优秀博士论文奖**。这些奖励得益于实验室师生的共同努力，以及我们一贯坚持的严谨的学风。此外我所主持的项目“基于受限网络的移动对象数据库关键技术研究（项目批准号：60573091）”被评为“特优”，这是我所主持的项目第二次被评为“特优”。

本年度我们还举办了两届闪存数据库系统研讨会（The Workshop on Flash-based Database Systems），这是在我们所主持的国家自然科学基金重点项目“闪存数据库技术研究”的支持下创立的学术交流平台，也是课题组探索的一种新的课题组织方式。我们发现，小同行间的深入交流，有助于研究研究水平的提高。目前课题组在闪存数据库存储管理、缓冲区管理、查询处理和事务处理，以及闪存开发板、闪存硬件测试等方面的取得最新研究进展和技术成果。研究表明目前闪存对现有数据库的性能提升在 10 倍左右，课题组的研究目标是将这一性能再提升 5-10 倍。

在此谨以此年报感谢来自学校方方面面的支持，感谢国家自然基金委和 863 计划的资助，感谢所有关心和支持过我们的人们。

孟小峰

2009 年 12 月 31 日于北京

# 目　录

## 近似字符串检索 (Approximate String Search)



## XML 关键字检索( XML Keyword Search)



## 闪存数据库系统 (Flash Database Systems)

# 实验室年度亮点

>> 孟小峰教授团队获“王选奖”一等奖殊荣

>> 刘伟博士获校优秀博士论文奖

>> 王仲远获 2008-2009 学年吴玉章奖学金

>> 实验室获得 863 计划重点项目支持

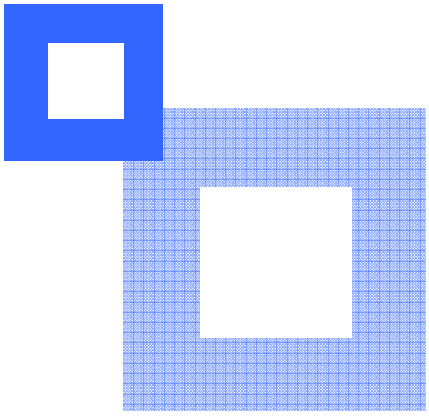
>> 出版网络与移动数据管理系列学术专著

>> 实验室发表多篇高水平学术论文

>> 国家自然科学基金结题项目获评“特优”

>> 举办第一届云数据管理国际研讨会

>> 实验室举办首届暑期研讨会



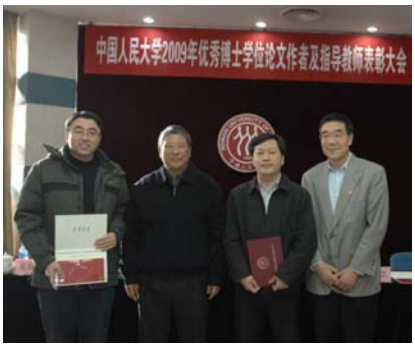
## ● 孟小峰教授团队获“王选奖”一等奖殊荣


2009 年 10 月 23-24 日，中国计算机学会 2009 年中国计算机大会暨理事会议在天津召开，中国计算机学会的最高奖项——“王选奖”在会上揭晓，中国人民大学孟小峰教授团队获“王选奖”一等奖殊荣。

进入新世纪以来，数据库技术面临一场变革，即在原有关系数据库技术成熟之后，新的数据库技术在哪里？一个重要的趋势是具有灵活的半结构化特性的 XML 数据的出现。XML 作为一种数据存储和交换格式，在互联网络环境中扮演着极其重要的角色，它已经成为数据交换事实上的标准，在电子商务、电子政务、金融、出版、科学数据与各种资源的数字化等方面得到越来越广泛和深入的应用。可以想象，在不久的将来，XML 数据的规模将可能达到或者超过各种关系数据库中的数据规模，从而成为继关系数据之后新的主流数据形式。如何有效管理 XML 数据自然成为寻找突破口的数据库界的热点研究问题。

孟小峰教授自 2000 年就开始了 XML 数据管理的研究工作，以构建纯 XML 数据库系统为目标，深入地研究了 XML 数据的相关技术，在 XML 数据的存储结构、索引、查询代数、查询优化、XML 数据更新、XML 近似查询以及纯 XML 数据管理系统实现技术等方面取得了大量的创新性成果，并在国际顶级会议和期刊发表多篇高水平论文。2002 年率先在国内开了发纯 XML 数据库系统 OrientX，历经八年，先后发布六个版本，有来自 50 多个国家 20 多万用户访问，有 1 万多用户下载使用，成为本领域一个具有代表性的成果，它对推动我国 XML 数据库技术的研究和开发以及高层次人才的培养具有重要的意义。

## ● 刘伟博士获校优秀博士论文奖




12 月 29 日下午，中国人民大学隆重表彰 2009 年优秀博士学位论文作者及其指导教师。 中国人民大学校长、学位评定委员会主席纪宝成，常务副校长、学位评定委员会副主席、研究生院院长袁卫出席表彰大会并为获奖博士研究生和指导教师颁发证书与奖金。

博士生刘伟的学位论文《Deep Web 数据集成中的关键技术研究》，获得中国人民大学优秀博士论文奖励。

刘伟博士师从孟小峰教授，研究领域是 Deep Web 数据集成技术，发表多篇高水平论文包括国际顶级期刊《IEEE 知识与数据工程学报》(IEEE Transactions on Knowledge and Data Engineering (TKDE) )论文，毕业后到北大王选院士创办的计算机科学与技术研究所从事博士后研究工作，获中国博士后科学基金资助。

## 王仲远获 2008-2009 学年吴玉章奖学金

吴玉章奖学金是人民大学的传统奖项，也是中国人民大学最高奖项。旨在鼓励和表彰学生全面发展。全校共有 27 名候选学生覆盖本科生、硕士研究生和博士研究生。最终王仲远同学在 27 名候选学生中脱颖而出，获评 2008-2009 学年吴玉章奖学金。

在学期间，王仲远同学在孟小峰教授的指导下，做事踏实，为人本分，先后获 SIGMOD Undergraduate Award，作为技术骨干开发若干系统如中文文献系统 C-DBLP 等，获得广泛的好评。即将毕业之际，王仲远同学顺利入职微软亚洲研究院。

## 实验室获得 863 计划重点项目支持

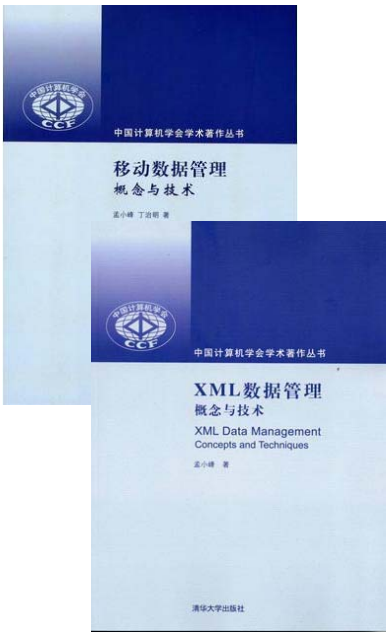
2009 年 5 月 10 日，国家 863“普适计算基础软硬件关键技术及系统”重点项目预启动会议在广州顺德召开。孟小峰教授领导的团队承担八个课题之一的“隐私保护技术”部分，会上人大课题组具体汇报了课题的实施方案和计划，孟小峰教授对项目汇报中的问题和方向进行了解答和指导。这是人民大学作为负责单位承担的首个 863 重点项目课题。

该 863 重点项目以《国家中长期科学与技术发展规划纲要》和“十一五”科技发展规划为指导，针对目前国际上普适计算技术的研究现状和最新发展趋势，在综合考虑我国对普适计算的应用需求、技术体系、研究积累与人才队伍的基础上设立的。此次预启动会议，共有项目相关的十一家单位参加，其中包括清华，浙大，人大，中科院软件所，中科院计算所，闪联，东软等。

## 出版网络与移动数据管理系列学术专著

2009 年孟小峰教授先后出版了《移动数据管理：概念与技术》（2009,3）和《XML 数据管理：概念与技术》（2009,10）。

2000 年以来，孟小峰教授将研究目标定位在创新数据管理的研究上，针对数据库技术与 Web 计算和移动计算交叉结合所产生的挑战性问题开展研究，包括 Web 数据管理、移动数据管理和 XML 数据管理。依据多年在移动数据管理和 XML 数据管理的研究积累，依托“中国计算机学会学术著作丛书”出版了《移动数据管理：概念与技术》和《XML 数据管理：概念与技术》学术专著。《移动数据管理：概念与技术》这本书总结了国内外有关移动数据的研究工作和具有代表性的关键技术，详细介绍了课题组近年来的一些研究成果；《XML 数据管理：概念与技术》从数据库系统实现的角度，全面系统地介绍了纯 XML 数据库系统相关技术，一本系统反映 XML 数据管理领域最新技术发展的书籍。




## 实验室发表多篇高水平学术论文

本年度实验室发表了多篇高水平的学术论文，其中刘伟，孟小峰和孟卫一合作的论文“ViDE: A Vision-based Approach for Deep Web Data Extraction”被数据库领域国际顶级期刊“IEEE Transactions on Knowledge and Data Engineering (TKDE)”作为 regular paper 录用。陈继东博士和孟小峰教授合作的论文“Update-efficient Indexing of Moving Objects in Road Networks”被国际顶级期刊 Geoinformatica 发表。(Vol.13 (4):397-424, September, 2009)。周军锋博士在中国科学（F 辑）发表一篇名为“Efficient Processing of Partially Specified Twig Pattern Queries”的论文。（Vol.52(10):1830-1847, Oct. 2009. Chinese version: 39(10):1034-1049）

## ● 国家自然科学基金结题项目获评“特优”

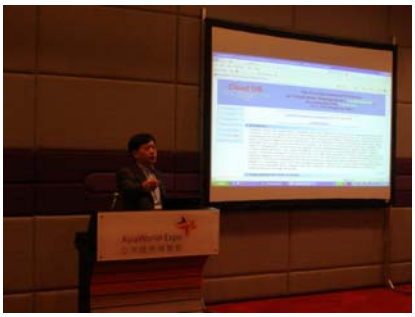
国家自然科学基金委员会信息科学部近日公布了2008年底结题项目的评估结果，人民大学信息学院申报结题的项目取得了较好的成绩。其中孟小峰教授主持的项目 “基于受限网络的移动对象数据库关键技术研究（项目批准号：60573091）”被评为“特优”，这也是孟小峰教授主持的项目第二次被评为“特优”。

国家自然科学基金项目评审一贯注重“绩效挂钩”原则，强调资助的连续性和绩效性。其中，特优项目数一般占当年结题评优项目数的5%，孟小峰教授连续获得“特优”，是我校国家自然科学基金历年结题项目中的最好成绩，也充分体现了我校科研人员的科研水平。


## ● 举办第一届云数据管理国际研讨会

**The First International Workshop on Cloud Data Management (CIKM 2009)**

由WAMDM实验室承办的第一届云数据管理国际研讨会于2009年11月2日在中国香港成功召开，孟小峰教授担任本次研讨会的联合主席。本次研讨会依附于第18届信息与知识管理国际会议（CIKM2009）。它是国际上第一个关于云数据管理的国际研讨会，吸引了国内外多所著名大学和研究机构的学者参与进来共同探讨云计算技术和发展。



## ● 实验室举办首届暑期研讨会

WAMDM实验室全体师生于2009年8月1日至8月4日在美丽的海滨城市秦皇岛成功的举办了第一次暑期研讨会。在这次研讨会中，首先进行了为期一天的学术研讨，并与燕山大学信息学院学生进行深入的学术交流；其次，经过了一个学期辛苦的学习，希望在这里让大家放飞一下心情，大家游览了北戴河的大好风光。

# 数据管理前沿技术报告

# 基于位置服务中的隐私保护

**孟小峰**

近年来随着传感器和无线移动设备的飞速发展，随时随地获得个人位置成为可能。一方面，促进了新一类应用程序——基于位置服务的出现与发展；另一方面，个人隐私保护问题引起人们的广泛关注。由于移动环境中位置信息的特殊性，造成无法直接利用现有的关系数据库隐私保护技术。本文分析了位置隐私保护中存在的挑战问题，从系统结构、位置匿名技术和查询处理技术三方面归纳总结了现有的研究工作，并指出了未来的研究方向。


# 基于位置服务中的隐私保护

**孟小峰**
**中国人民大学信息学院**



## 报告大纲

- 隐私保护问题及意义
- 隐私保护系统结构
- 隐私保护研究内容
- 隐私保护面临挑战
- 总结

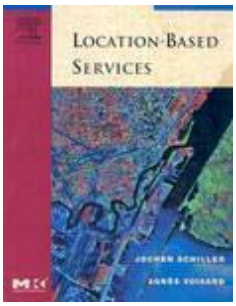
# 基于位置服务（LBS）




# 基于位置服务

- Services that integrate a mobile device's location or position with other information so as to provide added value to a user

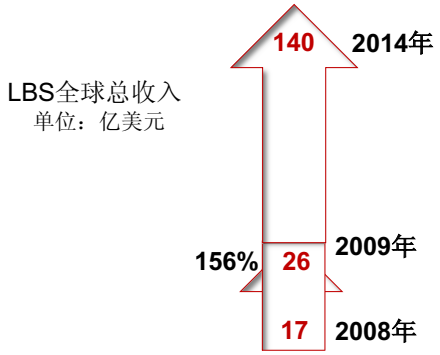
(基于位置的信息服务是将一个移动设备的**位置**或者**坐标**和其他信息整合起来，为用户提供增值服务)

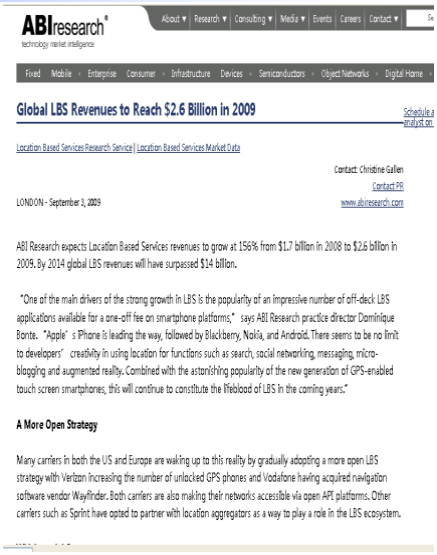



J. Schiller, Jochen, A. Voisard,Location-based Services, Elsevier Science Ltd, April 2004

# 基于位置服务

- 美国著名市场研究公司ABI research日前发布预测



ABIresearch
technology market intelligence

About ▾ | Research ▾ | Consulting ▾ | Media ▾ | Events | Careers | Contact ▾

Fixed · Mobile · Enterprise · Consumer · Infrastructure · Devices · Semiconductors · Object Networks · Digital Home

**Global LBS Revenues to Reach $2.6 Billion in 2009**

Schedule a analyst on

Location Based Services Research Service | Location Based Services Market Data

Contact: Christine Gallen
Contact PR
www.abiresearch.com

LONDON - September 3, 2009

ABI Research expects Location Based Services revenues to grow at 156% from $1.7 billion in 2008 to $2.6 billion in 2009. By 2014 global LBS revenues will have surpassed $14 billion.

"One of the main drivers of the strong growth in LBS is the popularity of an impressive number of off-deck LBS applications available for a one-off fee on smartphone platforms," says ABI Research practice director Dominique Bonte. "Apple's iPhone is leading the way, followed by Blackberry, Nokia, and Android. There seems to be no limit to developers' creativity in using location for functions such as search, social networking, messaging, micro-blogging and augmented reality. Combined with the astonishing popularity of the new generation of GPS-enabled touch screen smartphones, this will continue to constitute the lifeblood of LBS in the coming years."

**A More Open Strategy**

Many carriers in both the US and Europe are waking up to this reality by gradually adopting a more open LBS strategy with Verizon increasing the number of unlocked GPS phones and Vodafone having acquired navigation software vendor Wayfinder. Both carriers are also making their networks accessible via open API platforms. Other carriers such as Sprint have opted to partner with location aggregators as a way to play a role in the LBS ecosystem.



# LBS应用领域

- 军事和政府产业
  - 全球第一个位置系统GPS，最初主要用于军事和涉及国家重要利益的民用领域

- 紧急救援服务
  - 1996年，联邦通信委员会（FCC）颁布E-911法规要求移动运营商为手机用户提供紧急救援服务
    1999年FCC对E-911法进行修订
  - 欧洲于2003年1月1日开始实施 “US FCC”标准——建议使用E-OTD即“增强型观测时间差”技术

- 商业公司
  - 定位服务(TAGGING)、追踪服务( TRACKING )、导航服务( TRACING )等

# LBS应用分类

- 面向用户 LBS
- 面向设备 LBS
- Push服务
- Pull服务

| | Push服务 | Pull服务 |
|---|---|---|
| 面向用户服务 | 当你进入某城市时接到欢迎信息 | 请求查找最近邻餐馆 |
| 面向设备服务 | 在货物追踪应用中，当货物运送偏离预计轨道时给与警报信息 | 请求查找卡车现在所在位置 |



# LBS与隐私

- 欧洲委员会
  Directive 2002/58/EC 条款9
  - Location data may only be processed when it is made anonymous OR with the consent of the user for the duration necessary for the provision of a service(位置数据只有在匿名或用户同意的前提下为有效并必要的服务使用).

- Vodafone UK制定了一套隐私管理业务条例(privacy management code of practice),要求所有为Vodafone客户提供服务的第三方必须遵守

# LBS中的隐私泄露

- 位置隐私泄露
  - 位置，包括用户过去或现在的位置
- 查询隐私泄露
  - 查询内容，例如查询距离我最近的艾滋医院

行为模式、兴趣爱好、健康状况和政治倾向等个人隐私信息

9 /47

# LBS中隐私保护

- 位置隐私保护
  - 避免用户与某一精确位置匹配
- 查询隐私保护
  - 避免用户与某一敏感查询匹配

10 /47

# 位置服务 VS 隐私保护

- 位置服务
  - 提供精确位置
- 隐私
  - 模糊用户的位置

鱼与熊掌可否兼得？

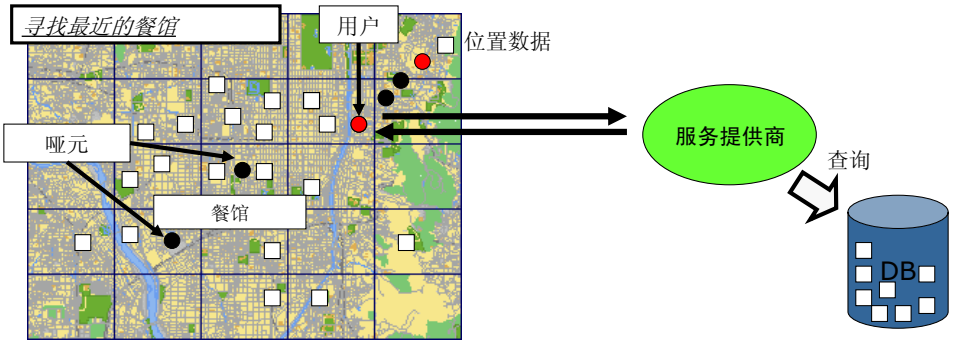



# 隐私保护的方法

- 假位置（Dummy）
- 时空匿名（Spatio-temporal Cloaking）
- 空间加密（Space Encryption）

# 隐私保护的基本方法--假位置

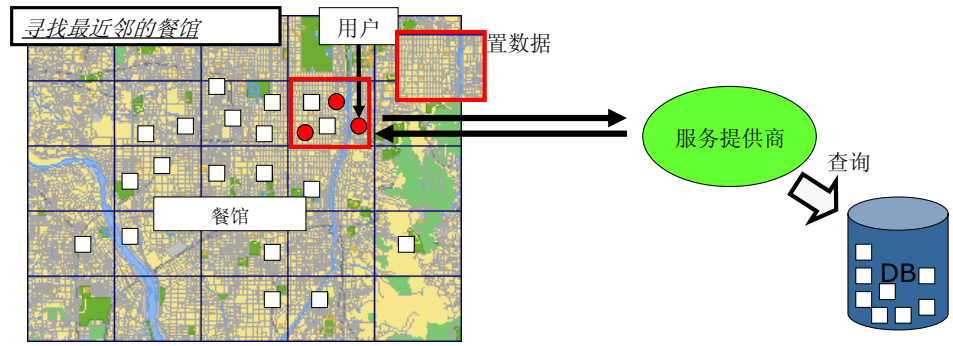
- 通过制造假位置，达到以假乱真的效果




13 /47

# 隐私保护的基本方法--时空匿名

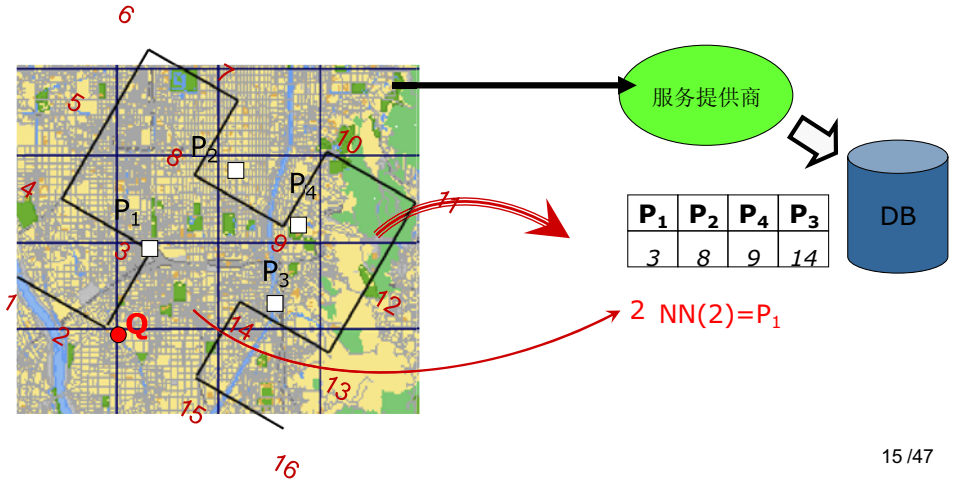
- 将一个用户的位置通过扩展变成时空区域，达到匿名的效果




14 /47

# 隐私保护的基本方法—空间加密

- 通过对位置加密从而达到匿名的效果




15 /47

# 感知隐私保护的查询处理

**假数据**

- 移动对象数据库中的查询处理器无需作任何修改

**时空匿名**

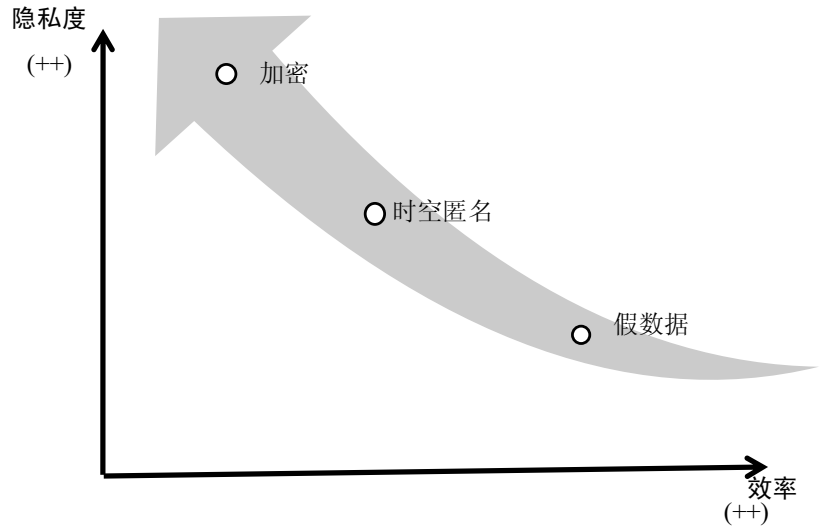
- 设计基于区域位置的查询处理技术；查询结果是一个包含真实结果的超集

**空间加密**

- 查询方法与使用的加密协议有关

16 /47

# 隐私度与效率对比




17 /47

# 存在的挑战


保护隐私与位置服务是一对矛盾


位置匿名的即时性


位置频繁更新以及位置依赖性


隐私需求个性化

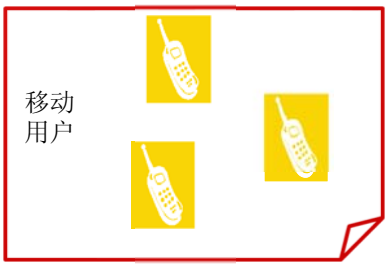
18 /47

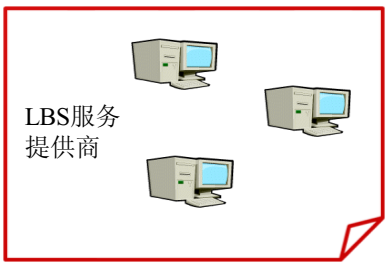
# 报告大纲

- 隐私保护问题及意义
- 隐私保护系统结构
- 隐私保护研究内容
- 隐私保护面临挑战
- 总结

19 /47

# 隐私保护系统结构

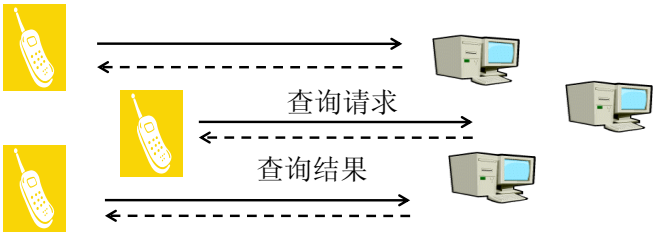



- 独立式结构
- 中心服务器结构
- 分布式结构
- 点对点结构

20 /47

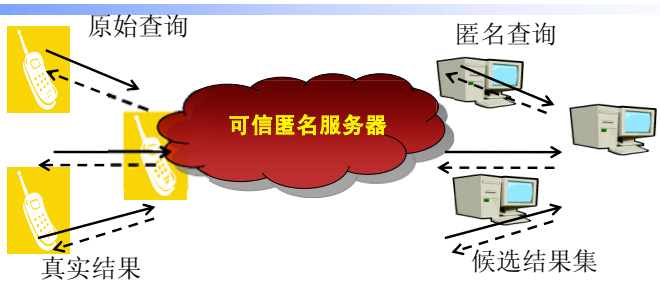
# 隐私保护系统结构-独立式结构




- 优点
  - 结构简单，易于配置
- 缺点
  - 增加客户端负担
  - 缺乏全局信息，隐蔽性弱

21 /47

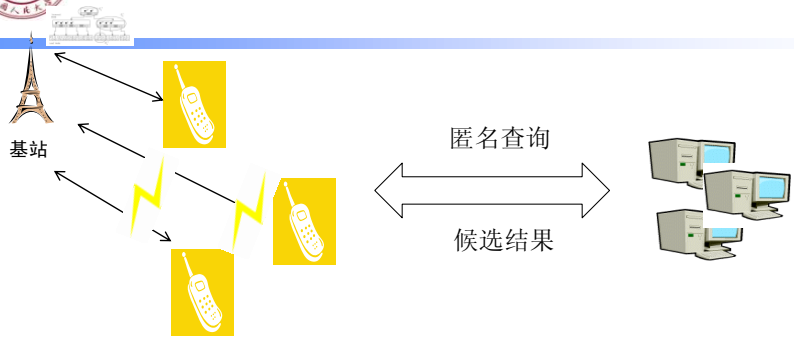
# 隐私保护系统结构-中心服务器结构




- 优点
  - 具有全局信息，隐私保护效果好
- 缺点
  - 可能成为系统瓶颈
  - 唯一攻击点

22 /47

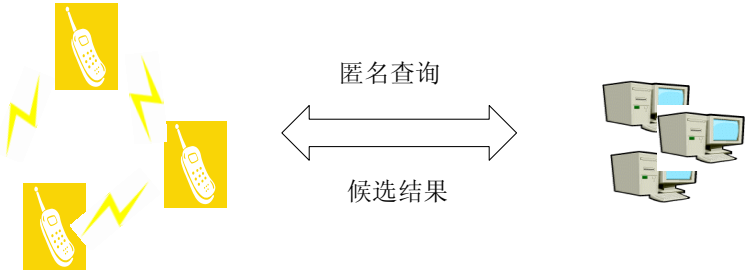
# 隐私保护系统结构-主从分布式结构




- 优点
  - 拥有全局信息，隐私效果好
  - 消除了系统瓶颈
- 缺点
  - 网络通讯代价高

23 /47

# 隐私保护系统结构-移动点对点结构




- 优点
  - 拥有全局信息，隐私效果好
  - 消除唯一攻击点
- 缺点
  - 网络通讯代价高

24 /47

# 报告大纲

- 隐私保护问题及意义
- 隐私保护系统结构
- 隐私保护研究内容
- 隐私保护面临挑战
- 总结



# 隐私保护研究内容

- 隐私保护方法
  - 位置隐私保护方法
  - 查询隐私保护方法
- 感知隐私的查询处理
  - 基于区域位置的查询处理技术
  - 基于加密位置的查询处理技术

# 隐私保护模型

- 位置 *k*-匿名
  - 当且仅当一个用户的位置与其他（*k*-1）个用户的位置无法区别时，称该用户满足*位置k-匿名*

原始查询

| 位置 | 查询 |
|---|---|
| (1, 6) | $Q_1$ |
| (1, 5) | $Q_2$ |
| (2, 9) | $Q_3$ |
| … | … |

匿名后查询

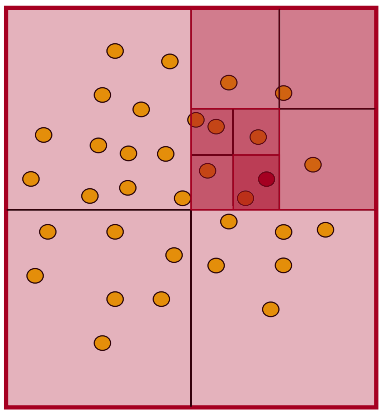
| 匿名位置 | 查询 |
|---|---|
| [(1,2)-(5, 9)] | $Q_1$ |
| [(1,2)-(5, 9)] | $Q_2$ |
| [(1,2)-(5, 9)] | $Q_3$ |
| … | … |

27 /47



# 基于四分树的隐私保护方法 (Quadtree based Cloaking)

- 问题
  - 面对大量移动用户，如何快速高效的为移动用户寻找匿名集
- 解决方法
  - 递归式的划分空间，直至在某一子空间内的用户数小于k，则返回其上一级的子空间作为位置匿名区域



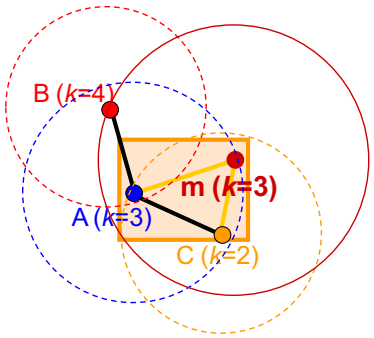
K=3

M .Gruteser,D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking.Proceedings of the International Conference on Mobile Systems, Applications, and Services(MobiSys'03), Scan Francisco, USA, 2003:163–168.

28 /47

# 个性化隐私需求匿名方法 (CliqueCloak)

- 问题
  - 如何为每一个用户提供满足个性化隐私需求的匿名
- 解决方法
  - 利用图模型形式化的定义此问题，并把寻找匿名集转化为在图中寻找k-点团的问题



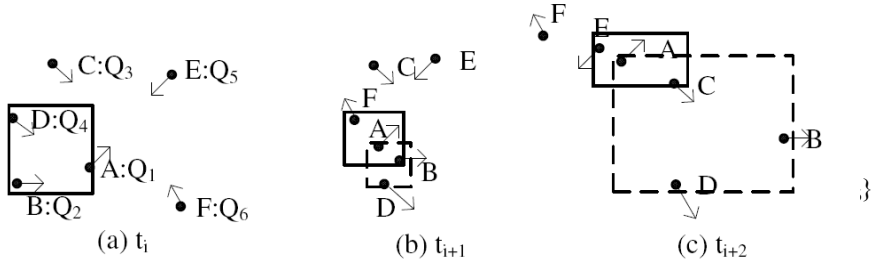

Gedik B, Liu L. Location privacy in mobile systems: a personalized anonymization model. Proceeding of the International Conference on Distributed Computing Systems(ICDCS'05), Columbus, OH, USA, 2005: 620–629





# 连续查询隐私保护技术（C.Chow and M. F. Mokbel,2007）

- 问题
  - 位置服务中现有的隐私保护工作均针对*snapshot*查询类型




- 解决方法
  - 连续查询的用户在最初时刻形成的匿名集在其查询有效期内均有效。

C. Chow and M. F. Mokbel. Enabling privacy continuous queries for revealed user locations. In *Proceedings of SSTD*,2007

# 感知查询差异性的隐私保护 (*p*-sensitive)

- 问题
  - 位置*k*-匿名只能防止用户与查询间的关联，但不能切断用户与查询内容的关联
  - 缺少语义的匿名，将产生查询隐私泄露的现象

| Location | Query |
|---|---|
| [(1,2)-(5, 9)] | Hospital |
| [(1,2)-(5, 9)] | Clinic |
| [(1,2)-(5, 9)] | Hospital |
| [(2,5)-(4, 7)] | Gas Station |
| [(2,5)-(4, 7)] | Gas Station |
| [(2,5)-(4, 7)] | School |

| Location | Query |
|---|---|
| [(1,2)-(4, 7)] | ** Club A |
| [(1,2)-(4, 7)] | Gas Station |
| [(1,2)-(4, 7)] | Gas Station |
| [(5,2)-(7, 9)] | Restaurant |
| [(5,2)-(7, 9)] | Clinic |
| [(5,2)-(7, 9)] | School |

- 解决方法
  - 考虑查询语义
  - 一个匿名集中所包含的敏感查询不能超过p%

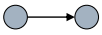
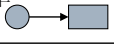
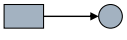
X.Zhen, J. Xu, and X. Meng. A Semantic Privacy-Protection Model for Location-based Services. In proceeding of MDM-PALM,2008

31 /47



# 感知隐私保护的查询处理

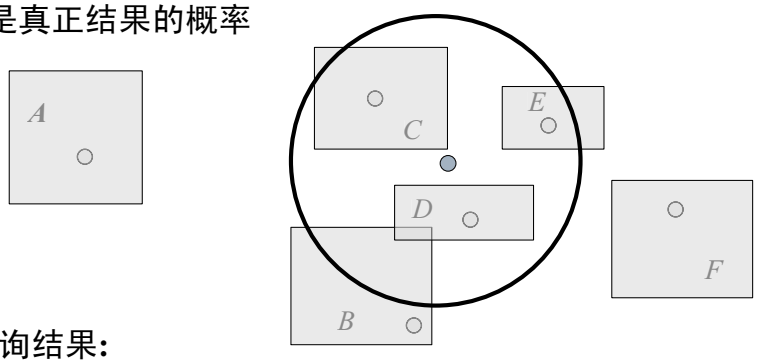
- 如何在位置被区域匿名后提供令用户满意的服务
- 两种位置数据类型：
  - 公开位置数据．如加油站、旅馆和警车
  - 隐私位置数据．如个人位置

| 查询类型 | | 被查询点 | |
|---|---|---|---|
| | | 公开数据 | 隐私数据 |
| 查询点 | 公开数据 | 基于公开数据的公开查询<br>如：在某电影院200m内所有餐馆  | 基于隐私数据的公开查询<br>如：某加油站500米内所有出租车  |
| | 隐私数据 | 基于公开数据的隐私查询<br>如：距离我最近的加油站  | 基于隐私数据的隐私查询<br>如：离我最近的朋友  |

32 /47

# 基于隐私数据的公开查询

- “某加油站500米内所有出租车”
- 将所有与查询范围相交的匿名区域都作为候选集。
- 根据匿名框与矩形框的重叠区域面积大小表示查询结果是真正结果的概率




查询结果：

- (B, 50%), (C, 90%), (D,1), (E,60%)

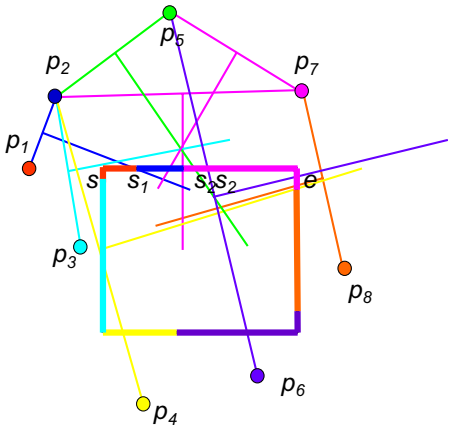
O. Wolfson, P.A. Sistla, S. Chamberlain, and Y. Yesha, Updating and Querying Databases that Track Mobile Units, Distributed and Parallel Databases, vol. 7, no. 3, pp. 257-387, 1999.





# 基于公开数据的隐私查询

- “距离我最近的加油站”
- 查找匿名区域中任意一个位置的最近邻
  - 查找匿名框所覆盖的对象
  - 查找基于每一条边上的任意点的最近邻
  - 二者的并作为最终的查询结果返回




H. Hu, D. Lee: Range Nearest-Neighbor Query. IEEE Trans. Knowl. Data Eng. 18(1): 78-91 ,2006

# 报告大纲

- 隐私保护问题及意义
- 隐私保护系统结构
- 隐私保护研究内容
- 隐私保护面临挑战
- 总结

35 /47

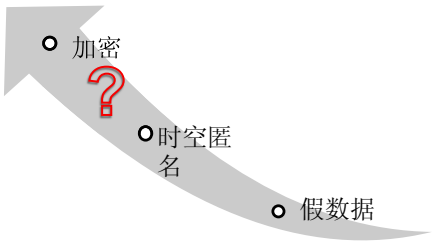
# 隐私保护技术面临的问题

- 多技术混合的隐私保护
- 移动对象轨迹的隐私保护
- 室内位置隐私保护

36 /47

# 多技术混合的隐私保护

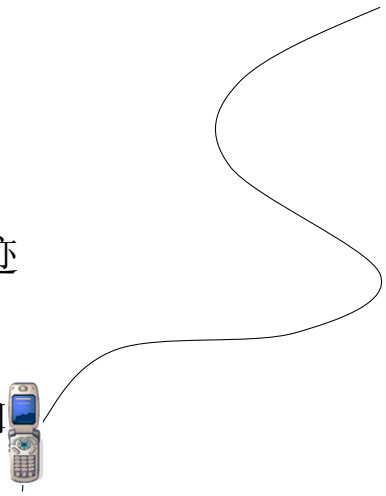
- 问题
  - 加密安全但查询代价高，时空匿名高效但不够安全
- 研究结合加密算法高隐私保护度，空间匿名算法的高效率的混合匿名模型和算法
- 研究基于混合匿名的感知隐私的查询处理算法




37 /47

# 移动轨迹的隐私保护

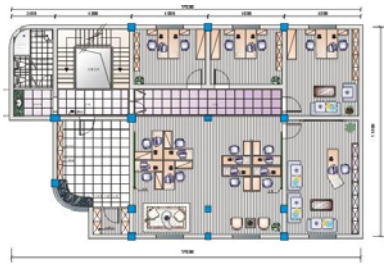
- 问题
  - 轨迹发布而产生的隐私泄露
- 研究基于时空匿名的轨迹匿名模型和算法
- 研究在线轨迹匿名模型和算法



38 /47

# 室内位置隐私

- 问题
  - 室内安装无限传感器收集用户位置用于安全控制、资源管理等
  - 室内提出的查询多位密度查询或聚类查询



- 研究基于室内位置隐私的攻击模型、匿名模型、匿名算法和查询处理算法

39 /47

# 可能的解决方法


40 /47

# 报告大纲

- 隐私保护问题及意义
- 隐私保护系统结构
- 隐私保护研究内容
- 隐私保护面临挑战
- 总结



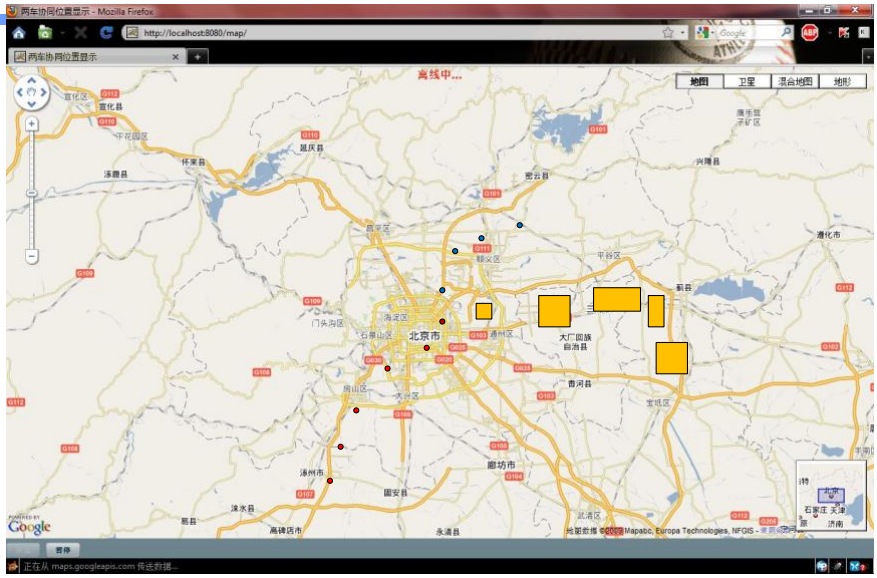
# 在研课题-隐私保护技术

- 国家863计划重点项目

“普适计算基础软硬件关键技术及系统”项目中“隐私保护技术”课题

- 研究目标
  - 在普适计算以人为中心的理念下，针对个人信息隐私、位置隐私和查询隐私等问题从模型、算法和评价等方面展开研究，开发可配置的分级隐私保护模块，为构建相应示范应用提供支持

# 在研课题-隐私保护技术




43 /47

# 在研课题-隐私保护技术

- **系统界面**


44 /47

# 参考文献


- O. Abul, F. Bonchi, and M. Nanni, Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases, In *Proc. of International Conference on Data Engineering (ICDE'08), pp.376–385, 2008.*
- B. Bamba and L. Liu, Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid, In *Proc. of International Conference on World Wide Web (WWW'08), 2008.*
- K. Bharath, G. Ghinita, and P. Kalnis .Privacy-Preserving Publication of User Locations in the Proximity of Sensitive Sites, *In Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)* , July 2008
- C. Bettini, X. S. Wang, and S. Jajodia, Protecting privacy against locationbased personal identification, In *Proc. of the VLDB Workshop on Secure Data Management (SDM'05), pp.185–199, 2005.*
- C. Chow and M. Mokbel. Privacy in Location-based Services: A System Architecture Perspective. *The SIGSPATIAL Special Newsletters, SIGSPATIAL Special,* Vol. 1, No. 2, pages 23-27, July 2009.
- C. Chow, M. F. Mokbel, and T. He, Tinycasper: a privacy-preserving aggregate location monitoring system in wireless sensor networks. In proceedings of SIGMOD08(demo), Pages 1307-1310 , Vancouver, Canada ,2008
- C. Chow and M. F. Mokbel, Enabling Privacy Continuous Queries for Revealed User Locations, In *Proc. of the International Symposium on Advances in Spatial and Temporal Databases (SSTD'07), 2007.*
- Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. *In Proceedings of the ACM Symposium on Advances in Geographic Information Systems, ACM GIS,* Arlington, VA, November 2006:171-178
- R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, Preserving User Location Privacy in Mobile Data Management Infrastructures, In *Proc. of Privacy Enhancing Technology Workshop (PET'06), 2006.*
- J. Du, J. Xu, X. Tang, and H. Hu. iPDA: Enabling Privacy-Preserving Location-based Services. In *Proc. of the International Conference on Mobile Data Management (MDM'07), 2007.*
- Electronic Frontier Foundation, *http://www.eff.org/issues/privacy*, December 01,2009
- G. Ghinita ,Understanding the Privacy-Efficiency Trade-off in Location-Based Queries*, ACM SIGSPATIAL GIS Workshop on Security and Privacy in GIS and LBS (SPRINGL), November 2008*
- *G.* Ghinita, Private Queries and Trajectory Anonymization: a Dual Perspective on Location Privacy*, Transactions on Data Privacy (TDP) 2009*


45 /47

# 参考文献


- G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan,Private queries in location based services: anonymizers are not necessary*, In Proc. of SIGMOD'08,* Vancouver, Canada, 2008.
- *G.* Ghinita, P. Kalnis and S. Skiadopoulos, MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries, *In Proceedings of International Symposium on Spatial and Temporal Databases (SSTD*), July 2007
- G. Ghinita, M. L. Damiani, and C. Silvestri, Preventing Velocity-based Linkage Attacks in Location-Aware Applications, In *Proc. of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems 2009 (ACM GIS'09), 2009.*
- M. Gruteser. and D. Grunwald, Anonymous Usage of Location-based Services Through Spatial and Temporal Cloaking, In *Proc. of the International Conference on Mobile Systems, Applications, and Services (MobiSys'03), pp.163–168, 2003.*
- G. Gidofalvi, X. Huang, and T. B. Pedersen, Privacy-preserving Data Mining on Moving Objects Trajectories, *In Proc. of the International Conference on Mobile Data Management (MDM'07*), 2007.
- B. Gedik and L. Liu, Location Privacy in Mobile Systems: A Personalized Anonymization Model, *In Proc. of the International Conference on Distributed Computing Systems (ICDCS'05),* 2005.
- Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. PRIVE: Anonymous Location based Queries in Distributed Mobile Systems. *In Proceedings of International Conference on World Wide Web, WWW*, pages 1–10, 2007.
- Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. MOBIHIDE: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries. *In Proceedings of the International Symposium on Advances in Spatial and Temporal Databases, SSTD*, 2007.
- GPS and Privacy Rights, The New Yorks Times, http://www.nytimes.com/2009/11/23/opinion/23mon3.html?_r=1, November 22, 2009
- H. Hu and J. Xu, Non-exposure Location Anonymity, *In Proc. Of ICDE'09*, 2009.
- http://wiki.media-culture.org.au/index.php/GPS_-_Privacy_Issues,29 Oct 2004
- Hu, H., Xu, J., Du, J., Ng, J.K.Y.: Privacy-Aware Location Publishing for Moving Clients. Technical report, Hong Kong Baptist University (2007) http://www.comp.hkbu.edu.hk/~haibo/privacy join.pdf.


46 /47

# 参考文献


- H. Hu and D. Lee, Range Nearest-neighbor Query, *IEEE Transactions on Knowledge and Data Engineering (TKDE),* vol.18, no.1, pp.843–854, 2006.
- H. Kido, Y. Yanagisawa, and T. Satoh,"Protection of location privacy using dummies for location-based services," *Proc.the 25th International Conference on Distributed Computing Systems(ICPS'05)*, 2005.
- H. Lu, C. S. Jensen, and M. L. Yiu, "A3D : anonymity area aware, dummy-based location privacy in mobile services," *Proc. 7th International ACM Workshop on Data Engineering for Wireless and Mobile Access(MobiDE'08*), 2008.
- L. Liu, From Data Privacy to Location Privacy: Models & Algorithms, *In VLDB07* ,2007.
- S. Mascetti, C. Bettini, X. S. Wang, D. Freni, and S. Jajodia, Preserving Anonymity in Location-based Services When Requests from the Same Issuer May be Correlated, *TR*, University of Milan, Italy, 2007.
- M. F. Mokbel, C. Y. Chow, and W. G. Aref, The New Casper: Query Processing for Location Services without Compromising Privacy, *In Proc. of the 32nd International Conference on Very Large Data Bases (VLDB'06),* 2006.
- M. F. Mokbel. Privacy in Location-Based Services: State of Art and research directions. In *MDM07*, 2007.
- X. Pan, X. Meng, J. Xu: Distortion-based Anonymity for Continuous Query in Location-Based Mobile Services. *In the proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009)*, November 4-6, 2009, Seattle, Washington.
- X. Pan, J. Xu, X. Meng: Protecting Location Privacy against Location-Dependent Attack in Mobile Services. *In Proceedings of the ACM 17th Conference on Information and Knowledge Management(CIKM2008)*, page 1475-1476, Napa Valley, California, October 26-30, 2008.
- L. Sweeney, K-anonymity: A Model for Protecting Privacy, International Journal on Uncertainty, *Fuzziness and Knowledge-based Systems*, vol.10, no. 5, pp.557–570, 2002.
- H. Shin, V. Atluri, and J. Vaidya, A Profile Anonymization Model for Privacy in a Personalized Location Based Service Environment, *In Proc. of the 9th International Conference on Mobile Data Management (MDM'08*), 2008.
- T. Xu and Y. Cai, Location Anonymity in Continuous Location-based Services, *In Proc. of GIS'07*, 2007.
- J. Xu, X. Tang, H. Hu, and J. Du, Privacy-Conscious Location-Based Queries in Mobile Environments, *IEEE Transactions on Parallel and Distributed Systems (TPDS),* accepted to appear, 2009.

# 云数据管理系统：挑战与机遇

陆嘉恒

随着信息产业的发展，企业和公司产生的数据量快速增长，通常数据规模可以达到 TB 甚至 PB 级别。如何管理和分析海量数据是目前很多领域所面临的问题，例如在医疗、通信和互联网领域。传统的数据管理技术已经不能完全满足海量数据管理的需求，云计算技术的出现为海量数据管理带来了机遇，利用云平台来存储和管理海量数据是当前的研究热点之一。该 ppt 主要介绍了云数据管理方面的挑战和机遇，包括云数据管理的必要性和需求，数据管理的弱一致性研究，数据拷贝的一致性维护，数据的并发处理和版本模型等前沿数据库研究问题。

# Cloud-based Data Management: Challenges & Opportunities

Jiaheng Lu

Renmin Universtiy of China



## Research experience and interesting

- **National University of Singapore PhD**
  - XML query processing and XML keyword search
- **University of California, Irvine Postdoc**
  - Approximate string processing
  - Data integration and data cleaning
- **Renmin University of China**
  - Cloud data management
  - XML data management

## Outline

- **Motivation: cloud data management**
- **Database Future and Challenges：**
  - Large-scale Data management & transaction processing
  - Cloud-based data indexing and query optimization



## Motivation: Internet Chatter

## BLOG Wisdom

- **"If you want vast, on-demand scalability, you need a non-relational database." Since scalability requirements:**
  - Can change very quickly and,
  - Can grow very rapidly.
- **Difficult to manage with a single in-house RDBMS server.**
- **Although RDBMS scale well:**
  - When limited to a single node.
  - Overwhelming complexity to scale on multiple sever nodes.



## Current State

- **Most enterprise solutions are based on RDBMS technology.**
- **Significant Operational Challenges:**
  - Provisioning for Peak Demand
  - Resource under-utilization
  - Capacity planning: too many variables
  - Storage management: a massive challenge
  - System upgrades: extremely time-consuming

## What is Cloud Computing?

- **Old idea: Software as a service (SaaS)**
  - Def: delivering applications over the internet
- **Recently: "[Hardware, infrastructure, Platform] as a service"**
  - Poorly defined so we avoid all "X as a service"
- **Utility Computing: pay-as-you-go computing**
  - Illusion of infinite resources
  - No up-front cost
  - Fine-grained billing (e.g. hourly)

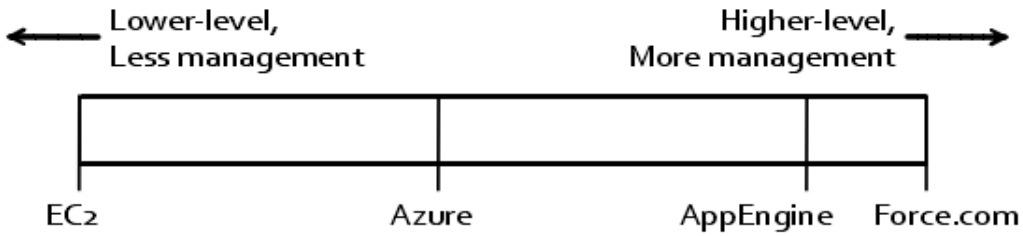


## Why Now?

- Experience with very large datacenters
  - Unprecedented economies of scale
- Other factors
  - Pervasive broadband internet
  - Pay-as-you-go billing model

## Cloud Computing Spectrum

- Instruction Set VM (Amazon EC2, 3Tera)
- Framework VM
  - Google AppEngine, Force.com



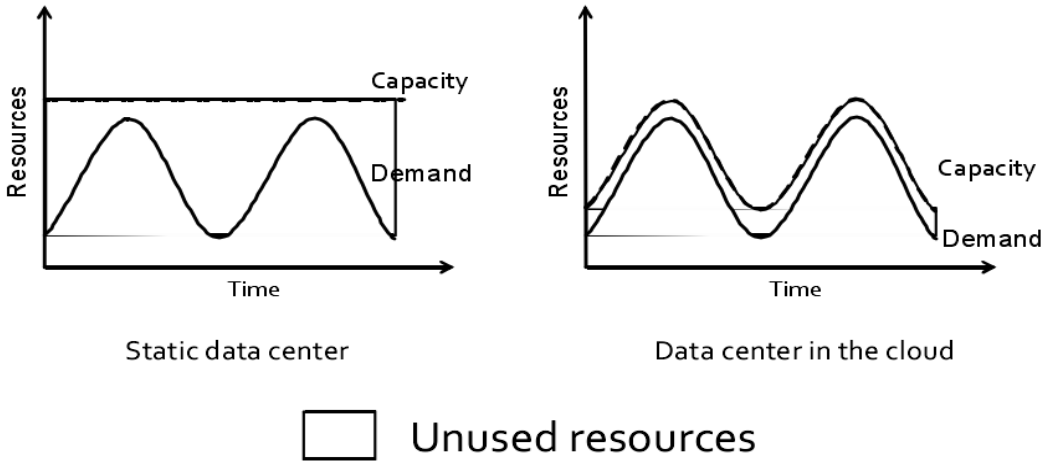



## Cloud Killer Apps

- Mobile and web applications
- Extensions of desktop software
  - Matlab, Mathematica
- Batch processing/MapReduce

## Economics of Cloud Users

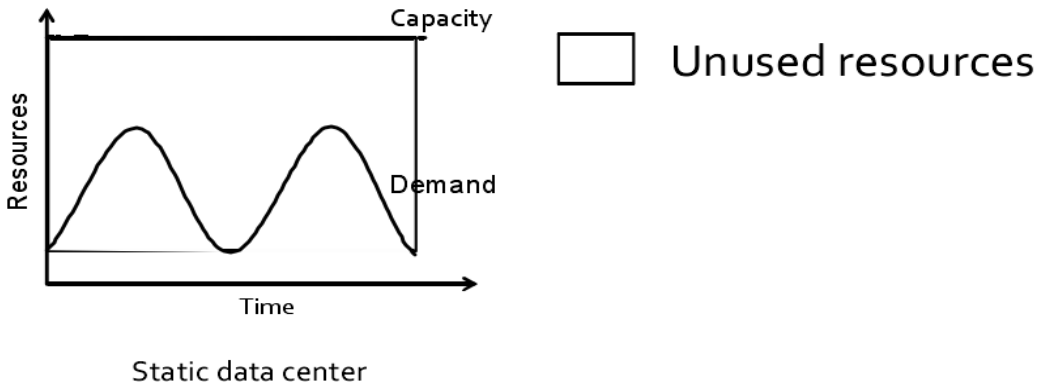
- Pay by use instead of provisioning for peak




11 /29

## Economics of Cloud Users

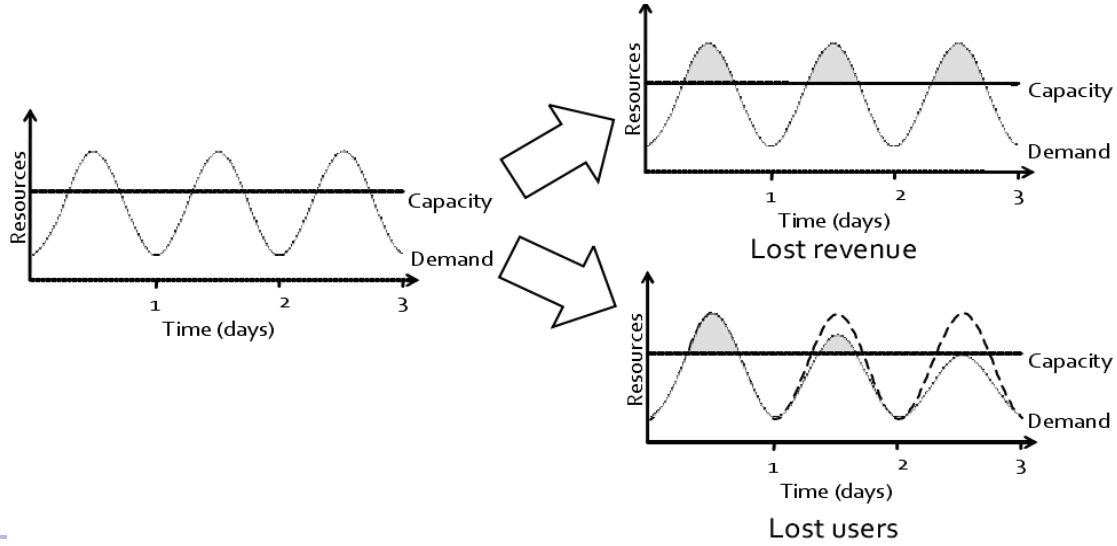
- Risk of over-provisioning: underutilization




12 /29

## Economics of Cloud Users

- Heavy penalty for under-provisioning






## Engineering Definition

- Providing services on virtual machines allocated on top of a large physical machine pool.

## Business Definition

- A method to address scalability and availability concerns for large scale applications.



## Data Management in the Cloud?

## The Vision

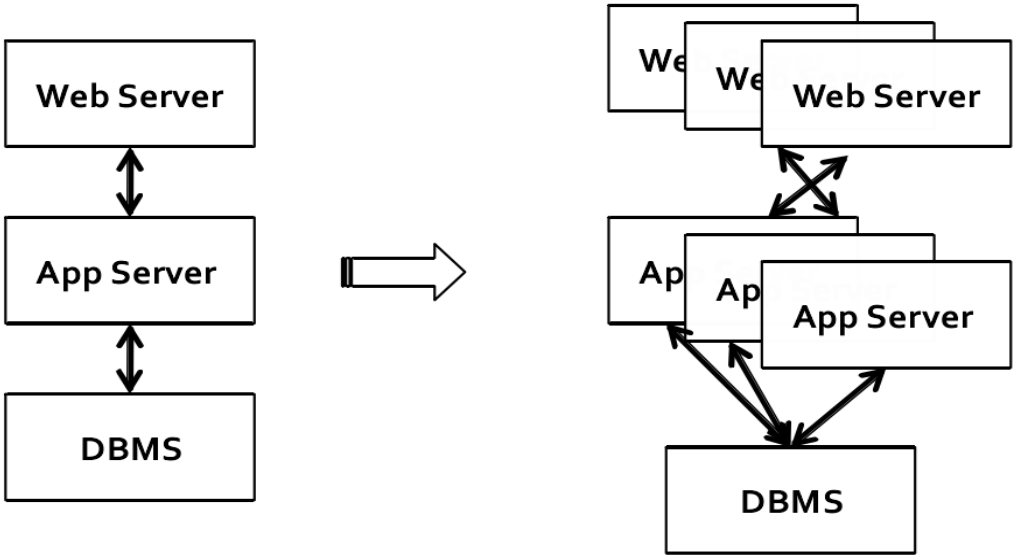
- R&D Challenges at the macro level:
  - Where and how does the DBMS fit into this model.
- R&D Challenges at micro level:
  - Specific technology components that must be developed to enable the migration of enterprise data into the clouds.



## Data and Networks: Attempt Ⅰ

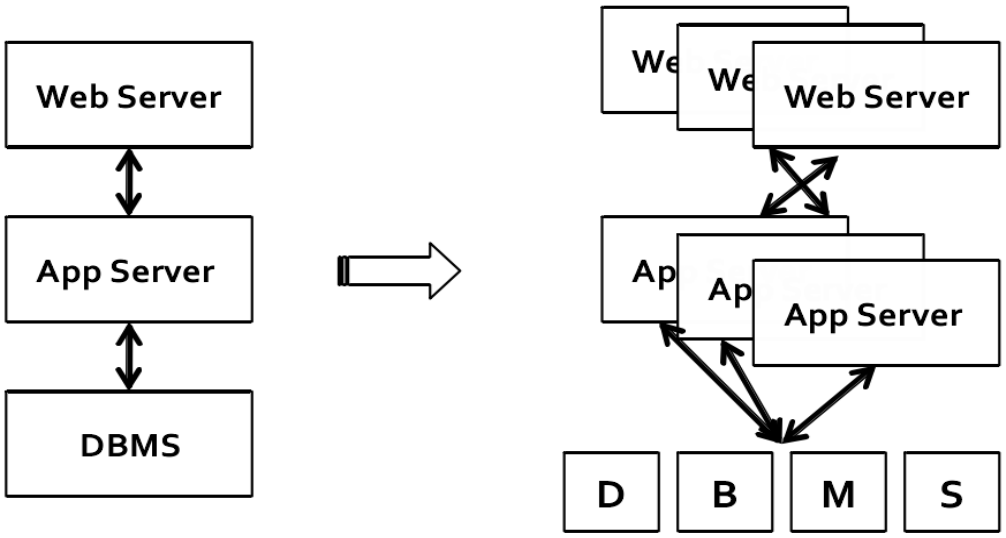
- Distributed Database (1980s):
  - Idealized view: unified access to distributed data
  - Prohibitively expensive: global synchronization
- Remained a laboratory prototype:
  - Associated technology widely in-use: 2PC

## Data and Networks: Attempt II






## Data and Networks: Pragmatics



## Database on S3: SIGMOD'08

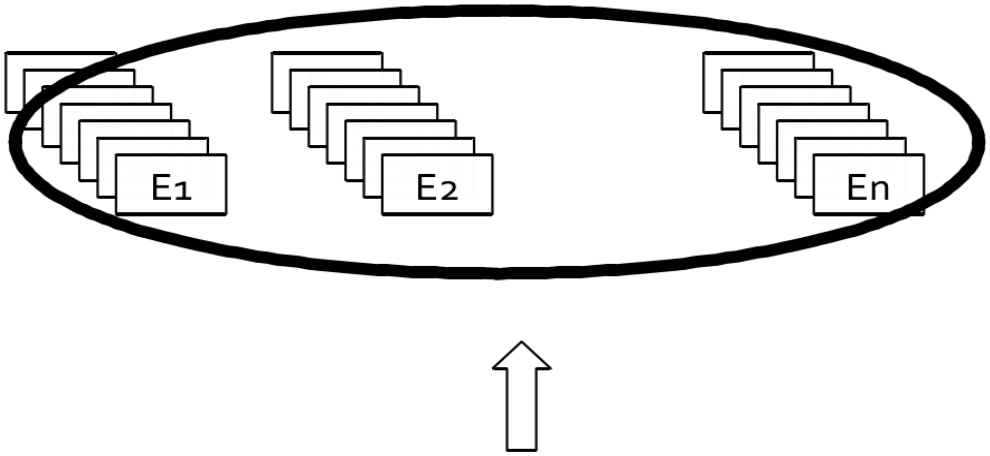
- Amazon's Simple Storage Service(S3):
  - Updates may not preserve initiation order
  - No "force" writes
  - Eventual guarantee
- Proposed solution:
  - Pending Update Queue
  - Checkpoint protocol to ensure consistent ordering
  - ACID: only Atomicity + Durability



## Unbundling Txns in the Cloud

- **Research results:**
  - CIDR'09 proposal to unbundle Transactions Management for Cloud Infrastructures
  - Attempts to refit the DBMS engine in the cloud storage and computing

## Analytical Processing




**Analysis Queries: Distributed Processing**



## Architectural and System Impacts

- **Current state:**
  - MapReduce Paradigm for data analysis
- **What is missing:**
  - Auxiliary structures and indexes for associative access to data (i.e., attribute-based access)
  - Caveat: inherent inconsistency and approximation
- **Future projection:**
  - Eventual merger of databases (ODSs) and data warehouses as we have learned to use and implement them.

## Underlying Principles: CIDR'2009

- **Business data may not always reflect the state of the world or the business:**
  - Inherent lack of perfect information
- **Secondary data need not be updated with primary data:**
  - Inherent latency
- **Transactions/Events may temporarily violate integrity constraints:**
  - Referential integrity may need to be compromised

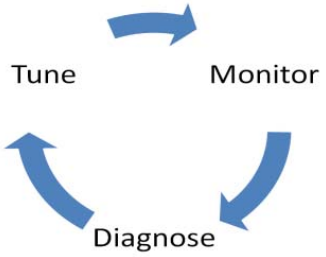


## Data Security & Privacy

- Data privacy remains a show-stopper in the context of database outsourcing.
- Encryption-based solutions are too expensive and are projected to be so in the foreseeable future:
  - Private Information Retrieval (Sion'2008)
- Other approaches:
  - Information-theoretic approaches that uses data-partitioning for security (Emekci'2007)
  - Hardware-based solution for information security

## Self management and self tuning in cloud-based data management

- Self management and self tuning




- Query optimization on thousands of nodes



## Remarks

- **Data Management for Cloud Computing poses a fundamental challenge to database researchers:**
  - Scalability
  - Reliability
  - Data Consistency
- **Radically different approaches and solution are warranted to overcome this challenge:**
  - Need to understand the nature of new applications

# References


- **Life Beyond Distributed Transactions: An Apostate's Opinion by P.Helland, CIDR'07**
- **Building a Database on S3 M.Brartner, D.Florescu, D.Graf, D.Kossman, T.Kraska, SIGMOD'08**
- **Unbundling Transaction Services in the Cloud D.Lo,et, A.Fekete, G.Weikum, M.Zwilling, CIDR'09**
- **Principles of Inconsistency S.Finkelstein, R.Brendle, D.Jacobs, CIDR'09**
- **VLDB Database School (China) 2009 http://www.sei.ecnu.edu.cn/~vldbschool2009/VLDBSchool2009 English.htm**

# Web Data Management for Mobile Users

Zheng Huo, Jing Zhao, Xiangmei Hu
huozheng123@gmail.com

## 1. Introduction

Mobile devices are becoming increasingly popular as a means of information access while on-the-go. With the emergence of web access friendly mobile devices, the number of mobile users who will access the web using their mobile devices is expected to increase drastically in the near future. Meanwhile most Web data are stored in millions of deep web data sources which can be accessed by desktop and also mobile users, however, the mobile users have other needs, or maybe they can't access deep web data conveniently as desktop users, such as the terminal have small screen, and the input capabilities is not as strong as desktop users. Sometimes mobile users' information needs are more location sensitive than desktop users. So the challenge is how to provide useful and convenient services for mobile users. In our survey of this topic, we found several questions to be solved, and we proposed an initial framework for web data management for mobile users.

## 2. Features of mobile users

There have bee several large scale examinations for user search behavior through search engine logs for both computer and mobile search. The result of this analysis have been used to improve performances of mobile users' access to the deep web.

**Shorter queries** - As analysis shows, the query length of the mobile users is shorter than desktop users. For computer-based search, the average number of words per query is 2.93 and the average number of characters per query is 18.72. The length of conventional mobile phone queries is the shortest of all the mediums, with an average query consisting of 2.44 words and 15.89 characters. The shorter query terms can be easily understood since the limitation of input function in mobile devices.

**Information needs** - Mobile users look for very different topics than standard desktop web users. Researchers find that the most popular mobile topics are local services and travel & commuting.

**Location of mobile users** - There is strong evidence indicating that location-based searches are popular among mobile searchers. By taking into account of users' location information, we can provide more personalized services.

**Small screen size** - This makes it difficult or impossible to see text and graphics dependent on the standard size of a desktop computer screen. So what kind of integrated interface is suitable for mobile users is a challenging problem.

**Lack of windows** - On a desktop computer, the ability to open more than one window at a time allows for multi-tasking and for easy revert to a previous page. On mobile devices, only one page can be displayed at a time, and pages can only be viewed in the sequence they were originally accessed.

**Computing and memory limits** - Most of them have slow computing speed and small storage capacity which restricts spatial search calculations, routing operations and the creation of a user specific "mobile" map.

**Type limitation of accessible pages** - Many sites that can be accessed on a desktop cannot on a mobile device. Many devices cannot access pages with a secured connection, Flash or other similar software, PDFs, or video sites, although recently this has been changing.

**Lower speed** - On most mobile devices, the speed of service is very slow, often slower than dial-up Internet

access.

**Compressed pages** - Many pages, in their conversion to mobile format, are squeezed into an order different from how they would customarily be viewed on a desktop computer.

**Size of messages limits** - Many devices have limits on the number of characters that can be sent in an email message.

**Expensive cost** - the access and bandwidth charges levied by cell phone networks are much, much higher than those for fixed-line internet access.

## 3. Main framework

We will introduce some web data integration issues in this part, this is specially for mobile users based on the behavior analysis and features of mobile users above. Figure 1 shows the main part of the deep web data integration modules for mobile users. Following are the functions of each part.

**WDS discovery**- Discovering accessible web databases in the web.

**Interface clustering**- This part classifies web data sources according to their domains.

**Interface analysis**- Analyzing and extracting the schema information in query interfaces.

**WDS profile (interface)**- Meta information about WDS query interface, including attribute, type, etc.

**Interface integration**- Integrating interfaces of several WDS to a global integrated interface

**Domain selection**- Specifying a suitable domain for users.

**Query predicate match**- Matching queries submitted to the easy query interface to the integrated interface

**WDS selection**- Selecting suitable web data sources for users

**Query translation**- Translating user queries to local queries.

**WDS connection**- Submitting queries to WDS

**WDS content analysis**- Analyze content of WDS

**WDS profile (content)**- Meta data of WDS, including scale of a Web database, distribution of values in each attributes.

**Result extraction**- Getting results from web pages

**Result annotation**- Finishing semantic annotation of the results

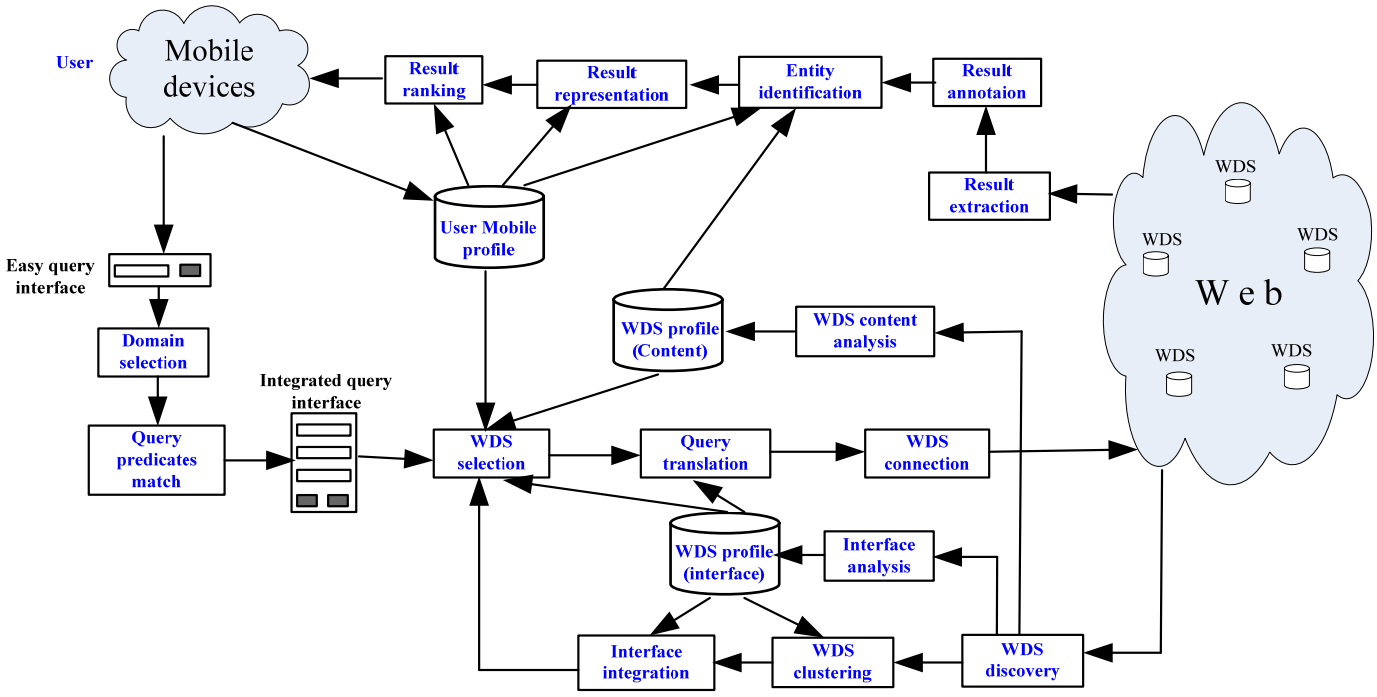



**Fig.1 Main Framework**

**Entity identification**- identifying records that are describing the same real word entity.

**Result representation**- Showing the results, including contents and layout of the presentation

**Result ranking**- Ranking the results according to the user context.

**User mobile profile**- information about users, including the screen size of user devices, computing resources and the location of users.

In the framework, when a mobile user input a query term to a easy query interface, the query is sent to the search domain selection part, which will select the most related domain according to the users' query terms. Then the query is sent to a integrated query interface. From the integrated interface, the query is sent to a traditional deep web data integration steps, such as ,WDS selection ,query translation and so on. What is special for mobile users is that, before the process of deep web data integration issues, the user mobile profile is introduced. As we showed above, the user mobile profile is key context for mobile searching, since it stores information about the users—location of users, computing ability, screen size of user devices etc. So the information can be used in the query processing, we will introduce it in detail in next section. After processing of the query, the result extraction will extract results from various web pages and send the collected results to the result annotation. The results are combined with some semantic meanings in this part. After entity identification, result representation and result ranking, the final results is sent to the users. In order to display more information in a small screen in a mobile device, it is always useful to have a result cluster in the system. The result cluster will cluster the results in hierarchy, each hierarchy is about a same topic, this is not yet included in the architecture.

Following are some topics on the frameworks above. Not every module of the framework is discussed in detail, since some of the modules are already been maturely researched. Topics from 3.1 to 3.4 are mainly concerned on web data integration issues, and the following four topics focused more on mobile issues.

## 3.1 Web database selection

(a) WDB selection based on content

When a query is submitted to an integrated interface, it needs to be passed to the Web data sources (WDSs) represented by the integrated interface. If the number of WDSs for this integrated interface is small, the query can be passed to all of them. However, if the number is large, it may be inefficient to invoke these WDSs for each query. Metasearch engines involving text search engines, only scattered work has been reported when deep web WDSs with structured data are involved. Metasearch engines involves only text documents and the representative of each search engine contains terms and some statistics for each term. In contrast, WDBs involve three types of attributes, i.e., textual, categorical and numerical attributes. Categorical attributes usually have a small number of distinct values and they are usually implemented as a selection list or a group of checkboxes or radio buttons on a search interface. The former can be considered as a special case of the latter when the structured data have just one textual attribute. The representative for each type of attribute may be different. We aimed at when the query interface has various attributes or some attributes is missed, it is necessary to find representations of WDBs and find useful methods based on this.

(2) WDB selection based on the location of mobile users

When it comes to mobile users, the location information is an important information. Ranking WDBs according to users' location can help users to get what they want.

(3) Service area identification of WDBs

We need to identify the service areas of a WDB. In order to match a mobile user's location with the information provided by a WDB, it is desirable to find out the intended service areas of WDBs. Some WDBs have very narrow service areas, some have multiple service areas, and some even have national or international coverage. We plan to study how to identify the service area of specialized local WDBs.

### 3.2 Entity identification across multiple deep web data sources

Entity identification is to determine if two or more records retrieved from different data sources actually correspond to the same real world entity. This is critical in several application scenarios in deep Web data integration. For example, in comparison shopping, it makes sense to compare the prices of two product records only if the two product records correspond to the same one. A general method for determining whether two records R1 and R2 are matched consists of two steps. First, values in corresponding attributes from R1 and R2 are matched. Specifically, for each attribute A, a similarity between R1[A] and R2[A] is computed. Second, the similarities between value pairs under all attributes are aggregated to determine whether R1 and R2 are matched.

Many researches have been done on entity identification, here, there are some initial thoughts on new method of doing this. For attribute value matching, we plan to develop a library of domain specific string matching functions.

### 3.3 Geo information on web pages

As we have analyzed above, mobile users is always "on-the-go" when they access to the web, another important feature is that mobile users search for location based information much more frequently than desktop users. One interesting issue is the problem of associating an address to each result or web page. Many web pages are associated with an organization or a unit of an organization. As a result, the address of the organization or the unit, whichever is more directly related to the page, can be considered as the address of the page, or the geo information of the web page, when the page itself does not contain an address, we can check if there are other implicit information which may contain geo information. We are interested in determining what address each page should be associated to. We can extract and index location information embedded in these resources, so it is easy for mobile users to receive the right location information from the web pages.

### 3.4 Search result extraction wrapper generation and maintenance

After the query is evaluated, the retrieved search result records are embedded in dynamically generated response pages. Specifically, there are two tasks – one is result extraction which is to extract the SRRs from the response pages and the other is result annotation which is to assign semantic meanings to the data units/instances within each SRR. The second task in turn consists of two subtasks, the first one is data alignment which aligns/groups data units from different SRRa on the same result page according to their semantics and the second one is data annotation which assigns a semantic label to each group of data units. As different search engines usually organizes and displays their SRRs differently and the SRRs returned by different search engines, even from the same domain, often consist of different types of information, different result extraction, data alignment and data annotation rules are needed for different search engines. Because millions of search engines are present on the Web and they frequently change their result display formats, highly automated solutions are needed to generate and maintain these wrappers.

For result extraction, we plan to carry out research in two directions. The first is to improve visual-feature based solution so that response pages where SRRs are organized into multiple columns and multiple sections can be handled accurately and the time needed to perform the extraction can be significantly reduced. The second is to combine visual-features and non-visual-features in a way that can maximize their contributions to accurate result extraction. For data alignment and data annotation, we plan to find solutions for the problems caused by attributes with multiple values or nearly identical values. We also plan to create a library of patterns for some common values such as email, telephone number, address, etc.

### 3.5 Location sensitive retrieval

For mobile search, a relevant result must match the user query by content and is close to the user's location. We plan to take WDS's service areas into consideration when performing search engine selection.

Furthermore, for results returned from local WDSs, we will try to identify the address associated with each result and perform location-sensitive result merging. A mobile user is more likely to prefer products/services close to his current location. The locations of mobile users can be determined by the mobile service provider when the mobile devices are in use.

A typical scenario is like this, suppose a mobile user is searching for information of the nearest restaurant, he not only needs the way to the restaurant or how the reach the nearest restaurant, he also wants more information about the restaurant, such as, services, prices or other guests' opinions. In this situation, traditional location servers or web servers can not provide services like this, so it is our motivation to do the research on web data integration for mobile users.

### 3.6 Search result clustering

As many analyses shows, mobile users tend not to "click" the search results, because it is not convenient for mobile users to "click into" a result, also, since the limitation of navigation systems in mobile devices, users do not always concern the search results which have bad ranking results. So it is important to improve the search result clustering methods.

Some researches have been done on search result clustering. In [4] they proposed a method to tackle the problem of mobile search using search result clustering, which consists of organizing the results obtained in response to a query into a hierarchy of labeled clusters that reflect the different components of the query topic.

By clustering the results, one single "page" on mobile device can display more information, it can improve users' experience in mobile search.

### 3.7 Concise snippet generation

Mobile devices usually have a small display screen, limiting the amount of information that can be displayed. Many investigation shows that, mobile users do not always "click" the search results, since the communication networks is slower than desktop users. So if shorter snippets for search results can be generated, then more results can be displayed on each screen, leading to better user experience. In this project, we are interested in reducing the size of the snippet returned from a WDS without compromising the effectiveness of the snippet in helping the user determine the usefulness of the result.

### 3.8 Result representation

Mobile devices have much more different user interface than desktop devices. Designing an effective mobile search users interface is challenging, as interacting with the results is often complicated by the lack of available screen space and limited interaction methods. In [7], the author proposed a method which can automatically compute categories to present the user with an overview of the result set.

## 4 Conclusions

In this paper, we figured out some research point on deep web data integration for mobile users. We are more concerned on the Geo information extraction on the web pages and the location sensitive retrieval during the process. The availability of location-driven data, location-enabled devices, and location application is guaranteed to expand the opportunities that exist in the combination of mobile users and the web. We proposed an initial framework based on the metasearch method, it will be optimized and extended in the future in order to deal the problems of location sensitive processes which is more concerned nowadays.

## References

[1] Kamvar, M., Kellar, M., Patel, R., and Xu, Y. 2009. Computers and iphonesand mobile phones, oh my!: a logs-based comparison of search users on different devices. In Proceedings of the 18th international Conference on World Wide Web (Madrid, Spain, April 20 -24, 2009). WWW '09.

[2] Kamvar, M. and Baluja, S. 2008. Query suggestions for

mobile search: understanding usage patterns. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy, April 05 -10, 2008). CHI '08. ACM, New York, NY, 1013-1016.
[3] Maryam Kamvar, Shumeet Baluja, Deciphering Trends in Mobile Search, Computer, v.40 n.8, p.58-62, August 2007
[4] Claudio Carpineto, Sefano Mizzaro, Mobile Information Retrieval with Search Results Clustering: Prototypes and Evaluations, ASIS 2008
[5] Church, K. and Smyth, B. 2009. Understanding the intent behind mobile information needs. In Proceedings of the 13th international Conference on intelligent User interfaces (Sanibel Island, Florida, USA, February 08 -11, 2009). IUI '09. ACM, New York, NY, 247-256.
[6] Christopher Jones, Christopher Jones, Location based Advertising, M-bussiness 2002
[7] Heimonen, T. and Käki, M. 2007. Mobile findex: supporting mobile web search with automatic result categories. In Proceedings of the 9th international Conference on Human Computer interaction with Mobile Devices and Services (Singapore, September 09 -12, 2007). MobileHCI'07, vol. 309. ACM, New York, NY, 397-404.
[8] Kamvar, M. and Baluja, S. 2008. Query suggestions for mobile search: understanding usage patterns. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy, April 05 -10, 2008). CHI '08. ACM, New York, NY, 1013-1016.
[9] W. Liu, X. Meng, and W. Meng. A Survey of Deep Web Data Integration. Chinese Journal of Computers, Vol.30, No.9, pp.1475-1489, September 2007.
[10] Y. Lu, H. He, Q. Peng, W. Meng, and C. Yu. Clustering E-Commerce Search Engines based on their Search Interface Pages using WISE-Cluster . Data & Knowledge Engineering (DKE) Journal, Vol.59, No.2, pp.231-246, November 2006.
[11] Y. Lu, H. He, H. Zhao, W. Meng, and C. Yu. Annotating Structured Data of the Deep Web. IEEE 23rd International Conference on Data Engineering (ICDE), 2007.

# 数据空间研究进展

## 1 引言

2007 年 7 月，我们开始承担 863 课题“海量数据空间模型、索引与查询技术研究”。本课题旨在研究海量数据空间的理论方法和实现技术，在数据空间模型、组织与分类、演化集成、查询优化等核心技术方取得进展。在此基础上开发具有自主知识产权的数据空间管理原型系统。2007 至 2008 年，我们在数据空间模型及查询技术方面进行了研究，取得了一些成果，并开发了两个数据空间原型系统：计算机科技文献集成系统 C-DBLP 和个人数据空间管理原型系统 OrientSpace。在这些成果的基础上，2009 年针对数据空间的演化集成、查询优化等问题进行了研究，基于取得的研究成果完成了相关专利的申请，并对两个数据空间原型系统进行了升级。图 1 是我们在 2008 年提出的数据空间管理系统框架。结合这一框架，对 2009 数据空间项目研究进展做一简单总结。

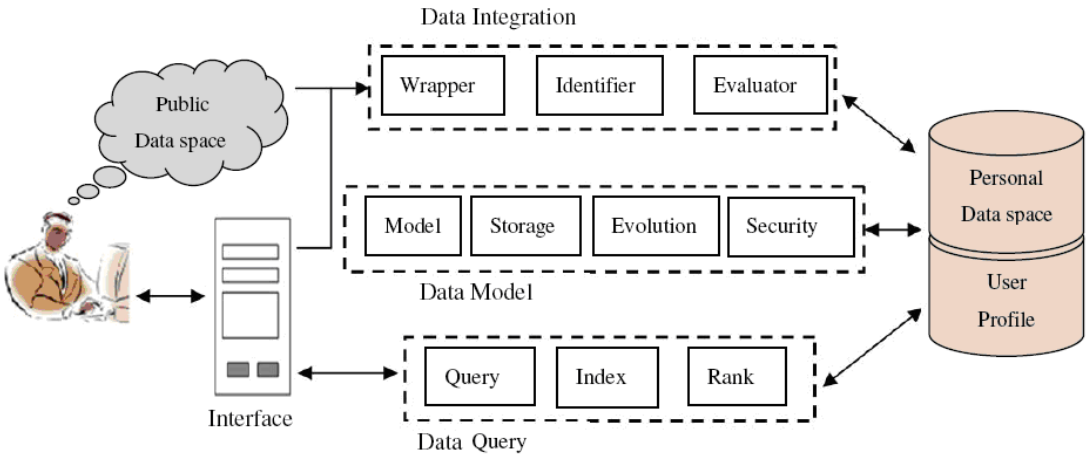



图 1 基于主体的数据空间管理系统框架

## 2 研究进展

由图 1 可以看出，数据空间模型、数据集成和数据查询是数据空间研究中的三个基本问题。在已有研究成果基础上，进一步针对这三方面的问题进行了研究，取得了一些研究成果。

### 2.1 数据空间模型

去年我们初步提出了任务空间模型和核心数据空间的思想。2009 年在新的研究成果基础上，对这两个概念进行了明确的阐述和形式化，使之更加完整。

**任务空间模型**

在维基百科中，任务定义为旨在完成特定目标的一系列行为的集合。在信息检索领域，人们很早就对于任务的概念、分类及特征进行了研究。随着信息技术的发展，计算机日益成为一个帮助人们解决问题、完成任务的重要工具，从而为任务管理带来新的特点。由此我们希望在相关工作基础上，从数据管理的角度对任务给出一个形式化的定义。

我们将任务定义为：用户具有明确目标的一系列数据操作的集合。静态的看，任务是一个与特定目标相关的数据集合；动态的看，任务是针对这一数据集合的操作序列。在完成任务目标的过程中，用户一方面会创造出新的数据对象，同时也会参阅许多已存在的数据文件。例如，当用户写一个项目报告时，任务目标可以看作最终的项目报告文件，在书写过程中，可能需要引用其他文档（网页、邮件等）上的一些内容（数字、图片、表格，等）。基于这一观察，我们将任务目标物化为用户生成的数据对象，并将与任务相关的数据集合分为两个子集：目标数据集和参考数据集。目标数据集是指用户在完成任务过程中生成的目标文件集合；参考数据集是指用户生成目标数据过程中参考过的数据文件。

基于目标数据集包含数据对象的数量，我们将任务区分为基本任务和复合任务。基本任务是指目标数据集中仅包含一个数据对象的任务。例如，用户写一封邮件可以看作是一个基本任务。复合任务是指涉及多个目标对象的任务。当用户写一篇论文的时候，往往会生成多个数据文件，这就构成一个复合任务。一个复合任务可以分解为多个基本任务。此外，时间是任务的一个重要属性，一个任务往往有一个生命周期。以上概念构成了任务概念模型。

在任务定义的基础上，我们进一步提出了任务空间的概念。任务空间定义为一个图，节点表示任务，边表示任务之间的关系，理论上可以定义多种任务关系，定义任务关系的目的是为了提高数据管理的效率。研究表明，内容和时间是人们基于任务查询个人数据时经常参考的两个因素。基于此提出并定义了两种任务关系：内容关系和时间关系。内容关系表示两个任务内容的相似性，可以支持基于内容的任务查询，如“查询与数据空间相关的所有任务”；时间关系表示两个任务的并行性，可以支持基于时间的任务查询，如“查询去年所完成的一些任务”、“查询准备 SIGMOD2010 论文期间所做生成的一些文件”等。任务空间为我们提供了一种用户任务描述数据的方法，基于此模型，用户可以执行基于任务的数据操作。

**核心数据空间**

2008 年我们提出了根据数据对象和用户的关联程度，构建核心数据空间的思想。核心数据空间由用户曾将访问过的数据对象组成，数据量往往还是比较大，用户很难通过直接浏览快速找到需要的数据对象。研究发现，用户记忆呈现出一定的规律性，这些记忆规律可以用来帮助用户提高访问效率。在信息检索、人机交互、认知行为学等领域，有许多关于用户认知行为规律的研究成果。通过将这些成果应用于数据管理，并结合数据空间查询的特点，我们将核心数据空间概念从以下几个方面进行了深化：

（1） 核心数据空间是一个用户曾经访问过的数据对象构成的数据集合。由于人们对个人数据信息的访问是一种“重访问“，因此我们提出只将用户曾经访问过的数据对

象作为核心数据空间的内容。

（2） 核心数据空间是基于用户记忆规律的多维视图。基于诸葛海研究员提出的资源空间模型，我们提出利用多维空间描述个人数据对象集合。其中每个数据轴对应一个数据属性，每个数据轴的坐标对应数据对象在该数据轴的取值。

（3） 数据轴上坐标的确定基于人的记忆规律。例如，研究表明，“随着时间推移，人们对于数据特征的记忆不断减弱”，基于此我们将最近访问时间属性区分为{今天，本周，本月，本年，一年以前}，等等。

基于以上核心数据空间的定义，我们提出了一个核心数据空间本体。其考虑了数据对象的 10 个共有属性作为数据轴，并给出了各个数据轴上坐标值的确定方法。

任务空间和核心数据空间是我们针对数据空间特点和用户查询需求，提出的两种框架模型，为进一步的研究奠定了基础。

## 2.2 数据空间演化集成

演化集成是数据空间管理的基本问题。传统的数据库往往是先构建数据模式，然后输入数据。在数据空间中，完全让用户手工地输入、修改数据信息是不现实的，因此必须探索一种自动的数据空间集成与演化策略，使数据空间能够在尽可能少的用户干预的情况下保持数据的高质量，以满足用户查询要求。数据空间演化集成包括很多研究问题，例如，如何从众多数据源中抽取数据；如何进行数据模式的匹配；如何实现“从数据到模式”的演化集成；如何高效地建立初始数据空间；如何通过演化保持数据质量。

**Web 数据抽取**

数据多样性是数据空间的主要特征之一，如何从众多数据源中抽取信息是数据空间集成面临的首要问题。目前 Web 已经成为最大的数据源，如何从 Web 数据库中高效地集成数据成为重要的研究问题。针对这一问题，我们创造性的提出了一种基于视觉的 Web 数据自动抽取方法（ViDE）。该方法考虑了网页信息的位置特征、布局特征、外观特征和内容特征，克服了原有方法依赖 HTML 树结构的不足。实验证明了这一方法的有效性。

**数据空间自动构建**

针对如何高效地构建数据空间的问题，提出了一种自动构建个人数据空间的策略。该策略通过分析操作日志，挖掘用户的兴趣特征，基于该特征计算数据对象与主体的相关度，从而自动识别与主体相关的数据对象，建立初始的数据空间。问题的挑战性在于如何自动挖掘用户兴趣特征，如何形式化的表示用户兴趣特征，以及如何计算数据对象与主体兴趣特征的相关度。我们提出了基于用户行为日志的方法，并从文件类型、文件夹、关键词等几个方面考虑了用户兴趣特征。从而对于任给的一个数据文件，通过计算该文件与用户兴趣特征的相似度，就可以计算出该文件属于数据空间的概率值。

**数据空间演化**

数据空间本质上可以看作一个语义链网络，其中每个节点是一个数据对象，两个数据对象之间的边称之为语义链。数据演化就是指系统能够自动地更新这个语义链网络，使之真实准确的反应数据空间的状态。例如，当用户访问了新的数据对象的时候，数据空间能够自动将该数据对象加入到该语义链网络中，并自动与相关的数据对象建立语义关联。数据空间中的语义链可以根据用户的需要建立很多种，例如，具有参考关系的文件之间可以建立语义链：曾经合作过的两个人可以建立语义链；同属于一个任务的两个数据对象可以建立语义链，等等。我们提出了基于用户行为的数据空间演化策略：基于用户操作，及时补充新的数据对象和语义关联，及时更新现有的语义链信息，从而使数据空间保持高数据质量。例如，当用户访问一个新的数据对象的时候，该数据对象会自动增加到核心数据空间中，会自动与相关的数据对象建立关联。

## 2.3 数据空间查询

传统的数据库查询往往是通过特定的查询语言实现，例如关系数据库中的 SQL 语言、XML 数据查询语言等。而数据空间面对的往往是没有数据管理背景的普通用户，因此需要为用户提供简洁的查询接口。在不同的场景下，用户往往需要不同的查询方式。例如，当用户编辑一个文档的时候，可能需要查询“该文档所引用的文件有哪些？”；当用户写一个总结报告的时候，可能需要查询“我最近半年完成了哪些任务？”；当用户想要查询一个文件的时候，可能只记得一些模糊的信息（如最近半年访问过、类型是 JPG 或 VSD，被存放在 D 盘等）。因此需要针对这些特定的查询场景设计与之相应的查询策略。针对不同应用需求，我们提出了基于核心数据空间的导航式查询策略、基于任务的数据查询方法、以及基于上下文的数据查询方法。

**基于核心数据空间的导航式查询**

该方法采用导航式查询，支持用户根据记忆线索，逐步定位到所查找的数据对象。提出了一种基于用户行为规律的简易的查询逻辑，从而支持用户表达比较复杂的查询语义。实现了一个个人核心数据空间本体，并提出了基于该本体建立多维查询接口的方法。

**基于用户操作上下文查询个人数据空间**

用户书写一个文档的时候，往往需要调用其他文档的一些数据信息，如数字、图片、表格、参考文献等。当用户重新编辑该文档的时候，经常需要访问该文档曾经引用的一些文档。对于这一用户需求，目前并没有好的解决方法。基于此我们定义了一种数据关系：上下文关联关系（Context-based Relation）,并提出了一种自动识别这种数据关系的方法，进一步提出了基于这一关系查询个人数据空间的策略（C-Query）。

**基于子图匹配的数据空间搜索技术**

传统数据库中数据关联基于表，而数据空间的数据关联是元组一级的，因此要复杂的多。数据空间本质上可以看作一个复杂的语义网络，也可以抽象为一个复杂的图结构。研究证明，一个图中的子图匹配查询是 NP 完全问题。针对这一问题，我们提出了一个双层索引的思想，对大图上每个点的两种特征进行索引：点的身份信息和位置信息。在进行查询处理时，首先通过第一层索引（点的身份信息）找到符合要求的点，然后再通过第二层索引（点的位置信息）进一步排除位置上不满足要求的点。实验证明我们提出方法是有效的。

## 2.4 数据空间管理原型系统

基于在数据空间模型、集成与查询方面的研究成果，我们对原有的两个数据空间原型系统进行了升级：

在 OrientSpace 系统中，增加了以下功能：

（1） 对于核心数据空间查询的支持

（2） 基于任务的数据查询功能

（3） 基于用户日志自动构建初始数据空间

（4） 基于 RDF 的数据存储系统

（5） 个人数据文件的自动标识

（6） 数据之间关联的挖掘等。

在 C-DBLP 系统中，增加了如下功能：

（1）同名区分功能；

（2）文献 BibTex 信息展示功能；

（3）作者的相关图片的自动收集与展示；

（4）合作关系的可视化展示，利用直观友好的图形化界面为用户提供更丰富的信息；

（5）论文数量趋势变化的图形化显示，帮助用户更直观地了解作者论文发表情况。

## 3 结论

在 2008 年研究成果基础上，进一步开展了深入细致的研究工作，特别加强了在数据空间模型、演化集成、查询优化方面的研究以及在原型系统研发方面的工作，取得了一些研究成果，并将取得的研究成果应用于已经开发的数据空间管理原型系统，验证了我们所提出的方法的可用性。这些成果为我们下一步的工作打下了坚实的基础。

# 云数据存储与管理

**柴云鹏**

IDC 的统计表明，2007 年一年的数据增量就达到了 281EB，最近几年的数据增长率在 60%左右，相当于每过 18 个月翻一番。在数据量如此迅速增长的背景下，传统的存储系统和数据管理系统在扩展性、效率和成本等方面遇到了巨大的挑战，无法满足需求。而云数据管理具有扩展性强、性价比高、容错性好等优势，可以承担起超大规模数据存储的重任。

我们目前在云数据管理系统方面与 IBM、NSN 等公司进行合作，研究和开发适合企业级应用的云数据管理系统。具体来说，在云数据库方面，我们基于 Map-Reduce 模型，并行实现 SQL 操作，并通过多维索引结构来优化系统的性能；在云存储系统方面，我们在云存储中引入了闪存缓存层，一方面能够提高系统的访问性能，另一方面通过合理的数据分布策略来减少整个系统的能耗。

# Cloud Data Management (CDM)

Yunpeng Chai

## Outline

- **Motivation of CDM**
- Survey of CDM
- IBM SUR Cloud DB Project
- NSN Cloud Project
- Future Research Work

# Motivation of CDM

- Rapid Data Increase
  - 2007: 281EB (1EB = 1000PB)
  - IDC: 60% per year (double - 18month)
  - Facebook: 850 million photos
    & 8 million videos ONE day
  - Web pages, logs, media contents



# Motivation of CDM

- Challenges
  - Scalability
    - Load Balance
    - Congestion / Delay
  - Efficiency
  - Cost
    - Fault Tolerance
    - Energy Conservation

# Motivation of CDM

- Advantages of CDM
  - Scalability
  - Fault Tolerance
  - Performance/Efficiency
  - Performance-cost
  - Living and evolving



# Motivation of CDM

- Comparison
  - Traditional Storage and DB
    - RAID/NAS/SAN
    - Hundreds of nodes
  - Cloud Data Management
    - Commodity Computers + SATA Disk
    - Tens of thousands of storage nodes

# Motivation of CDM

- Cloud Data Management Systems
  - Scalability
    - 100 → 10,000 nodes
  - Efficiency
    - Application-driven
  - Cost
    - Commodity Computers + SATA Disks
    - Flash / Disk Energy Conservation



# Motivation of CDM

- CDM Industry
  - Internet Service Cloud Platform
    - Google/Amazon
  - Cloud Products
    - IBM – Blue Cloud
    - MS - Azure
  - Enterprise-class Private Cloud
    - China Mobile
    - National Health Care

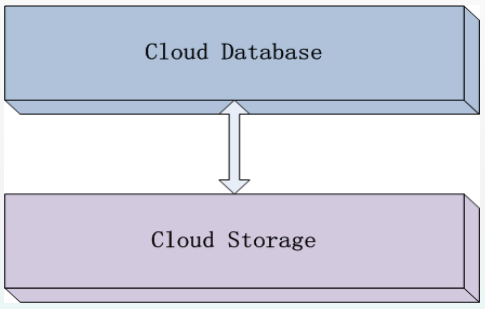
# Outline

- Motivation of CDM
- **Survey of CDM**
- IBM SUR Cloud DB Project
- NSN Cloud Project
- Future Research Work



# Survey of CDM

- CDM System Survey
  - Commercial Systems
    - Google- GFS + BigTable
    - MS- Azure
    - Yahoo!- PNUTS
    - Amazon- Dynamo
  - Open Source Systems
    - HDFS
    - KFS
    - Hbase
    - HyperTable
    - Cassandra
    - CouchDB
    - Voldemort



# Survey of CDM

- Cloud Database:
  - Key-value data model
  - R/W performance / Parallelism
  - No/ Simple SQL operations



# Survey of CDM

- Cloud Storage:
- Architecture:
  - Master-Slave
    - GFS, HDFS
    - Baidu Cloud
  - P2P
    - Cassandra

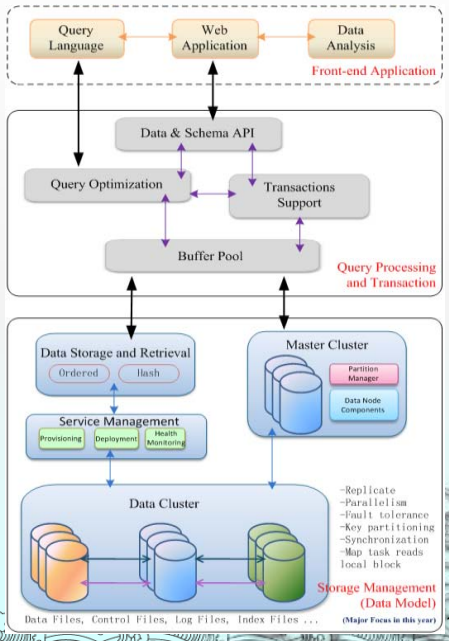
# Outline

- Motivation of CDM
- Survey of CDM
- **IBM SUR Cloud DB Project**
- NSN Cloud Project
- Future Research Work



# IBM SUR Cloud DB Project

- Cloud Data Management Systems
  - Based on Cassandra (Facebook)



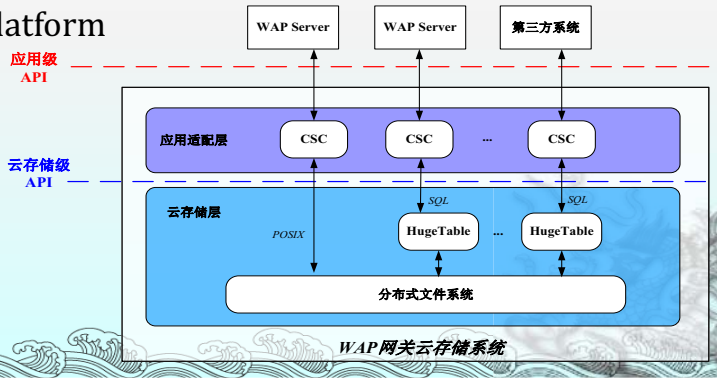
# Outline

- Motivation of CDM
- Survey of CDM
- IBM SUR Cloud DB Project
- **NSN Cloud Project**
- Future Research Work



# NSN Cloud Project

- Background
  - Telecom Cloud
    - Scalability
    - Cut costs
    - Unified platform



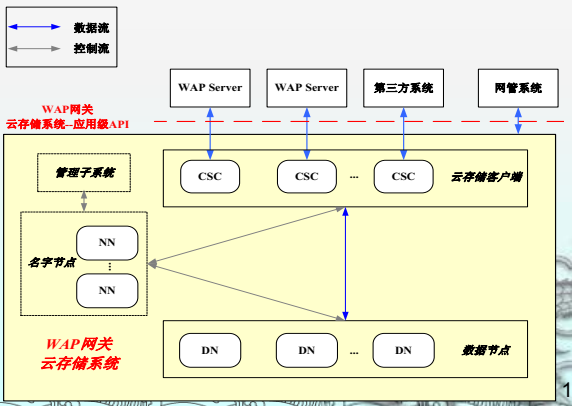
# NSN Cloud Project

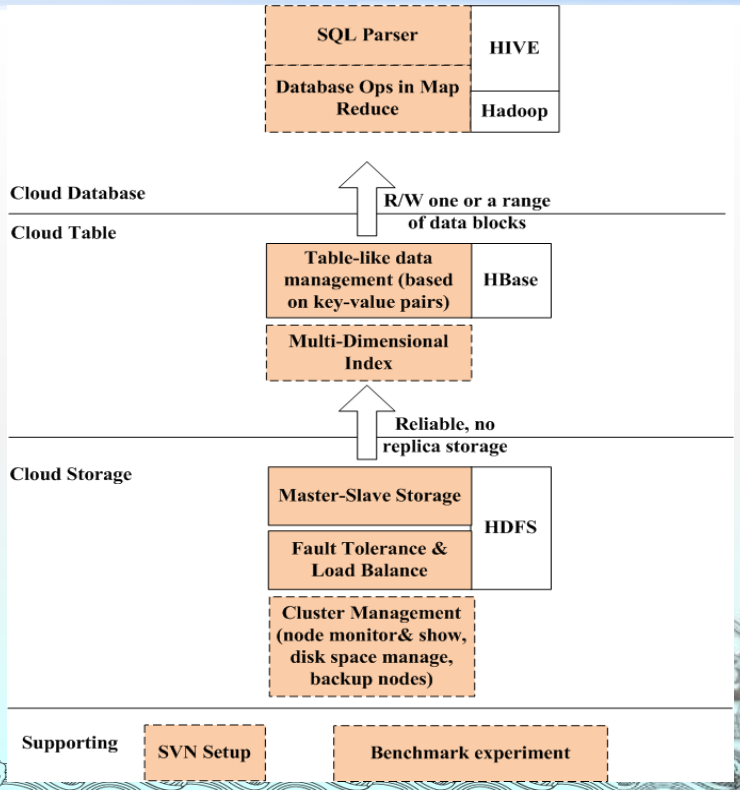
- Deliverables
  - Package of source code for CDM
    - CMRI WAP specification as input
    - Based on open source cloud project
    - Suit telecom app/workload
  - Demo of CDM*
    - Integrated with NSN WAP Server
  - Relevant investigation report and technical documentation



# NSN Cloud Project

- Solution
  - Cloud Storage
    - Based on HDFS (Java, M-S arch)
    - Data dist strategy
  - Cloud Database
    - Hbase (Java)
    - Data type
    - DB Func
    - SQL support
    - Performance opt








# Outline

- Motivation of CDM
- Survey of CDM
- IBM SUR Cloud DB Project
- NSN Cloud Project
- **Future Research Work**

# Future Research Work

- Cloud Database
  - Data model
  - DB operation extension
  - Efficiency
    - Multi steps
    - Distributed operations



# Future Research Work

- Cloud Storage
  - Architecture: Master-Slave vs P2P
  - Flash
  - Energy Conservation

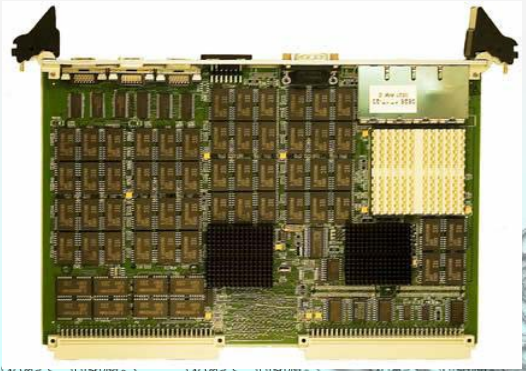
# Future Research Work

- Architecture: Master-Slave vs P2P
  - P2P
    - Easy to manage, scalable
    - Fault tolerant
  - Master-Slave
    - High efficiency



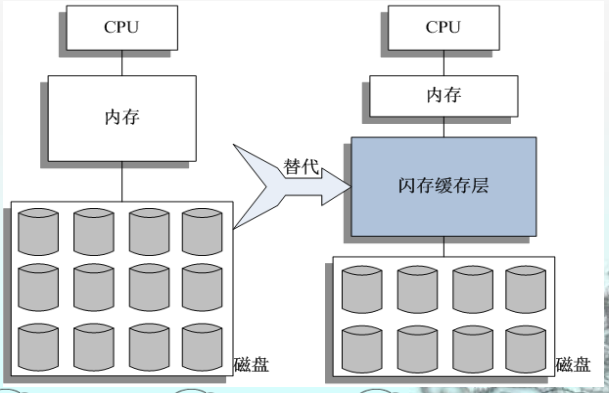
# Future Research Work

- Mixed Storage (Flash + Disk)
  - Flash Database
    - Commercial SSD based optimization
    - Flash storage board

# Future Research Work

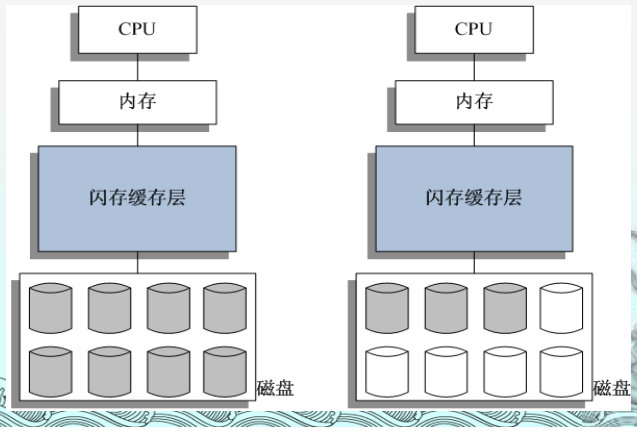
- Mixed Storage (Flash + Disk)
  - Flash's Drawbacks:
    - Costs
    - Erase




25 /26

# Future Research Work

- Energy Conservation
  - More Energy-Efficient SSD
  - Dynamic data distribution




26 /26

# 系统评测报告

# 云数据存储管理系统评测报告

**史英杰 王仲远 王海平 刘兵兵**

云计算是当今信息产业最受关注的一种计算模式，在这种模式下，企业和个人可以根据自己的需要购买存储设备和计算能力，而不是花费大量资金购买大规模高性能计算机。作为云计算的一项关键技术，云数据存储和云数据管理为业界带来巨大的潜在商用价值。随着信息产业的发展，企业和公司产生的数据量快速增长，通常数据规模可以达到 TB 甚至 PB 级别。如何管理和分析海量数据是目前很多领域所面临的问题，例如在医疗、通信和互联网领域。云环境是由大量的性能普通、价格便宜的计算节点组成的一种无共享大规模并行处理环境，所以从成本和性能两方面考虑，越来越多的企业更愿意把自己的数据中心从昂贵的高性能计算机转移到共有或私有云环境中。在互联网时代，海量数据的存储和处理操作非常频繁，很多研究者都在从事这方面的研究，也涌现出很多云数据管理系统。下面的 ppt 就介绍了部分当前的云数据管理系统，并对它们的结构、数据模型及数据一致性进行了比较分析。

# Survey on Data Management in the Cloud

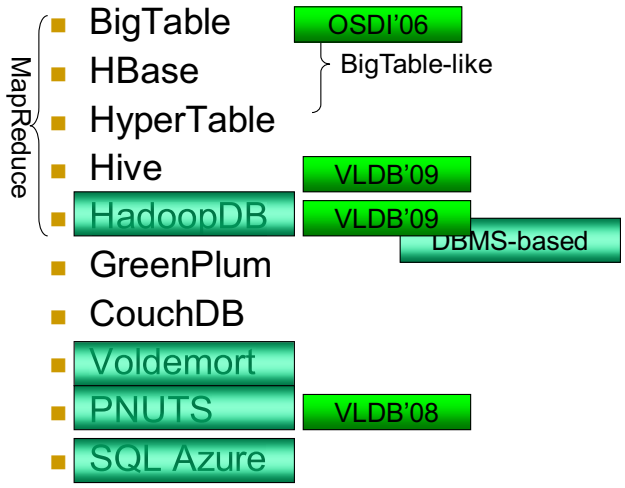
Yingjie Shi



## Outline

- Systems surveyed
- Comparison of Systems
- Experiment Benchmark

# Systems Surveyed






# BigTable – Basic Information

- To manage structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers
- Motivations
  - Scale is too large for most commercial databases
  - Even if it weren't, cost would be very high
  - Low-level storage optimizations help performance significantly

Google

# BigTable – Goals

- Fault-tolerant, persistent
- Scalable
  - 1000s of servers
  - Millions of reads/writes, efficient scans
- Self-managing
- Simple!



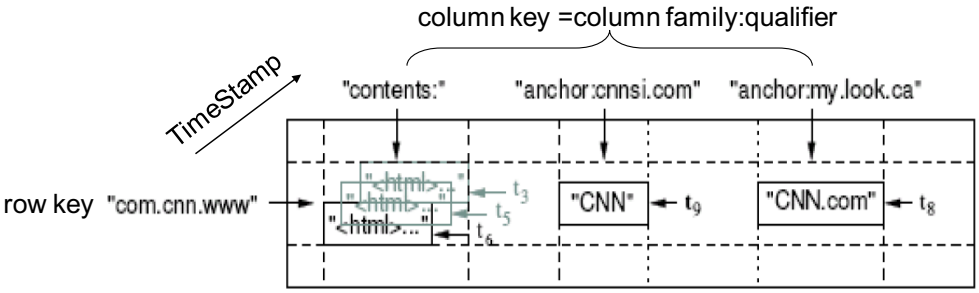
# BigTable – Applications

- Based on: GFS(Google File System)
- Applications: Google 地球 Google finance Google maps Google Analytics
- Scale of servers:

| No. of tablet servers | No. of clusters |
|---|---|
| 0 … 19 | 259 |
| 20 … 49 | 47 |
| 50 … 99 | 20 |
| 100 … 499 | 50 |
| >500 | 12 |

# BigTable – Data Model

- It is a sparse, distributed, persistent multidimensional sorted map.




The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.



# BigTable – Storage

- Column family – oriented storage(key->value)
  - (row:string, column:string, time:int64) ->string

row=row0, column=anchor:cnnsi.com, timestamp=1174184619081 → XXXXXXXXX
row=row0, column=anchor:my.look.ca, timestamp=1174184620720 → XXXXXXXXX
row=row0, column=anchor:my.look.ca, timestamp=1174184617161 → XXXXXXXXX
row=row1, column=anchor:cnnsi.com, timestamp=1174184619081 → XXXXXXXXX
row=row1, column=anchor:my.look.ca, timestamp=1174184620721 → XXXXXXXXX
row=row1, column=anchor:my.look.ca, timestamp=1174184617167 → XXXXXXXXX
row=row2, column=anchor:cnnsi.com, timestamp=1174184619081 → XXXXXXXXX
row=row2, column=anchor:my.look.ca, timestamp=1174184620724 → XXXXXXXXX
row=row2, column=anchor:my.look.ca, timestamp=1174184617167 → XXXXXXXXX
row=row3, column=anchor:cnnsi.com, timestamp=1174184619081 → XXXXXXXXX
row=row3, column=anchor:my.look.ca, timestamp=1174184620724 → XXXXXXXXX
row=row3, column=anchor:my.look.ca, timestamp=1174184617168 → XXXXXXXXX
row=row4, column=anchor:cnnsi.com, timestamp=1174184619081 → XXXXXXXXX
row=row4, column=anchor:my.look.ca, timestamp=1174184620724 → XXXXXXXXX
row=row4, column=anchor:my.look.ca, timestamp=1174184617168 → XXXXXXXXX
row=row5, column=anchor:cnnsi.com, timestamp=1174184619082 → XXXXXXXXX
row=row5, column=anchor:my.look.ca, timestamp=1174184620725 → XXXXXXXXX
row=row5, column=anchor:my.look.ca, timestamp=1174184617168 → XXXXXXXXX
row=row6, column=anchor:cnnsi.com, timestamp=1174184619082 → XXXXXXXXX
row=row6, column=anchor:my.look.ca, timestamp=1174184620725 → XXXXXXXXX
row=row6, column=anchor:my.look.ca, timestamp=1174184617168 → XXXXXXXXX

## HBase

- A clone project of BigTable using Java
- Developers: Apache Software Foundation
- Runs on top of Hadoop core
- Production users: Powerset streamy openplaces WorldLingo Adobe YAHOO! HBASE

9 /31

## Hypertable

- A clone project of BigTable in C++
- Developers: Zvents Baidu 百度 rediff.com
- Runs on top of CloudStore(KFS,Kosmos File System)

10 /31

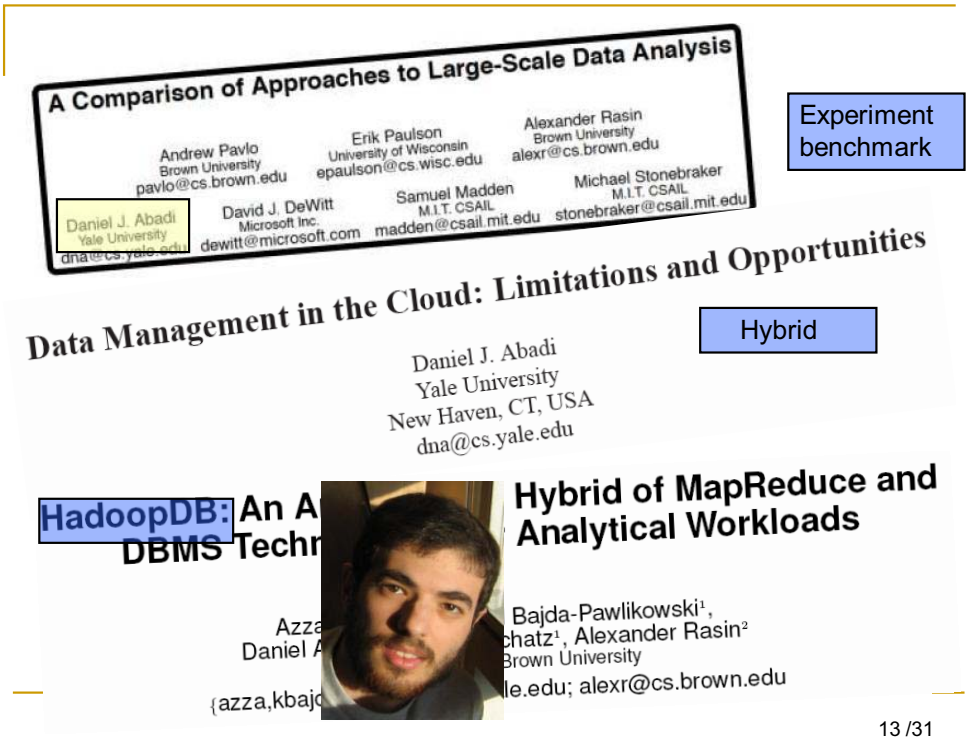
## BigTable-like VS RDBMS

- Fast Query Rate
  - No Joins, No SQL support, column-oriented database
  - Uses one Bigtable instead of having many normalized tables
- Is not even in 1NF in a traditional view
- Support historial queries



## Hive - Basic Information

- A system for managing and querying structured data built on top of Hadoop
  - Map-Reduce for execution
  - HDFS for storage
  - Metadata on raw files
- Key Building Principles:
  - SQL as a familiar data warehousing tool
  - Extensibility - Types, Functions, Formats, Scripts
  - Scalability and Performance






# HadoopDB-Philosophy

- Two largest components of the data management market
  - Transactional data management
  - Analytical datamanagement

Moved to cloud

- Two technologies used for data analysis in a shared-nothing MPP architecture
  - Parallel database
  - MapReduce-based system

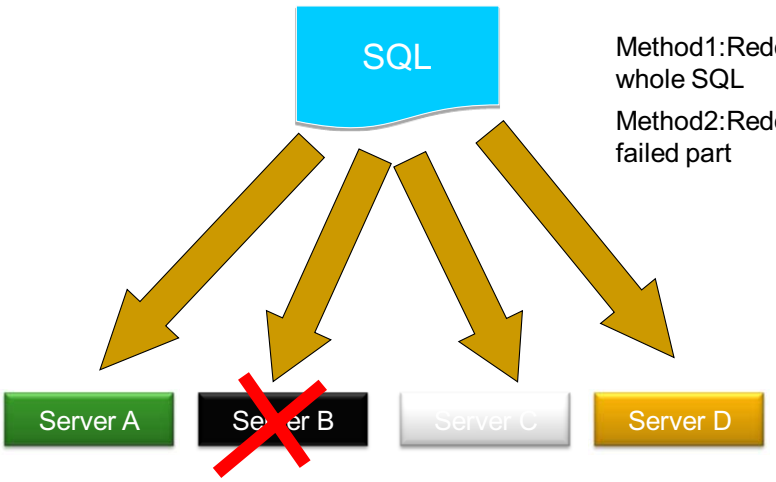
# HadoopDB-Philosophy

| | Scalability | Tolerance | High Performance |
|---|---|---|---|
| Parallel database | ✗ | ✗ | ✓ |
| MapReduce | ✓ | ✓ | ✗ |
| What we want | ✓ | ✓ | ✓ |

Scalability:1000 nodes
High Performance:Queries on structured data



# Query Tolerance




Method1:Redo the whole SQL

Method2:Redo the failed part

# HadoopDB-Philosophy

- Goals
  - Performance
  - Tolerance
  - Scalability
  - Flexible query interface
- Design idea
  - Multiple, independent, single-node databases coordinated by Hadoop

Translation layer--Hive

Communication layer--Hadoop

Database layer--PostgreSQL



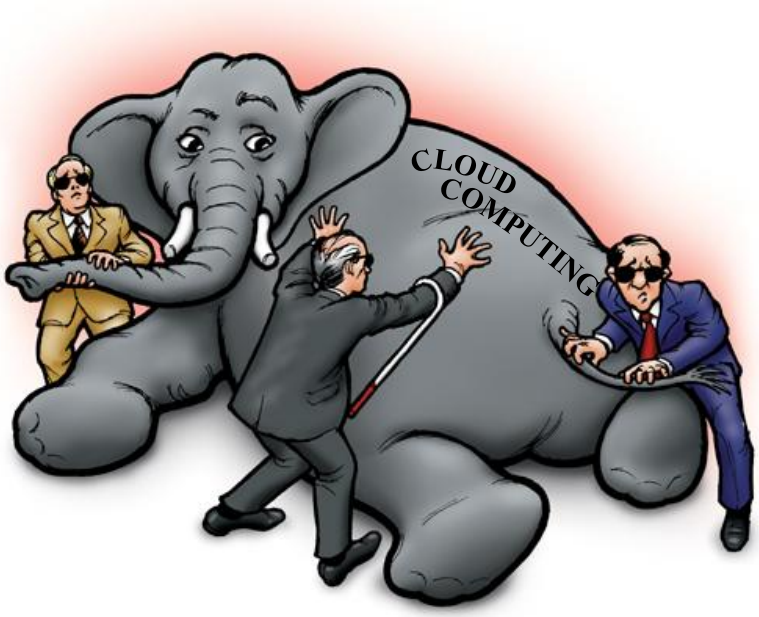
# PNUTS

- Developer: YAHOO!
- Applications: Social network, advertising application
- Application characteristic:
  - Scalability
  - Geographic scope
  - Fast response requirement
  - High availability
  - Simplified query needs
  - Relaxed consistency needs

# SQL Azure

- A relational database service on the Windows Azure Platform that is built on SQL Server technologies
- Objects can be created on SQL Azure:
  - Tables
  - Indexes
  - Views
  - Stored Procedures
  - Triggers





# Outline

- Systems surveyed
- Comparison of Systems
- Experiment Benchmark



# Characteristic of Cloud Database

- Performance
- Scalability
  - Ability to scale by adding resources with minimal operational effort and minimal impact on system performance
  - Performance increases with the scale of the system extends
- High Availability and Fault Tolerance
- Ability to run in a heterogeneous environment
- All applications are read-only or read-mostly

# Summary of Applications

- Data Analysis
  - Internet Service
  - Private Cloud
- Web Applications
  - Some operations that can tolerate relaxed consistency

BigTable HBase HyperTable Hive HadoopDB…

PNUTS



# Architecture

| MapReduce-based | DBMS-based | Hybrid of MapReduce and DBMS |
|---|---|---|
| BigTable HBase Hypertable Hive | SQL Azure PNUTS Voldemort | HadoopDB |
| scalability<br>fault tolerance<br>ability to run in a heterogeneous environment<br>data replication in file system | easy to support SQL<br>easy to utilize index, optimization method | sounds good |
| a lot of work to do to support SQL | bottleneck of data storage<br>data replication upon DBMS | Performance? |

## Data Model

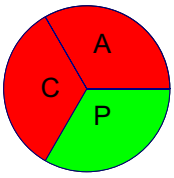
- Big Map Model
  - BigTable,HBase,Hypertable
- Simple Relational Data Model
  - Hive, PNUTS, SQL Azure and HadoopDB

It depends on the real application!

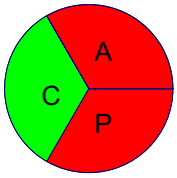


## Consistency

- Two kinds of consistency:
  - strong consistency – ACID(Atomicity Consistency Isolation Durability)
  - weak consistency – BASE(Basically Available Soft-state Eventual consistency )

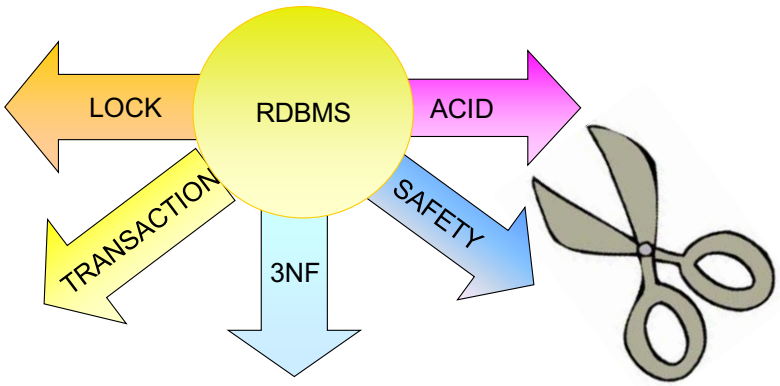



BigTable,HBase,
Hive,Hypertable,HadoopDB




PNUTS

# A tailor






# Outline

- Systems surveyed
- Comparison of Systems
- Experiment Benchmark

# Experiment Benchmark

MapReduce
DBMS

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo, Brown University, pavlo@cs.brown.edu
Erik Paulson, University of Wisconsin, epaulson@cs.wisc.edu
Alexander Rasin, Brown University, alexr@cs.brown.edu
Daniel J. Abadi, Yale University, dna@cs.yale.edu
David J. DeWitt, Microsoft Inc., dewitt@microsoft.com
Samuel Madden, M.I.T. CSAIL, madden@csail.mit.edu
Michael Stonebraker, M.I.T. CSAIL, stonebraker@csail.mit.edu

**A Benchmark for Hive, PIG and Hadoop**[1]

Yuntao Jia, Zheng Shao

July 12th, 2009

HadoopDB
Hadoop
DBMS

**6. BENCHMARKS**

In this section we evaluate HadoopDB, comparing it with a MapReduce implementation and two parallel database implementations, using a benchmark first presented in [23][4]. This benchmark consists of five tasks. The first task is taken directly from the original MapReduce paper [8] whose authors claim is representative of common MR tasks. The next four tasks are analytical queries designed to be representative of traditional structured data analysis workloads that HadoopDB targets.



# Experiment Benchmark

- Tasks:
  - Data Load
  - Grep Task
  - Selection Task
  - Join Task
  - Aggregation Task
- Data
  - Grep
  - UserVisits → Structured data
  - Rankings
  - Documents → Unstructured data

# References


- Daniel J. Abadi. Data Management in the Cloud: Limitations and Opportunities. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering
- Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff and Raghotham Murthy. Hive-A Warehousing Solution Over a MapReduce Framework. VLDB 2009
- Azza Abouzeid, Kamil BajdaPawlikowski, Daniel Abadi, Avi Silberschatz, Alexander Rasin. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. VLDB 2009
- Armando Fox, Eric A. Brewer. Harvest, Yield, and Scalable Tolerant Systems. Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems 1999
- J. Hamilton. Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services. In Proc. of CIDR, 2009.
- J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI, 2004.
- Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein,Philip Bohannon, HansArno Jacobsen, Nick Puz, Daniel Weaver and Ramana Yerneni. PNUTS: Yahoo!’s Hosted Data Serving Platform. VLDB2008.

# 固态硬盘 I/O 特性测试

**周大**

众所周知，固态硬盘是一种由闪存作为存储介质的数据库存储设备。由于闪存和磁盘之间物理特性的巨大差异，现有的各种软件系统无法直接使用闪存芯片。为了提供对现有软件系统的支持，往往在闪存之上添加一个闪存转换层来实现此目的。固态硬盘就是在闪存上附加了闪存转换层从而提供和磁盘相同的访问接口的存储设备。一方面，闪存本身具有独特的访问特性。另外一方面，闪存转换层内置大量的算法来实现闪存和磁盘访问接口之间的转换。因此和磁盘相比，固态硬盘具有复杂的访问特性。本测试集中在展示各种固态硬盘在不同访问模式下的 IOPS 值，同时和磁盘加以比较。本测试中，采取的访问模式有：随机读、连续读、随机写，连续写，以及分别加入地址跳跃、时间延迟、批量提交、延迟+批量提交、读写混合等影响因数。通过在各种访问模式下的实验结果可知，固态硬盘普遍具有较好的读性能和连续写性能，在随机写表现出复杂的特性。

# Understanding IO patterns of SSDs

Da Zhou

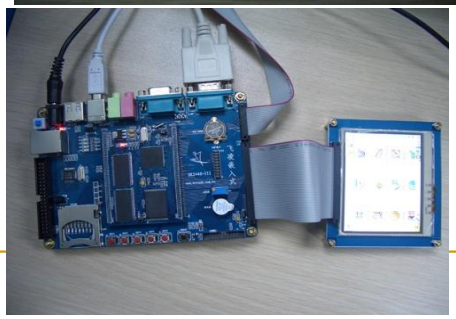


## Outline

- Random/Sequential read/Write
- Address offset
- Delay
- Relay
- Burst
- Delay + Burst
- Semi Access
- Conclusion

# Experiment Setup

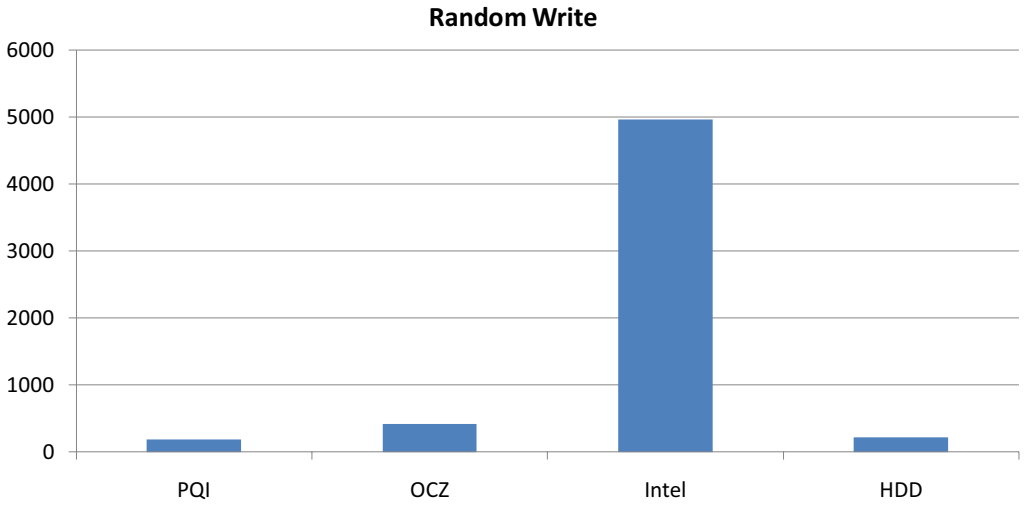




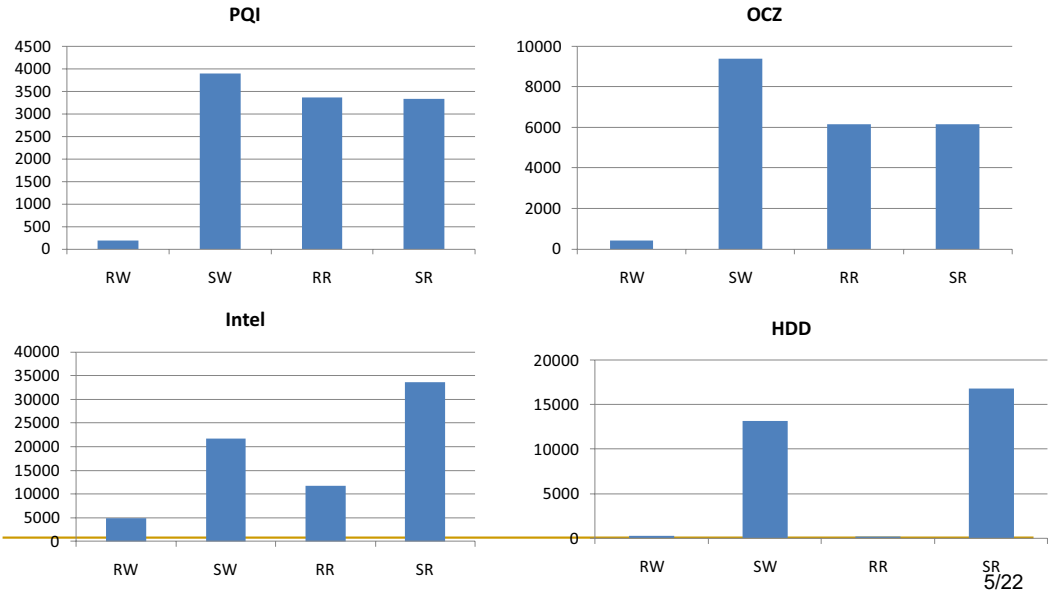
SSD:
Access granularity: 512 bytes
Test tool: IOMeter



# Random Write

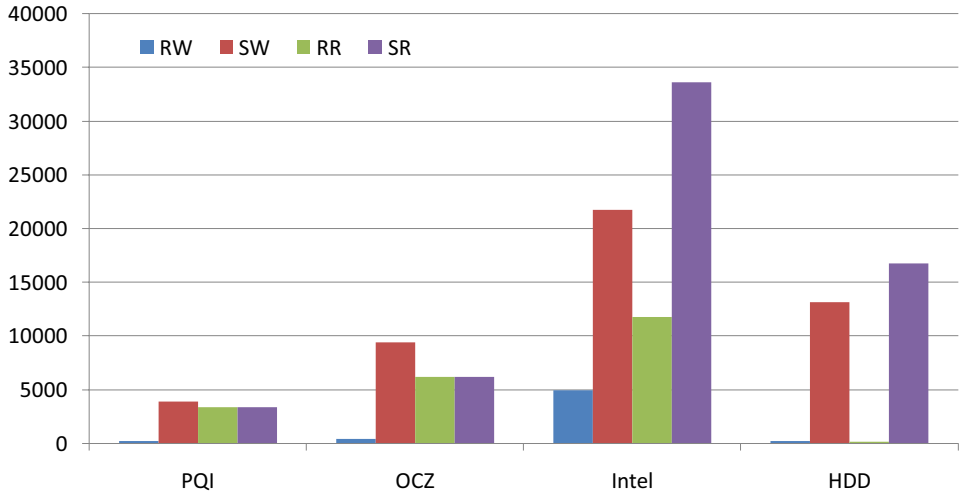


# Random Write




5/22

# Random Write



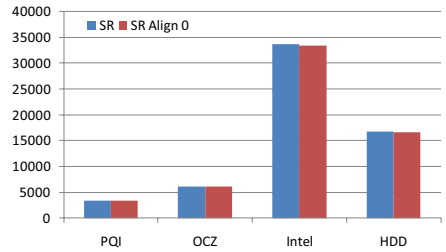

6/22

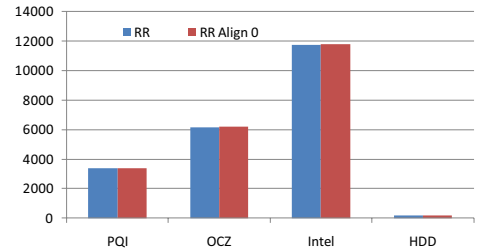
# Hints

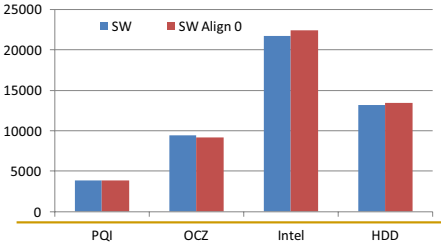
- Random write performance is low for most of SSDs
- New high-end SSD has high random write performance
- Random write is slower than sequential write.

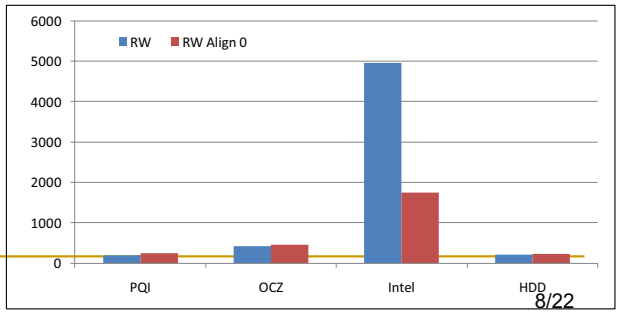
7/22

# Sequential Write VS Align Write



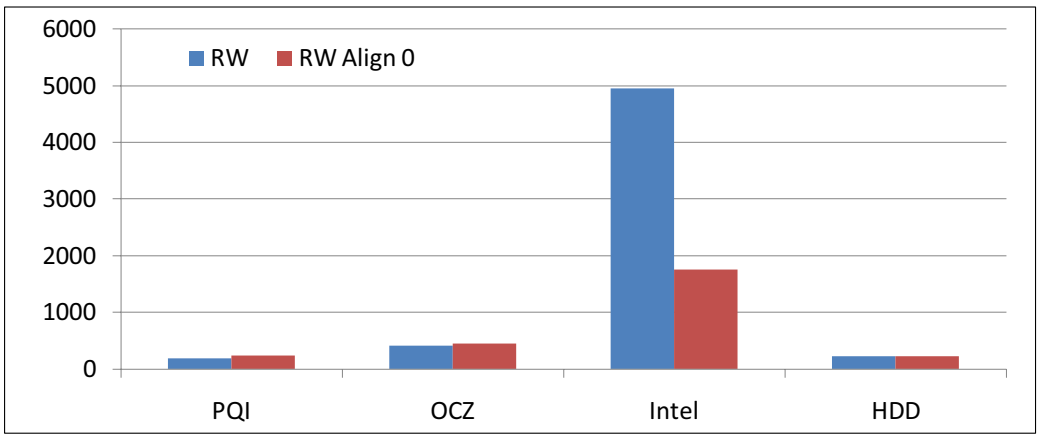




8/22

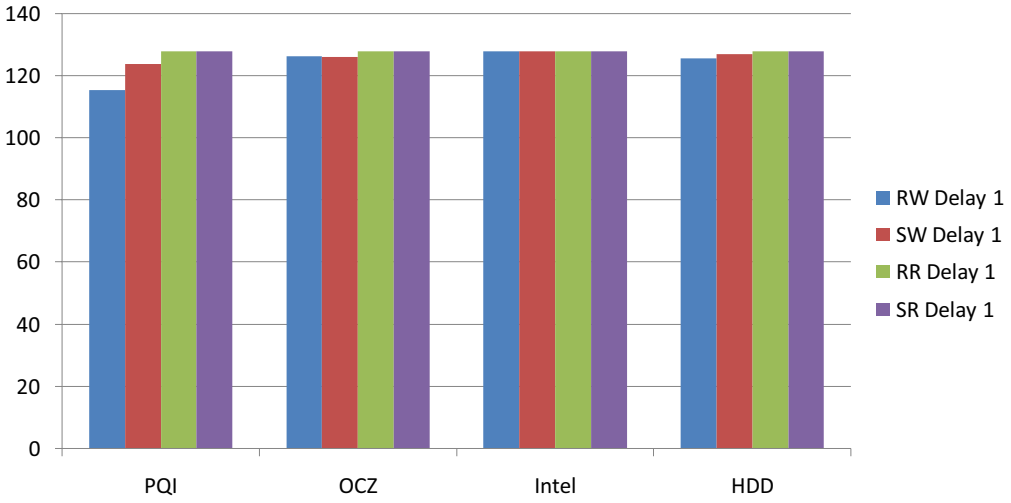
# Random write VS Random write Align




PQI, OCZ: directly written into flash memory chip
Intel: group written because of cache. Align leads to flush directly.

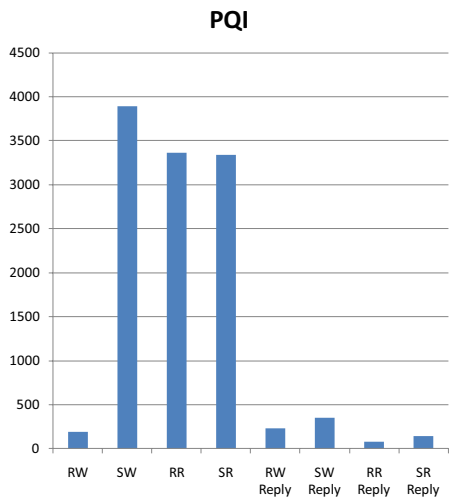
9/22

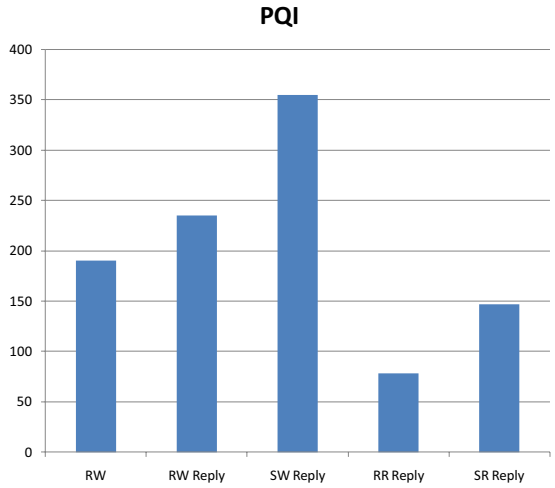
# Delay




10/22

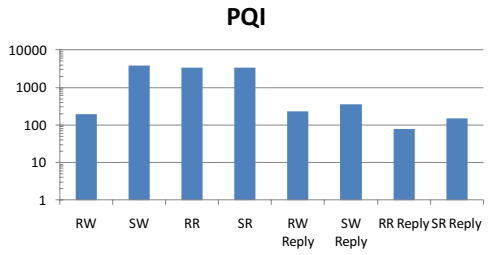
# Replay

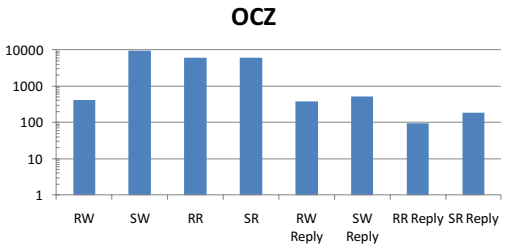


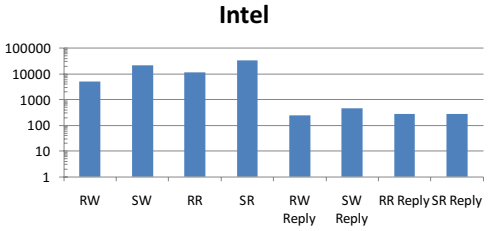



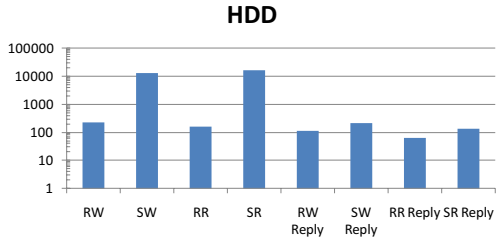

11/22

# Write-Read



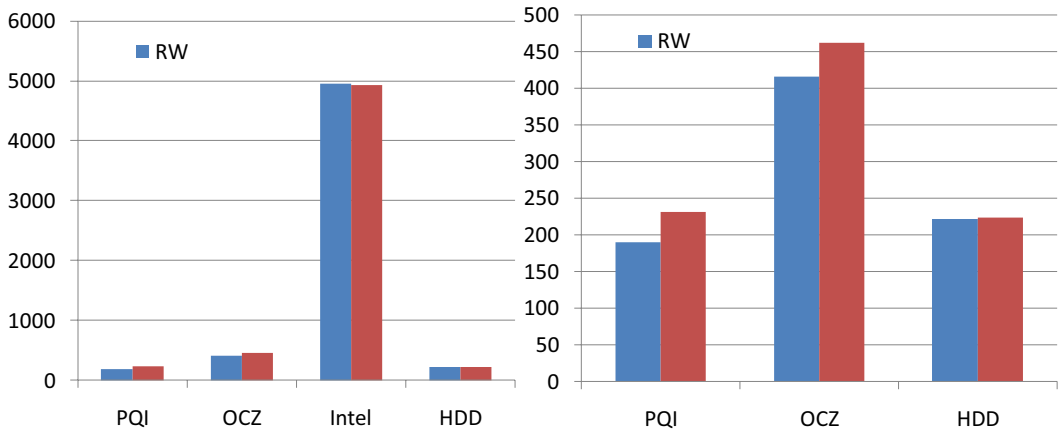




12/22

# Burst 5

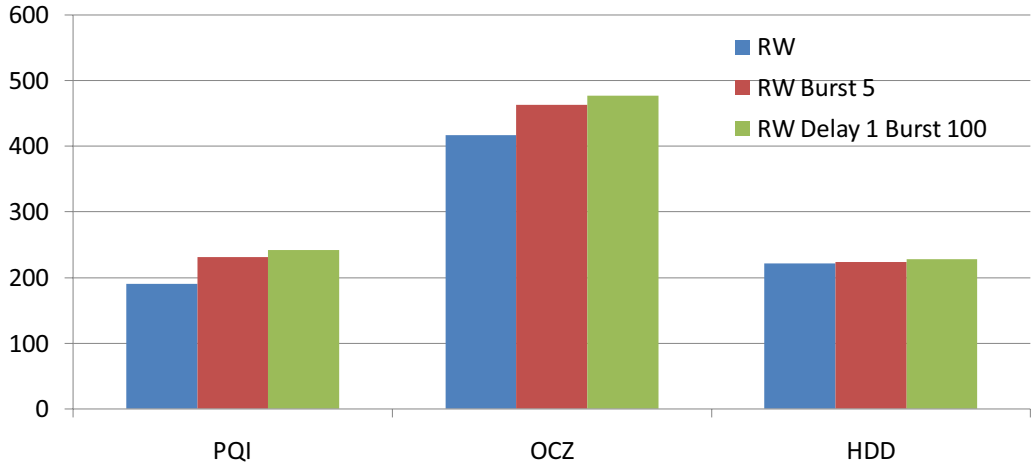



PQI, OCZ : Data are written into flash memory directly. While burst will improve the efficiency of Cache.

Intel, HDD: The capacity of cache is larger. More data are cached in RAM.

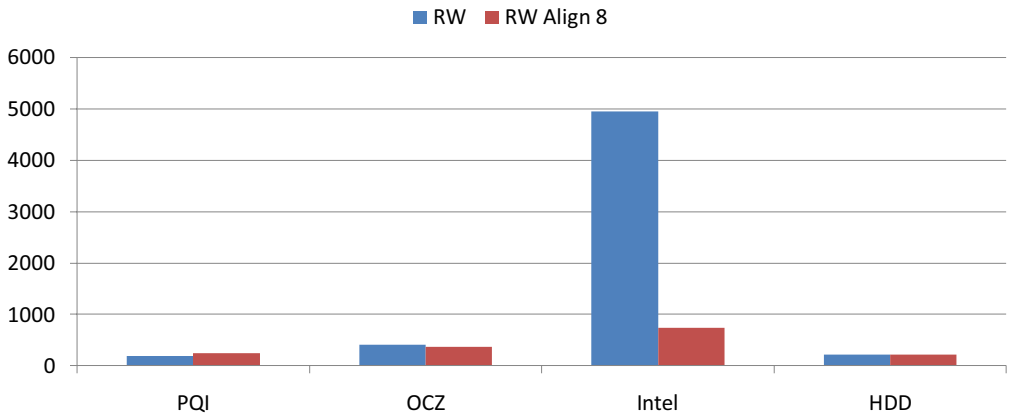
13/22

# Burst and Delay




14/22

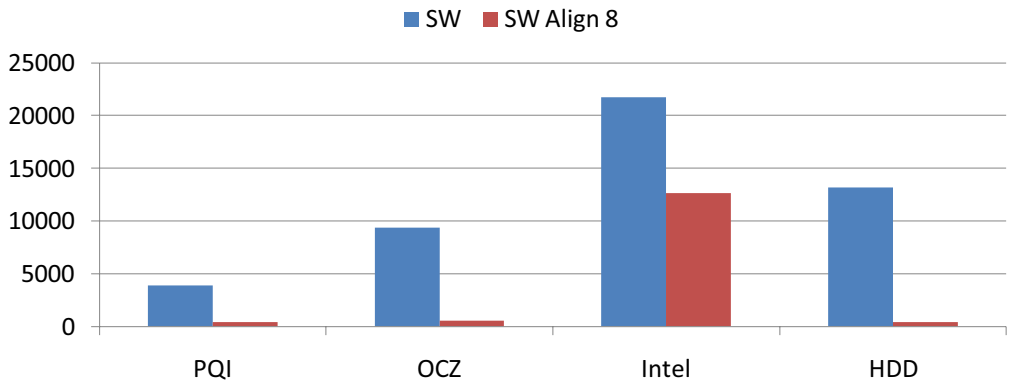
# Random write alignment




Intel: Data in the same 4kb area are flushed in the same time. When data do not locate in the same 4kb area, they will be flushed independently.
PQI, OCZ: data are flushed into SSD directly.

15/22

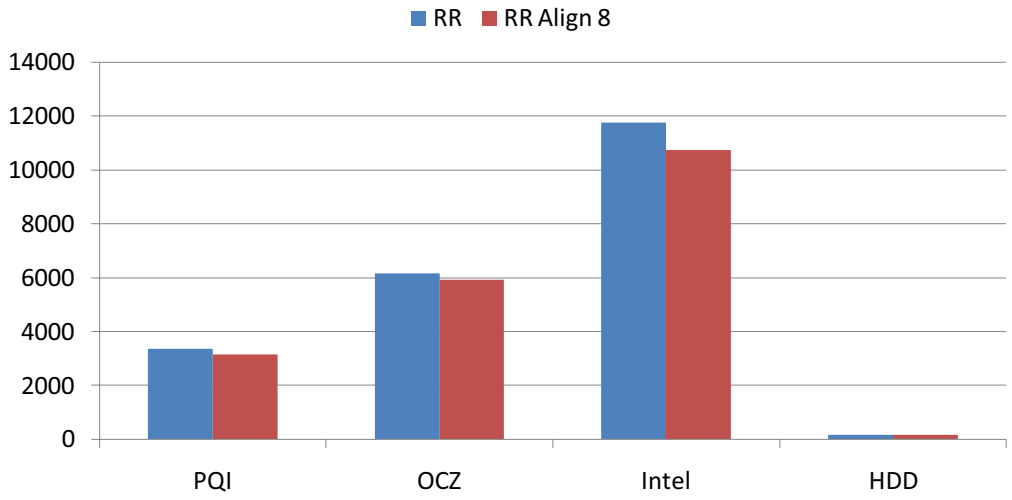
# Sequential write alignment




SSD: More erase operations will be triggered.
HDD: Sequential write ➔ random write

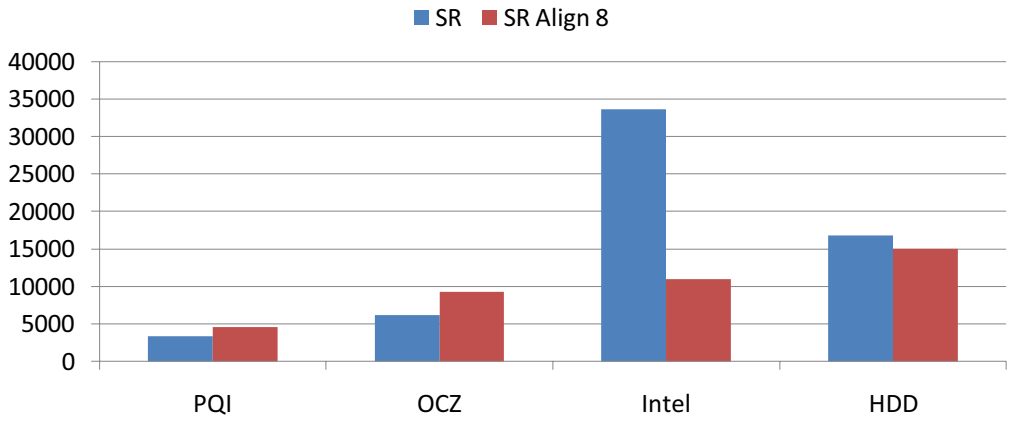
16/22

# Random read alignment




17/22

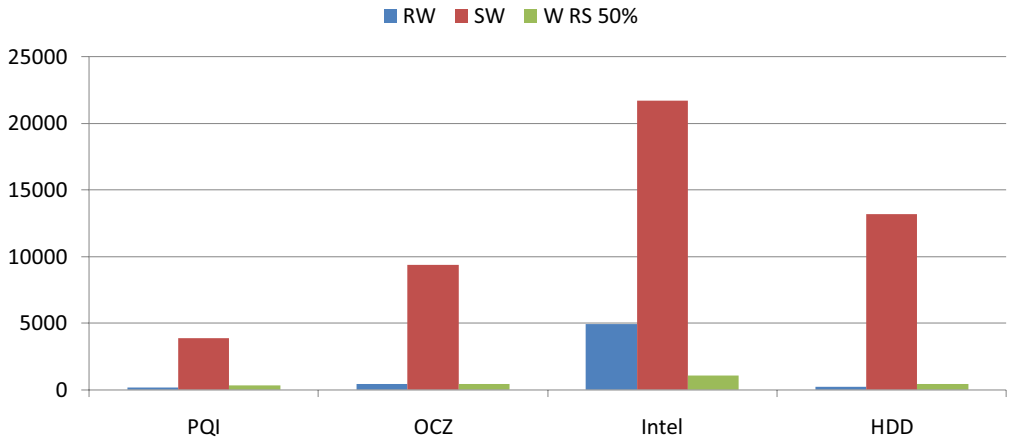
# Sequential read alignment




PQI, OCZ: ???
Intel: Prefetch is not utilized. Suppose the size is 4 KB, the remain data is needed.

18/22

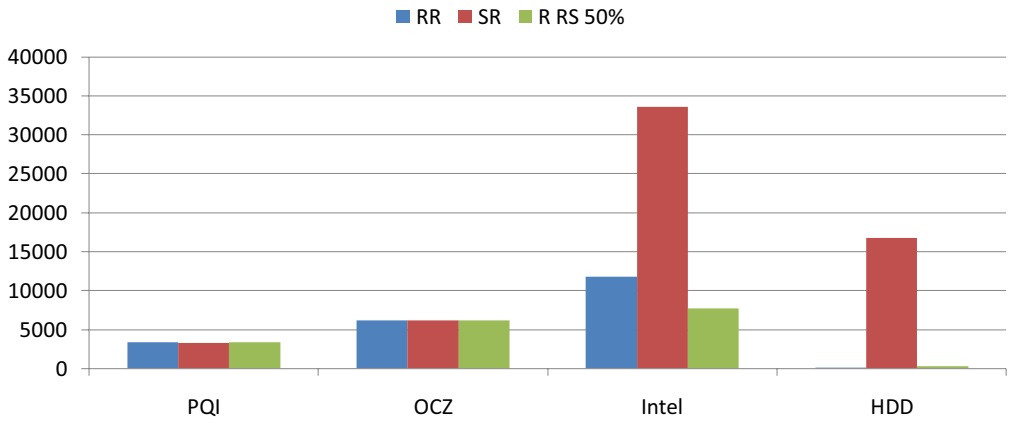
# Random Write VS Sequential Write VS Semi-Random write




***Intel***: RW, SW: Cache; Read RS 50%: Data which is cached by RW is flushed by SW quickly. The reason maybe is the sequential write need a lot of cache. Or sequential write has higher priority than random write to use cache.

19/22

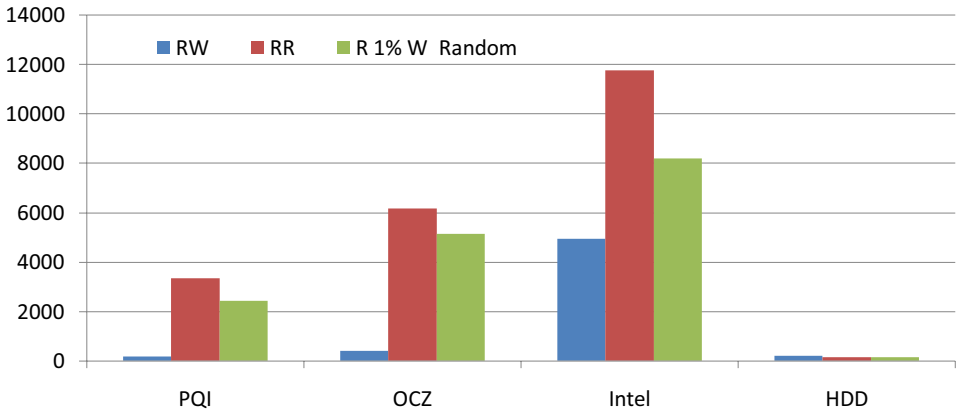
# Random Read VS Sequential Read VS Semi-random Read




***Intel***: RR, SR: Prefetch; Read RS 50%: Data which is prefetched by RR is evicted by SR quickly. SR need more cache or high priority to use cache.

20/22

# 99% random Read, 1% random write




**Write affects read**

| PQI | |
|---|---|
| RW | 190.2775 |
| RR | 3360.399 |
| R 1% W Random | 2441.968 |

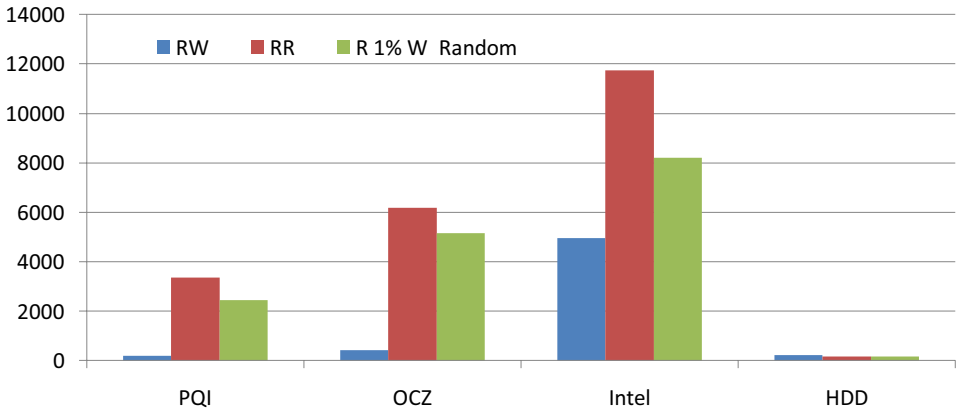
1 Write = 3360/190 reads= 18 reads

Ideal Read: 100/118= 84%

Real Read: 2441/3360= 72%



# 99% random Read, 1% random write




**Problem is more obvious**

| Intel | |
|---|---|
| RW | 4954.68 |
| RR | 11753.65 |
| R 1% W Random | 8200.149 |

1 Write = 11754/4954 reads= 2.4 reads

Ideal Read: 100/112.4= 97%

Real Read: 8200/11754= 69%

# 论文精选

In Proceedings of the CIKM Workshop on Cloud Data Management (CloudDB2009),
November 2, 2009, Hong Kong, China

# An Efficient Multi-Dimensional Index for Cloud Data Management

Xiangyu Zhang, Jing Ai, Zhongyuan Wang, Jiaheng Lu, Xiaofeng Meng
School of Information, Renmin University of China
Beijing, China, 100872
zhangxy@live.com {aijingruc,zhywang,xfmeng}@ruc.edu.cn jiahenglu@gmail.com

## ABSTRACT

Recently, the cloud computing platform is getting more and more attentions as a new trend of data management. Currently there are several cloud computing products that can provide various services. However, currently the cloud platforms only support simple keyword-based queries and can't answer complex queries efficiently due to lack of efficient index techniques. In this paper we propose an efficient approach to build multi-dimensional index for Cloud computing system. We use the combination of R-tree and KD-tree to organize data records and offer fast query processing and efficient index maintenance. Our approach can process typical multi-dimensional queries including point queries and range queries efficiently. Besides, frequent change of data on big amount of machines makes the index maintenance a challenging problem, and to cope with this problem we proposed a cost estimation-based index update strategy that can effectively update the index structure. Our experiments show that our indexing techniques improve query efficiency by an order of magnitude compared with alternative approaches, and scale well with the size of the data. Our approach is quite general and independent from the underlying infrastructure and can be easily carried over for implementation on various Cloud computing platforms.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; H.2.4 [**Database Management**]: Systems—*concurrency, transaction processing*

## General Terms

Algorithms

## Keywords

multi-dimensional index, distributed index, query processing

## 1. INTRODUCTION

Internet has been developing at an astonishing speed. Each day a huge amounts of information is put on the Internet in the form of digital data. Many new Internet applications emerge and most of them require to process a large scale of data efficiently. However, traditional data management tools have been insufficient for this new demands. For example, database systems softwares often are multi-tenancy, which means that online users must share the same software's resources simultaneously. When unexpected spikes come, users may meet the situation of shortage of resources and a drop of quality of service. Therefore, scalability is a crucial requirement for future Web applications. Under those circumstances, a new computing infrastructure, *cloud computing*, emerges. Though the unified definition of cloud computing has not been confirmed[1], it is considered as a revolution in IT industry. Systems supporting cloud computing dynamically allocate computational resources according to users' requests. Existing Cloud computing systems include Amazon's Elastic Computing Cloud(EC2)[2], IBM's Blue Cloud[3] and Google's MapReduce[4]. They adopt flexible resources management mechanism and provide good scalability. Scalable data structures can satisfy resource demands of Cloud systems' users. Cloud computing systems are usually comprised of a large number of computers, store huge amounts of data, and provide services for millions of users. Resources allocation is typically elastic in cloud systems, which makes each user feel that he owns infinite amount of resources. A typical example of scalable data structure is Google's BigTable[5].

Currently, most of Cloud infrastructures are based on Distributed File Systems. DFS usually use key-value storage models to store data. The data in Cloud systems are organized in the form of key-value pairs. Therefore, current Cloud systems can only support keyword search. When a query comes, result data are retrieved from DFS in accordance with contained keywords. Although many famous Cloud systems uses this information storage pattern, such as Google's GFS[6] and Hadoop's HDFS[7], they only provide services of keyword queries for users. Therefore, users can only access information through "point query" which matches records to satisfy the verbal and/or numerical values.

The emergence of cloud computing is due to the need of increasing advanced data management. And it needs to serve a large variety of applications better for more Web users. Therefore, future cloud infrastructures should be developed to support more types of queries with more functions, e.g. muti-dimensional queries.

Cloud computing platforms contain hundreds and thousands of machine nodes, and they process workloads and tasks in parallel. This is a typical characteristic of cloud computing infrastructures. When a user submits a query, result data are retrieved from underlying storage tables and then distributed to a set of processing nodes for parallel scanning. Without the support of efficient index structure, query processing is quite time-consuming, especially for complex queries. Therefore, building more efficient index structure


*CloudDB'09,* November 2, 2009, Hong Kong, China.

is a pressing demand. Moreover, because of huge amounts of data in cloud systems, the index should be able to provide high retrieval rate.

Up to now, there are some proposals of building efficient index for cloud infrastructures. Aguilera et al.[8] proposed a scalable distributed B-tree for their Cloud system. Other research work proposed a kind of index based on hash index structure. However, these indices just can index single column. They can not efficiently support range queries referring to multi-columns' data.

In order to support range queries efficiently in the Cloud system, we present a scalable and flexible multi-dimensional index structure based on the combination of R-Tree and KD-tree.

In summary, this paper makes the following contributions:

- We propose an efficient and scalable multi-dimensional index structure. With this structure we can answer typical point queries and range queries efficiently. Our index scales very well as the data volume or cluster size grows.
- We propose a cost estimation-based index update strategy. With this strategy we can assure that update will only be done when it's necessary and the benefit of update is ensured.
- We perform a series of experiments on large scale of machine nodes with large volume of data. The experiment confirms that our index structure is quite efficient and scalable.

## 2. RELATED WORK

Cloud computing brings new ways of Web services for Web users and enterprises. There have been some cloud computing systems. Typical examples include Amazon's Elastic Computing Cloud(EC2)[2], and IBM's Blue Cloud[3] and Google's MapReduce[4]. These systems designed for cloud computing usually only support basic key/value based queries, and lacks more efficient index structures.

The concept of cloud computing initially comes from search engines' infrastructure. Unlike DBMS, search engines usually does not adopt order-preserving tree indexes, such as B-tree or hash table. To improve performance and support more types of queries, some works tried to build index on cloud computing platforms. The work in [9] proposed an extension of MapReduce to join heterogeneous datasets and execute relational algebra operations. And searching tree indexes were built in bulk MapReduce operations. However, this work mainly focused on improving search engines' performances and might lack generality when used in cloud computing platforms.

B-tree is a very commonly used index in database management systems, and most prior work on B-tree usually focused on ones stored in a single computer's memory space. The work in [8] presented a more general and flexible index structure: a fault-tolerant and scalable distributed B-tree for cloud systems. Distributed transactions is used to make changes to B-tree nodes. B-tree nodes can be migrated online between servers for load-balancing. This design is based on a distributed data sharing service, Sinfonia[10], which provides fault tolerance and a light-weight distributed atomic primitive. However, this index schema may cause high memory overhead because of inner nodes' replication, especially for client computers. Moreover, although B-tree has been widely used as single-attribute index in database systems, it is inefficient in dealing with indices composed of multi-attributes.

The paper by S. Wu and K.-L. Wu[11] proposed an improved indexing framework for cloud systems. This indexing framework supports all existing index structures. The hash index and $B^+$-tree index are used to demonstrate the effectiveness of the framework.

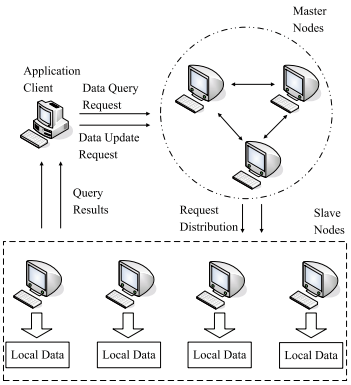



**Figure 1: Framework of Request Processing in Cloud**

And machine nodes are organized in a structured Peer-to-Peer network which can effectively reduce the index maintenance cost. Although this index schema is scalable and flexible, the Peer-to-Peer structure is not very suitable for cloud systems.

There are also some algorithms of distributed B-tree in distributed file systems and databases(e.g.,[12, 13]). However, these distributed B-tree indices can not support multi-dimensional query answering effectively. Because even if an attribute column in the data table is indexed by a distributed B-tree, answering multi-dimensional queries still need to find eligible result records on other attributes. And query answering is still possible to have long response time.

Much work on distributed index structures has been done by researchers, such as distributed hash tables(DHT) (e.g. [14, 15]). However, these indexes in [14, 15, 16] are designed and deployed on Peer-to-Peer data structures. Although some DHT extensions can support range queries[16], P2P structures work with little synchrony and may cause weak even no consistency. In contrast, cloud systems must be able to provide certain level of consistency. Moreover, nodes in P2P structures are equal with each other. Cloud system has master nodes which are responsible for distributing computing tasks and resources to slave nodes. Therefore, these distributed hash index can't meet the demand of cloud systems.

In contrast with that, our distributed index can efficiently support various queries(e.g. point query, range query), and provide high retrieve and update rates.

## 3. QUERYING AND UPDATE IN THE CLOUD

As a basic characteristic of the cloud platform, a cluster consisting of hundreds or thousands of PC is responsible for the mission of computing and storage of data. As Figure 1 shows, machine nodes in the cluster can be categorized into two types: master nodes and slave nodes. Master nodes and slave nodes are not too much different except that if a machine is playing the role of master node it will store some meta data about the whole system along with other regular data that slave nodes also have to store. Slave nodes store data records and their replicas for efficiency and security. Although one of the distinguishing characteristics of Cloud platform from the Client-Server architecture based systems is that the Cloud systems don't need central servers, it still needs a set of machines to maintain meta data about the whole system, and this makes many operations more efficient.

In the cloud platform, client requests are often posed against the master nodes. After that the master nodes decide which slave nodes are relevant to the request and then the client will communicate

with those nodes directly. The framework of request process in cloud platform is in Figure 1. So as a typical request, query processing in the cloud platform can be divided into two phases: locating relative nodes and processing the request on selected slave nodes. The procedure could be expressed as algorithm 1.

**Algorithm 1** Process query on cloud

```
1: procedure SET PROCESSQUERY(Query q)
2:     Set nodes = empty;
3:     nodes.add(getRelativeNodes(q));
4:     Set results = empty;
5:     for (each node n in the nodes) do
6:         results.add(n.retrieveRecords(q));
7:     end for
8:     return results;
9: end procedure
```

Maintenance of the index upon data insertion and deletion is also a major aspect of an index. Like the query processing procedure, insertion and deletion can also be divided into locating relative nodes and performing the operation on relative nodes. The two procedures can be described as algorithm 2 and 3:

**Algorithm 2** Record insertion to cloud

```
1: procedure BOOLEAN INSERTRECORD(Record r)
2:     Set nodeSet = empty;
3:     nodeSet.add(getNodesForRecord(r));
4:     for (each node n in nodeSet) do
5:         if (n.insertRecord(r) == false) then
6:             return false;
7:         end if
8:     end for
9:     return true;
10: end procedure
```

**Algorithm 3** Record deletion from cloud

```
1: procedure INT DELETERECORDS(Query q)
2:     Set nodeSet = empty;
3:     nodeSet.add(getRelativeNodes(q));
4:     int count = 0;
5:     for (each node n in the nodeSet) do
6:         count += n.deleteRecords(q);
7:     end for
8:     return count;
9: end procedure
```

From the above discussion we can see that the key functional components of a multi-dimensional index are:

**Query Processing**

- Locating relative slave nodes for query
- Processing query on each slave node and fetch results

**Index Maintenance**

- Locating appropriate slave nodes for record insertion
- Locating relative slave nodes for data deletion (same as that in query processing)
- Inserting records into individual slave node
- Deleting records from individual slave node

In the following part of the paper, we will discuss how to build and maintain multi-dimensional indices in cloud computing environment by conducting the 6 key functional components listed above.

# 4. MULTI-DIMENSIONAL INDEX

As the Cloud computing platform can be considered as a cluster of PC machines, we can build a global index of the platform by building local indices on each individual machine. Requests to the virtual global index could be answered by executing the query on local indices and then combining the returned results. Before introducing our index approach, we give a short introduction to the structures we will use.

## 4.1 R-Tree and KD-tree

R-tree[17] is a popular multi-dimensional index, which is usually used in spatial and multi-dimensional applications. R-tree index is a data structure that captures some of the spirit of the B-tree for multi-dimensional data. A R-tree index represents data that consists of 2-dimensional, or higher dimensional regions. An interior node of an R-tree corresponds to some interior region. In principle, the region can be of any shape.

A kd-tree[18] is a binary tree in which each interior node has an associated attribute a and a value V that splits the data points into two parts: those with a-value less than V and those with a-value equal to or greater than V. The attributes at different levels of the tree are different, with levels rotating among the attributes of all dimensions.

## 4.2 Basic Structure

For the basic structure, we build the multi-dimensional index for the platform by building local KD-tree index for each slave nodes. The reason for our choice of KD-tree instead of other structures is that the KD-tree can efficiently support point query, partial match query and range query.

**Query Processing**

In the relative node locating phase we choose all the nodes in the cluster as candidates of the query since currently we don't have the knowledge about data distribution on each slave node, and thus makes all nodes possible to contain records relative to the query. And in the record retrieving phase, each node utilizes the local KD-tree index to get records on that node. The procedures are describe as algorithm 4 and 5:

**Algorithm 4** Get candidate nodes to search for the query

```
1: procedure SET GETRELATIVENODES(Query q)
2:     return all the nodes of the platform;
3: end procedure
```

**Algorithm 5** Get records satisfying the query on the node

```
1: procedure SET RETRIVERECORDS(Query q)
2:     Set recordSet = lookupKDTree(q);
3:     return recordSet;
4: end procedure
```

**Index Maintenance**

For data insertion, since in the basic structure there is not any metadata to consider so we only take load balance into consideration. Hence, we pick a set of nodes based on some load balance

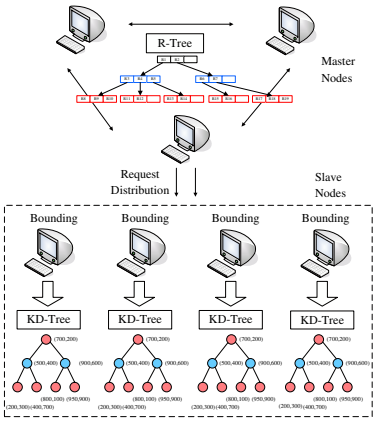



**Figure 2: Framework of EMINC**

approach, which is not the focus of this paper so we will not discuss it. After that, we apply the insert function defined on the local KD-tree. For data deletion, as each node is a potential node for query processing, we need to perform local deletion on every slave node.

## 4.3 Pruning Irrelevant Nodes with R-tree

The approach we have shown distributes local indices on slave nodes without maintaining any meta-data and this directly leads to inefficiency of query processing. The efficiency of locating relative nodes can be improved by maintaining bounding information of each dimension on each node and prune irrelevant nodes during query processing.

To prune irrelevant nodes on query processing, we construct a *node cube* for each slave node. A node cube indicates the range of value on each indexed attribute in this node.

*Definition 1.* A **node cube** is a sequence of value intervals, and each interval represents the value range of one indexed attribute on this node. If there's only one value on some dimension, the corresponding interval regresses to a value point.

**Example 1 :**If we construct a two-dimension index on attribute *age* and *salary* of a table, we can make a node cube of {[30, 40], [100, 200]} meaning that records on this node have *age* attribute between 30 and 40 and *salary* attribute between 100 and 200.

After we build a cube for each slave node, we maintain the cubes on master nodes with an R-tree. The reason why we choose R-tree instead of KD-tree for cube information is that the R-tree was designed for managing data regions and in our scenario the cubes are actually multi-dimensional data regions. We call this index approach *EMINC*: Efficient Multi-dimensional Index with Node Cube. Framework of EMINC is shown in Figure 2

*Definition 2.* **EMINC** index structure consists of a R-tree in master nodes and one KD-tree on each slave node. Each leaf node of the R-tree contains a node cube and one or more pointers that point to the slave nodes corresponding to the node cube.

With the node cube information in EMINC, query processing can be greatly improved by pruning irrelative nodes in the nodes locating phase. And in order to keep cube information available and useful, insertion and deletion on slave nodes that may change their cubes should inform master nodes for update of cube.

**Query Processing**

When a query is posed, we first get the key value or key range on each demand, construct a *query cube* for query. Query cube is an analogy concept to node cube with the following definition:

*Definition 3.* A **query cube** is a sequence of intervals, and each interval represents the value range of one attribute in this query. If either side of the attribute is not specified, we assign it the biggest negative or positive value accordingly.

With the query cube we resort to the R-tree for nodes that are related to the query by issuing the classic "where am I" query. Specifically, we look up the R-tree to find those slave nodes whose node cubes intersect with the query cube of the query. The definition of cube intersection is as follows:

*Definition 4.* **Intersection** of two cubes (node cube or query cube) means that for each attribute the two corresponding intervals must have overlap. If one of the two intervals has regressed to a point, then the intersection semantic is replaced by the interval containing the point or the two points being equal, depending whether there is one point or two.

We can see from the definition of intersection that this can be done with the typical "where am I" query on R-tree. And only the intersecting nodes are possible to contain records satisfying the query, so that a big part of irrelevant nodes are pruned. After locating relative nodes, we do local search on the slave node. The new nodes locating procedure is as algorithm 6:

**Algorithm 6** Get candidate nodes using cube

```
1: procedure SET GETRELATIVENODES(Query q)
2:     QueryCube cube = getCubeForQuery(Query q);
3:     Set nodeSet = getNodesForCube(cube);
4:     return nodeSet;
5: end procedure
```

**Index Maintenance**

In order for the node cube information to stay effective, we have to update the cube on master nodes if the cube is out-of-date due to data insertion or deletion on slave nodes. If the cube information on master nodes is not updated in time, the query processing will either miss relative records or search more irrelative nodes.

Recall that the first step of data insertion is to select the appropriate slave nodes to insert to. Selection of nodes can affect future query processing efficiency. If we can select the nodes in a way that data records are "clustered" by their indexed attribute values, then future query processing can benefit greatly from this since less slave nodes need to be explored for one query. Based on this idea, we try to give such nodes higher priority in data insertion : nodes that have node cube that can cover the record to be inserted, and by cover we mean that each indexed value of the record is in the corresponding interval of the node cube. Under this principle, the node selection procedure is shown as algorithm 7.

After selecting the nodes to insert to, record is inserted into them. Insertion of a new data record may cause the node cube of this node to expand on one or more dimensions if the current cube can't enclose the new record. And if this happens, this node must inform the master node of the change and give the new node cube to master node to keep it up to date. The update on master node is a typical update operation on R-tree.

**Example 2 :** Suppose the current node cube is {[30, 40], [100, 200]}. If we insert a new record (42, 210), and then the new cube

**Algorithm 7** Select nodes for insertion

```
 1: procedure SET SELECTNODES(Record r)
 2:     candidateSet = empty;
 3:     int count = 0;
 4:     for (count < replica amount) do
 5:         Node node = selectNodeWithCoveringCube(r);
 6:         if (node is not null) then
 7:             candidateSet.add(node);
 8:         else
 9:             // This means there is no more such nodes.
10:             break from loop;
11:         end if
12:     end for
13:     if (count < replica amount) then
14:         // Choose rest of the nodes.
15:         int remaining = replica amount - count;
16:         // Choose nodes based on load balance, etc.
17:         Set remainingSet = chooseTheRest(remaining);
18:         candidateSet.add(remainingSet);
19:     end if
        return candidateSet;
20: end procedure
```

will be {[30, 42], [100, 210]}. If the cube information is not updated in time, a query looking for records with the first attribute between 41 and 50 will ignore this node.

The new insertion procedure goes as algoritm 8.

**Algorithm 8** Insert record to slave node

```
 1: procedure BOOLEAN INSERTRECORD(Record r)
 2:     Boolean b = insertToKDTree(r);
 3:     if (b == false) then
 4:         return false;
 5:     end if
 6:     if (current cube is empty) then
 7:         Make cube based on this record;
 8:     end if
 9:     if (current cube has expanded or a new cube is created)
    then
10:         Update cube on master nodes;
11:     end if
12:     return true;
13: end procedure
```

Likewise, deletion of a data record on slave nodes will possibly cause certain intervals to shrink if the deleted record lies on one of the vertices of the cube and there is no record on that vertex after the deletion. If this happens, the node cube will also be updated. If the new cube is not update in time, further queries will still think this node to contain some data this node is not holding any more. Therefore, the deletion procedure goes like algorithm 9:

**Algorithm 9** delete records on slave node

```
1: procedure INT DELETERECORDS(Query q)
2:     Int count = deleteFromKDTree(q);
3:     if (current cube has shrunk) then
4:         Update cube on master nodes;
5:     end if
6:     return count;
7: end procedure
```

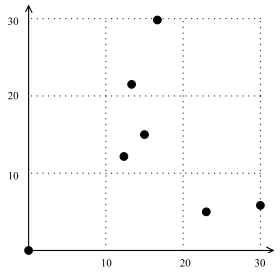



**Figure 3: Cutting Node Cube**

## 4.4 Extended Node Bounding

With EMINC, we use bounding technique to filter unnecessary queries. However, EMINC has some limitations and could be further extended to provide much better performance.

In EMINC, we make one cube for each node to describe the smallest and biggest key value on this node. But under some occasions, the performance could still be poor.

**Example 3 :** Suppose we have two nodes now: data on node A have key value on attribute X from 1 to 100, each integer included; data on node B have only three values, 1, 50 and 100. By the previous approach, node cubes of the two nodes will both be [1,100] on dimension(attribute) X. And now we have a query asking for records with attribute X between 60 and 80. In the current situation both of the nodes will be selected as candidate since their cubes both intersect with the query cube. But we can easily see that search on node B will end up getting nothing since node B doesn't hold any record between 60 and 80.

We can see from the extreme case stated in above example that if we use one node cube to describe a node, the cube may be so sparse that it will lead to a great number of waste of searching since sparse distribution on nodes will cause many unrelated queries. To deal with this problem, we propose to extend EMINC to use multiple node cubes to represent a slave node more precisely, and by doing this we will be able to filter out much more irrelevant queries.

In order to achieve higher accuracy, we need to cut the original node cube into several denser smaller cubes, and then adjust the smaller cubes by checking data records within each cube. We name this approach *EEMINC*: Extended EMINC.

*Definition 5.* **EEMINC** is an extension of EMINC. The difference from EMINC is that in EEMINC data records on one slave node will be represented by multiple node cubes. The shape and amount of node cubes is dependant on the method used for cutting the original single node cube.

Here we give an example on turning the node cube of EMINC into cubes of EEMINC. Discussion on different methods of cutting attributes will be presented later in the section.

**Example 4 :** Suppose on some node A, we have 7 data records: [0, 0], [12, 12], [15, 15], [13, 21], [17, 30],[23, 5], [30, 6]. See the distribution of data in the coordinate in Figure 3. The node cube of this node is {[0, 30], [0, 30]}. Now we cut both axis X and Y to three equal pieces and get nine small regions. From the distribution of records we can see that only four of the nine regions have records in them, and we only keep those four regions. Now we make four node cubes by checking the actual records within each of the four regions, and what we get are: {[0, 0], [0, 0]}, {[12, 15], [12, 15]}, {[13, 17], [21, 30]}, {[23, 30], [5, 6]}

In the above example we divided a node cube into nine smaller cubes and picked four of them. The next step is to deliver the cubes to master nodes. After maintaining the new smaller cubes, master nodes can direct queries more accurately. After the reconstruction of node cubes, we turn one sparse cube into several smaller but denser cubes, and that is the key factor of efficiency improvement. With this approach we can further filter out more irrelevant queries in query processing.

**Query Processing**

The query processing procedure will not present much difference from that in EMINC. The difference lies in the efficiency. With node cubes with better granularity, the chance of forwarding queries to nodes that don't have relative records will be greatly diminished.

**Index Maintenance**

When a new data record needs to be inserted, we first check if it can be enclosed by one of the existing cubes. If we fail to find such cube, we expand the nearest cube to enclose this record. When a record needs to be deleted, we look for the cube it's in and check if this record is on the vertex of that cube, and if the answer is yes, the node cube shrinks.

However, as the slave node accumulates more and more data update operations, node cubes may need to be updated since the data distribution within a node cube may be sparse or uneven again. And when this happens, we shall need to update that node cube to maintain the efficiency. To achieve this we have to answer two questions: when to update the cube and how to reshape the cube ?

We answer the second question first. The reshaping process is similar to the process of cutting the original single cube into several small cubes. The core problem is how to cut each attribute dimension into small intervals. In this paper we try several methods to do the cutting and compare their performances in our evaluation.

- **Random cutting.** Pick several random value points on the attribute and cut by the points. This cutting method may seems somewhat too "random" to be effective, however, in many occasions the distribution of data also shows certain level of randomness and under that circumstance this method may give good performance, but it is not guaranteed due to the randomness.
- **Equal cutting.** Cut the attribute into several equal intervals. If data insertion operations are controlled by the master in a way that data on each slave node shows approximate hypodispersion, the equal cutting would give good performance.
- **Clustering-based cutting.** Cut the attribute by clustering values on the attribute using clustering algorithms and cut between clusters. On some occasions, data records are inserted in a batch way, a transaction for example. So one batch of records may appear to be a cluster and the total records on this node can be seen as a set of clusters. Under this circumstance, clustering-based cutting method will perform much better than random and equal cutting since it captures the characteristic of data. However, the cost of this method is also higher since a typical clustering algorithms is in $O(n^2)$ time complexity.

No matter what method is used for cutting, we should stop cutting when the total number of nodes cubes reaches a certain amount because the number of cubes should be kept relatively small comparing to the number of records since large number of cubes will bring down the efficiency of the master nodes.

## 4.5 Cost Estimation based Update Strategy

Now we go back to the first problem of when to do the reshaping of node cubes. As we can see that updating node cubes can give great benefit to query processing, but the cost of updating is also nontrivial since even the fastest cutting method is in $O(n)$ time complexity where $n$ is the number of data records on this slave node. So when to do the update depends on the comparison of benefit and cost of doing so. So the basic idea is: benefit > cost.

Here we propose a cost-estimation-based approach to handle the cube update problem. First we introduce several concepts and parameters we will need for estimation.

*Definition 6.* **Volume** of a node cube is defined as the maximum number of unique records that this cube can cover. We note volume of a cube by $v$.

**Example 5 :** For the node cube {[1, 11], [2, 5]}, the volume is (11-1)*(5-2) = 30.

To simplify the discussion, we make the following assumption:

*Assumption 1.* The amount of queries forwarded to each slave node is proportional to the total volume of all the node cubes of the slave node.

This assumption is reasonable since the bigger is the total volume, the more data records it is likely to hold. Thus more queries are directed to this node. Then we can easily conclude that in the process of reshaping one node cube into one or more smaller cubes, the smaller the total volume of the small cubes is, the better. This also means that the cutting method must be able to decrease the volume otherwise it makes no sense to do the update.

For each node cube, we use $nq$ to denote the number of queries whose query cubes intersect with this cube in each time unit. We can see that $nq$ describes the contribution of this cube to the query load on this node. And from the above assumption we can see that $nq$ is proportional to the volume $v$ of the cube. We use $qt$ to denote the average time needed to process a query on this node. $mt$ is used to denote the time needed to do a update of cube. Those parameters could be maintained by the cloud platform as metadata.

To make the benefit and cost of update comparable, we express both of them in the metric of number of queries. In other words, we express benefits by how many unrelated queries we can avoid after the update, and express cost by how many queries we could have processed if we use the update time for answering queries.

Benefit of the update can be evaluated by the number of queries that will no longer be forwarded to this node due to this update. And recall that the number of queries is proportional to the volume of cube, so we have:

With the metric of number of query, we express benefit of an update as:

$$benefit = (\delta v/v) \times nq \times \delta T \quad (1)$$

$\delta v$ refers to the decrement of volume after the update, so $\delta v/v \times nq$ means the amount of query that will no longer be forwarded to this node due to this update in one time unit since the amount of query is proportional to the volume. We denote the $\delta v/v$ as the *benefit ratio* of this update since it tells the percentage of queries this update can save. $\delta T$ is the time span from now to when next update happens. So the benefit of this update can be measured by how many irrelative queries we can avoid.

The time cost of the whole reshaping procedure consists of two major parts: reshape the cube into one or more cubes and insert the new cubes to the R-tree on master nodes. As we mentioned that the

number of cubes is kept relatively small. Thus, the cost of inserting them into the R-tree on master nodes will also be neglectable comparing to the reshaping time which is often at least proportional to the size of data records (we express this trivial cost as $\epsilon$). So what we really care is the time cost of reshaping. Using the query number metric, we express cost as:

$$cost = (mt + \epsilon)/qt \approx mt/qt \quad (2)$$

We use an iterative two phase approach for the update strategy. After each update, we first calculate a minimal time span before the next update could happen - the $\delta T$ we introduced. When the time span expires, we calculate the benefit of doing the next update, and if the benefit is acceptable based on several factors, we do the update, and if the benefit is not qualified, we wait another $\delta T$ and check it again.

Substitute formula 2 and 3 into formula 1 we get

$$(\delta v/v) \times nq \times \delta T > mt/qt \quad (3)$$

Reform formula 4 we have

$$\delta T > (mt \times v)/(qt \times \delta v \times nq) \quad (4)$$

And this is the condition that $\delta T$ must meet to ensure the benefit. All the parameters concerned could be got from the last update process or maintained by the platform.

And when the $\delta T$ expires, we have to check each of the descendant cube if we need to update them. The reason for this check is highly dependant on several aspects such as amount of data update operation, expected benefit of next update, performance requirement of the platform and so on.

- **Record update frequency.** If there has not been many update operation since last cube update, we may gain little from another update.
- **Expected benefit ratio.** Expected benefit ratio refers to the benefit ratio of next update if we do the update now, and it is also calculated based on estimation. For example, if the operations occurred didn't quite change the distribution of data in the cube, the expected benefit of doing an reshaping will be pretty small.
- **Performance requirement.** Although we use a $\delta T$ to assure each update being enough utilized, the real time cost of the update process is still inevitable as the update will slow down the real time query processing. So if the performance requirement of the platform is hight, the cube update should not be frequent.

The factors are highly dependant on the specific requirement of the platform and application. We leave the study of combining these factors as future work.

From the discussion we can summarize the process of doing update after one update as algorithm 10 :

Now we can see that after the first update of node cube, the index maintenance could continue as we discussed. The only remaining problem is when to perform the first update of node cube. In fact, when to do the first update has high correlation with what we discussed in the checking phase. That is, the environment and requirement of the application and platform must be carefully studied in order to make the decision. Due to limit of time and length of this paper, we will take this as a future work.

**Algorithm 10** Deciding next update

```
1: Time t = calculate minimal time span;
2: wait(t);
3: boolean isToUpdate = check();
4:
5: while (isToUpdate == false) do
6:     wait(t);
7:     isToUpdate = check();
8: end while
9:
10: if (cube number below limit) then
11:     updateCube();
12:     handle new cubes in R-tree on master nodes;
13: end if
```

# 5. EVALUATION

We now evaluate the performance and scalability of our multi-dimensional index in cloud computing systems. Our testing infrastructure includes 6 machines which are connected together to simulate cloud computing platforms. Communication bandwidth was 1Gps. Each machine had a 2.33GHz Intel Core2 Quad CPU, 4GB of main memory, and a 320G disk. Machines ran Ubuntu 9.04 Server OS.

We use this infrastructure to simulate different sizes of cloud computing systems. We conducted 10 simulation experiments, ranging from 100 nodes to 1000 nodes. Each time 100 more nodes are considered to be added into the cloud computing system. In our infrastructure, one machine plays the role of master nodes and store metadata and control distribution. Each of the other five machines simulates 100 to 1000 slave nodes.

We design two sets of experiments to evaluate our multi-dimensional index's performance and scalability. One is point query (equivalence queries), the other is range query with selectivity being about one ten thousandth. Respectively, we measured the query answer time through using scan, multi-dimensional index with a node cube (EMINC), multi-dimensional index with fine-grained granularity cube(extended EMINC) by random cutting, Equal cutting, K-Means clustering cutting, DBScan clustering cutting. For each experiment, we obtain the result based on 3 runs.

Firstly, we used six methods to execute the point query. Scan method used Map-Reduce functions to scan data on every slave node. EMINC method build a KD-tree on every slave node and construct a node cube for R-Tree on the master node. EEMINC method used several methods to do the cutting. We adopt two clustering algorithms: K-Means clustering and DBScan clustering. The first is a classical clustering algorithm, and the second is a density based clustering algorithm. These cutting algorithms make better granularity and improve query processing efficiency. Figure 4 shows the point query experiment result. As can be seen, EMINC can solve the multi-dimensional query inefficient problem, but EEMINC perform much better than table scan and EMINC. Equal cutting, random cutting and K-Means clustering cutting have similar performance in our dataset. These three methods answered the point query in 1 thousand nodes and 10 million records only cost 40 ~ 50 ms, which shows our method is efficient. Figure 4 illustrates the scalability of our methods. This graph show that our multi-dimensional distributed index scales almost linearly with the number of nodes in the system.

Secondly, we used six methods to execute the range query. Each query got one ten thousandth of the size of all records. Figure 5 shows the range query experiment result. Compared with the cost of point access, the range query execution time shows that our

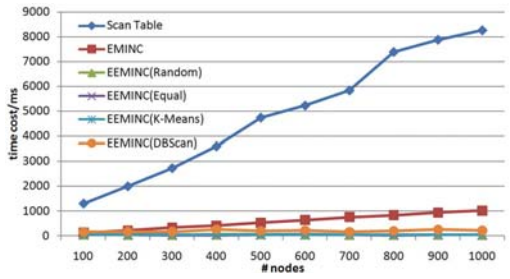



**Figure 4: Point Query Experiment Result**

multi-dimensional distributed index can also support range query efficiently. And EEMINC method is still the most effective index to answer these queries. Figure 5 shows our methods are high availability and scale to hundreds of nodes.

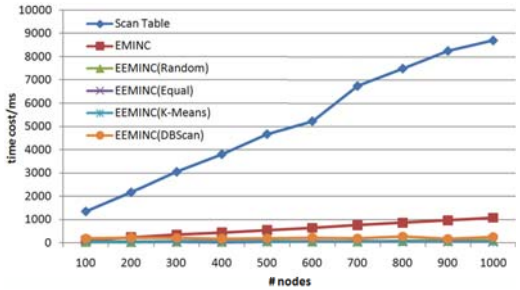



**Figure 5: Range Query Experiment Result**

## 6. CONCLUSION

In this paper we presented EMINC and EEMINC for building efficient multi-dimensional index in Cloud platform. We used the combination of R-tree and KD-tree to support the index structure. We developed the node bounding technique to reduce query processing cost on the Cloud platform. In order to maintain efficiency of the index, we proposed a cost estimation-based approach for index update. And we proved the efficacy of our approach with vast experiment.

For future work, we will study how to cut node cube according to data distribution in the cube to achieve better performance, both index building performance and query processing performance. We also plan to complete the cost estimation model by formalize the check phase of our two-phase estimation approach. And the approaches we proposed didn't give much attention to multiple replicas of data and left that to the underlying file system, however, if we take that into consideration the efficiency and stability of the index can be further enhanced.

## 7. ACKNOWLEDGMENTS

The authors thank anonymous reviewers for their constructive comments. This research was partially supported by the grants from 863 National High-Tech Research and Development Plan of China (No: 2007AA01Z155, 2009AA01Z133, 2009AA011904), National Science Foundation of China (NSFC) under the number (No.60833005) and Key Project in Ministry of Education (No: 109004).

## 8. REFERENCES

[1] Timothy. (2008, July) Multiple experts try defining cloud computing. [Online]. Available: http://tech.slashdot.org/article.pl?sid=08/07/17/2117221

[2] M. Lynch. Amazon elastic compute cloud (amazon ec2). [Online]. Available: http://aws.amazon.com/ec2/

[3] IBM. Ibm introduces ready-to-use cloud computing. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

[4] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107–113, January 2008.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A distributed storage system for structured data," in *Proceedings of the 7th Conference on USENIX Symposium on Operating Systems Design and Implementation*, Seattle, Washington, November 2006, pp. 205–218.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of SOSP'03*, New York, USA, December 2003, pp. 29–43.

[7] Hadoop. [Online]. Available: http://hadoop.apache.org

[8] M. K. Aguilera, W. Golab, and M. A. Shah, "A practical scalable distributed b-tree," in *Proceedings of VLDB'08*, Auckland, New Zealand, August 2008, pp. 598–609.

[9] H. chih Yang and D. S. Parker, "Traverse: Simplified indexing on large map-reduce-merge clusters," in *Proceedings of DASFAA 2009*, Brisbane, Australia, April 2009, pp. 308–322.

[10] M. Aguilera, A. Merchant, M. Shah, A. Veitch, and C. Karamanolis, "Sinfonia: a new paradigm for building scalable distributed systems," in *Proceeding of SOSP'07*, Washington, USA, October 2007, pp. 159–174.

[11] S. Wu and K.-L. Wu, "An indexing framework for efficient retrieval on the cloud," *IEEE Data Eng. Bull.*, vol. 32, pp. 75–82, 2009.

[12] T. Johnson and A. Colbrook, "A distributed data-balanced dictionary based on the b-link tree," in *Proceeding of IPPS'92*, California, USA, March 1992, pp. 319–324.

[13] J. MacCormick, N. Murphy, M. Najork, C. Thekkath, and L. Zhou, "Boxwood: Abstractions as the foundation for storage infrastructure," in *Proceeding of OSDI'04*, San Francisco, California, USA, December 2004, pp. 105–120.

[14] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of SIGCOMM'01*, San Diego, California, USA, August 2001, pp. 149–160.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of SIGCOMM'01*, San Diego, California, United States, August 2001, pp. 161–172.

[16] A. Andrzejak and Z. Xu, "Scalable, efficient range queries for grid information services," in *Proceedings of the 2nd International Conference on Peer-to-Peer Computing (P2P 2002)*, LinkÃűping, Sweden, September 2002, pp. 33–40.

[17] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The r -tree: A dynamic index for multi-dimensional objects," in *The VLDB Journal*, 1987, pp. 507–518.

[18] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database System Implementation*. Upper Saddle River, NJ, USA: Prentice Hall, Inc, 1999.

In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009): 1437-1440, November 2-6, 2009, Hong Kong, China

# Supporting Context-based Query in Personal DataSpace

Yukun Li
School of Information
Renmin University of China
Beijing, China
liyukun@ruc.edu.cn

Xiaofeng Meng
School of Information
Renmin University of China
Beijing, China
xfmeng@ruc.edu.cn

## ABSTRACT

Many users need to refer to content in existing files (pictures, tables, emails, web pages and etc.) when they write documents(programs, presentations, proposals and etc.), and often need to revisit these referenced files for review, revision or reconfirmation. Therefore it is meaningful to discover an approach to help users revisit these references effectively. Traditional approaches(file explorer, desktop search, and etc.) fail to work in this case. In this paper, we propose an efficient solution for this problem. We firstly define a new personal data relationship: Context-based Reference(CR), which is generated by user behaviors. We also propose efficient methods to identify CR relationship and present a new type of query based on it: Context-based Query(C-Query), which helps users efficiently revisit personal documents based on CR relationship. Our experiments validate the effectiveness and efficiency of our methods.

## Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous

## General Terms

Algorithms, Human Factors, Performance

## Keywords

Context, Query, Personal DataSpace

## 1. INTRODUCTION

With development of information technology, more and more personal data items are collected, and Personal Information Management (PIM) [1] becomes a critical problem and a promising research area. Studies show that many personal data accesses($> 58\%$) are "revisit" [3, 4, 5], and "meaningful" data relationships(senderOf, authorOf, publishedIn and etc.) can help users relocate expected items more effectively [2]. However, there are two basic questions need to be answered: (1)what relationships are "meaningful" and (2) how to identify these "meaningful" relationships.

Since the aim of identifying data relationships is to improve effectiveness of data query, the definition of "meaningful" depends on user query requirements. When users produce personal documents(programs, presentations, proposals, and etc.), they often refer to some contents in existing files. In addition, when a user accesses one of her documents or redo a task, she often needs to revisit its references for revision or reconfirmation. Therefore "referenceOf" is one of the "meaningful" relationships of personal data.

The popular tools used by persons to revisit expected documents are folder explorer and desktop search. Folder explorer demands users remember path and name of the expected files. If a user can only remember fuzzy information, she has to try possible paths many times. Therefore folder explorer can not work well in this case. Desktop search demands users remember keywords included by the expected files, which does not work well when a user can not remember exact keywords.

There are also some works on Personal DataSpace(PDS) model [6], personal data integration [7, 8], index [9] and query [11]. But all these works focus on improving efficiency of personal data operation by identifying objective associations of items(senderOf, authorOf, and etc.). Our work is different, we focus on proposing a new personal data relationship based on user behaviors and a new type of query. Our main contributions can be summarized as below.

- Propose a new semantic relationship between personal data items: *Context-Based Reference(CR)*, which is generated by user behaviors. We also propose an effective method for identifying CR relationship.
- Propose a new type of query in PDS: *Context-based Query(C-Query)*, which is based on CR relationship. We give a solution framework of C-Query and propose an efficient approach for C-Query processing.

The rest of this paper is organized as follows: In Section 2, we give a solution framework. In Section 3, we describe the algorithms for identifying CR relationship. In Section 4, we introduce the approach for C-Query processing. Section 5 evaluates our measures and section 6 concludes this paper.

## 2. SOLUTION FRAMEWORK

As shown in Figure 1. Our solution framework includes four parts: CR Database(CDB), Context-based Reference Relation(CRR) Identifier, C-Query Engine and Query Interface.


*CIKM'09,* November 2–6, 2009, Hong Kong, China.

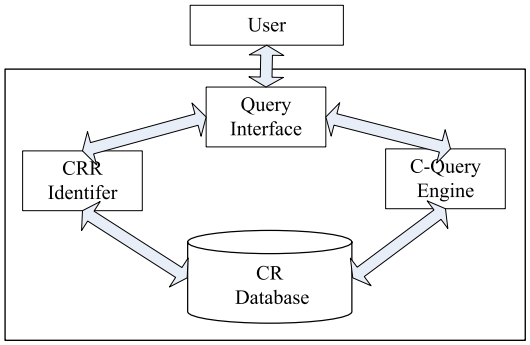



**Figure 1: C-Query Implementation Framework**

- **CR Database:** It is the data structure for describing personal data items and data relationships. We take several relation tables to store CR relationship, because relation tables are simply implemented and work at a high performance.
- **CRR identifier:** It monitors user operations. When a user conducts an operation, it captures user operation and update CR database in time.
- **C-Query Engine:** It handles user query and produces results. When a user submits a C-Query requirement, it produces result based on the CR database and user input.
- **Query Interface** It is utilized to handle user input and display query results. To help users relocate expected items quickly, it allows users to refine query results easily.

In this framework, CRR identifier is the basis of our solution, and is also a big challenge, because (1) there is no explicit information for mining this relationship and (2) the approach for identifying CRR shouldn't increase users' burden. Studies [10] show that the two items accessed continuously often has associations. Inspired from this idea, we propose a method for identifying CR relationship based on user behaviors. To make it more clear, we first introduce the following concepts.

Definition 2.1 (Personal Data Item). *A personal data item(PDI) is the basic element of personal data, and is the smallest unit of personal data operation(read, modify, delete, and etc.).*

There are multiple data relationships among personal data items, such as senderOf, authorOf, referenceOf, and so on. Based on personal data items and their relationships, we propose a new concept: Personal DataSpace.

Definition 2.2 (Personal DataSpace). *A Personal dataspace $\mathcal{D}$ is described as a 2-tuple ($\mathcal{N}$, $\mathcal{R}$), where $\mathcal{N}$ is a set of personal data items and $\mathcal{R}$ is a set of personal data relationships.*

Definition 2.3 (Context-based Reference Relation). *We denote it as $R^{CR}(I_1, I_2, U)$, where $I_1$ and $I_2$ are two personal data items, and $U$ is a user, which means there is a reference relationship between $I_1$ and $I_2$, which is generated by activities of $U$.*

# 3. CR-RELATIONSHIP IDENTIFYING

In this section we first overview CR relationship, then present algorithms for identifying CR-Relationship.

## 3.1 Overview CR Relationship

To tackle the problem of absence of public personal data set, we implement a prototype to capture user access behaviors(operations on desktop, email box and web pages). We run it in personal computers of five persons of our group, and obtain a data set, which includes the access logs of the five users in two months. Based on analyzing these user access logs we propose a new concept: *Time Sequential List.*

Definition 3.1 (Time Sequential List). *A Time Sequential List(TSL) is an item list ordered by access sequence. We say it $(I_1, I_2, ..., I_n)$, where $I_i$ is a personal data item, and $\forall i$, if $1 \leq i \leq n-1$, $I_i <> I_{i+1}$.*

In a TSL, there is no two sequent items mapping to a same item. Figure 2 shows an example of time sequential list. By analyzing the access logs of the five users, we discover three types of CR relationship: Sequence Adjacent Relation(SAR), Sequential Inclusive Relation(SIR) and Lineage Relation(LR). To make them clear, we define them as below.

Definition 3.2 (Sequencial Adjacent Relation). *We denote it as binary relation. Let $I_i$ and $I_j$ are two items of PDS, if $I_i$ and $I_j$ appear in TSL sequentially, $(I_i, I_j) \in R^{SAR}$.*

We define SAR as a symmetrical relation, it means if $(A, B) \in R^{SAR}, (B, A) \in R^{SAR}$. Take the access list shown in figure 2 for example, A and C are accessed frequently, then $(A, C) \in R^{SAR}$ and $(C, A) \in R^{SAR}$.

Definition 3.3 (Item Sequential Loop). *Let $L'$ be a TSL and $L' = (X_1, X_2, ..., X_n)$. If $L'' = (X_i, X_{i+1}, ..., X_j)$ is a sub list of $L'$, $j - i \geq 2$ and $X_i.item = X_j.item$. We call $L''$ an item sequential loop(ISL). We call $X_i.item$ the master item, and call the items of $\{X_{i+1}, ..., X_{j-1}\}$ slave items.*

In the example shown by figure 2, there are following ISLs (A,B,A), (A,C,D,A), (B,E,F,G,B), and so on.

Definition 3.4 (Sequential Inclusive Relation). *A sequential inclusive relation $R^{SIR}$ is a binary relation. Let $L'$ be an item sequential loop, $X'$ is the master item, and $Y_1, Y_2, .., Y_m$ are the slave items of it, $\{(X', Y_i) | 1 \leq i \leq m\} \subseteq R^{SIR}$.*

In the example shown by figure 2, $(A, C, D, A)$ is a ISL, then $\{(A, C), (A, D)\} \in R^{SIR}$.

Definition 3.5 (Lineage Relation). *We denote it as $R^{LR}(I_1, I_2)$, where $I_1$ and $I_2$ are two items of PDS, $R^{LR}(I_1, I_2)$ denotes $I_1$ and $I_2$ are two versions of a same personal data item. We define LR as symmetrical relation.*

Figure 2 shows an example of user access sequential list, where A is the early version of H, therefore $(A, H) \in R^{LR}$ and $(H, A) \in R^{LR}$. LR is also generated by user behaviors(copy to, save as, and etc.).

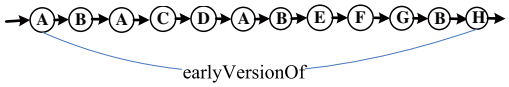



**Figure 2: An example of time sequential list**

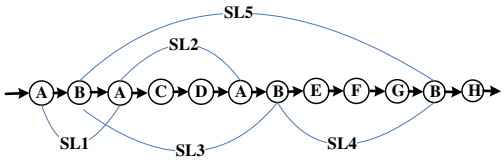



**Figure 3: Sequential loop examples**

## 3.2 Identify Sequential Adjacent Relation

We take a triple set $TS = \{(x_i, x_j, w)\}$ to specify SAR relationship, where $x_i$ and $x_j$ are two items and $w$ is the weight of $R^{SAR}(x_i, x_j)$. We define $w$ as the times the two items orderly appear in ISL. Based on the access sequence list, we can construct TS easily. Its input is an item access list $(X_1, X_2, ..., X_n)$, and its output is a triple set $T'$. In the list each $X_i$ represents an operation, and $X_i.I$ represents the item referenced by $X_i$. Firstly, we scan the items of the access list one by one, if $(X_i.I, X_{i+1}.I) \in T'$, $W(X_i.I, X_{i+1}.I)$ is added by 1, otherwise we insert a new tuple $(X_i.I, X_{i+1}.I)$, and set its weight as 1. It is an incremental process to build TS. Every time when a new operation is monitored, TS will be updated at once.

## 3.3 Identify Sequential Inclusive Relation

Based on Item Sequence Loop(ISL), we can derive Sequence Inclusive Relation(SIR). In the definition of ISL, we do not set limitation for the length of ISL, obviously it results in low precision. Therefore we propose a new concept.

DEFINITION 3.6 (MINIMUM SEQUENTIAL LOOP). *Let $L'$ be a SL, and $\nexists L''$, $L''$ be a SL and $L'' \subset L'$, we call $L'$ a minimum sequential loop(MSL).*

For example, as shown in figure 3, $SL_2$ is included in $SL_3$, and $SL_3$ is included in $SL_5$, thus $SL_3$ and $SL_5$ are not MSL. Because there are not SLs in $SL_1$, $SL_2$ and $SL_4$, they are MSLs.

Our method for identifying SIR is based on MSL. Let $(X_1, X_2, ..., X_n)$ be a time sequence list, and $(I_1, I_2, ..., I_n)$ is the corresponding item list. Assume a new operation $X_{n+1}$ is monitored, and its item is $I_{n+1}$, we scan backwards to find a MSL mastered by the new item $I_{n+1}$. When we find the nearest $X_i$, where $Xi.I = X_{n+1}.I$, and there does not exist a SL in the list $(X_i, X_{i+1}, ..., X_{n+1})$, it means we find a MSL $(X_i, X_{i+1}, ..., X_n, X_{n+1})$, and we can identify the following inclusive relations: $\{(I_{n+1}, I_{i+1}), (I_{n+1}, I_{i+2}), ..., (I_{n+1}, I_n)\}$. The same as SAR, We take a triple set to describe SIR.

## 3.4 Identify Lineage Relation

As the naive method, we can identify it by monitoring the special operations of users(copy to, save as, and etc.). Because this method depends on monitoring specific applications, it is a challenging problem to monitor all possible applications. By analyzing user access logs we find there is a high similarity between the names of two personal files which are two different versions of one document. And users prefer to name different versions of a document with similar strings, and tend to distinct them with prefix or postfix. Based on the observation, we present an edit distance-based approach to decide LR by computing the name similarity of two files. For the reason of space limitation, we do not introduce it in details here.

# 4. QUERY PROCESSING

After identifying CR relationship, we can revisit personal data items based on it. In this section, we introduce the processing of C-Query. In our method, we take three adjacency matrixes $M^{SAR}$ , $M^{SIR}$ and $M^{LR}$ to specify the three relationships SAR, SIR and LR. Therefore we can compute the query results based on the following formula:

$$M^R = M^I \times M^{LR} \times (M^{SAR} + M^{SIR})$$

Here $M^I$ is an entry vector $(x_1, x_2, ..., x_n)$, if $I_i$ belongs to the input items, $x_i = 1$, else $x_i = 0$. Based on the formula, we can get a result vector $M^R$. Let t be the threshold predefined, if $M^R(i) > t$ then $I_i$ is one item of the query results. To get a high recall here we take a relax policy: if $M^R(i) > 0$, we think item $I_i$ belongs to the result set. It means we take all "suspicious" items as results.

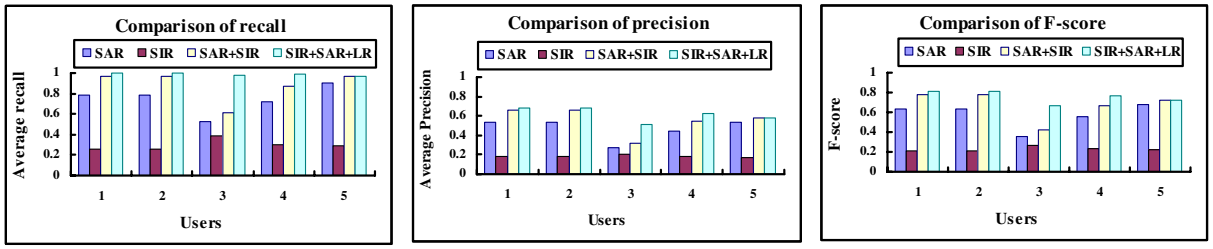
According to the characteristics of C-Query, we design a friendly and flexible interface. Users can input an existing item, or select it by exploring personal resources. It also provides multiple ways for users to refine query results, such as filter or sort the results based on type, access time, keywords, and etc. If the input is a multi-version item, it can display all versions of it and all references of each version.

# 5. EVALUATION

We selected 5 students(2 undergraduate students, 2 master students and 1 PhD student) as participants of our experiments. They run our prototype on their computers. By this way we collected a data set for experiments. We collected two-month access logs(include accesses to desktop files, emails and web pages) of each participant. Table 1 shows the specification of the data set. The meaning of each column is specified as follows.

- $\mathcal{S}_{logs}$ is the number of tuples of user access log file.
- $\mathcal{S}_{items}$ is the number of items which has been accessed by user $U_i$ in the two months.
- $\mathcal{S}_{LR}$ is the number of elements in the set of Item Lineage Relationship(LR).
- $\mathcal{S}_{SAR}$ is the number of elements in the set of Sequencial Adjacent Relation(SAR).
- $\mathcal{S}_{SIR}$ is the number of elements in the set of Sequencial Inclusive Relationship(SIR).

Our aim is to help users revisit personal documents based on CR relationship. Therefore effectiveness and efficiency are the key factors of our approach. Let $I'$ be the input item, based on the three CR relationships(SAR, SIR and LR), we derive several algorithms to identify CR relationship. (1) SIR. It takes the items with SIR to $I'$ as results. (2) SAR. It takes the items with SAR to $I'$ as results.




(a) (b) (c)

**Figure 4: Evaluation of recall, precision and F-score**

**Table 1: Specification on dataset of experiments**

| $User$ | $\mathcal{S}_{logs}$ | $\mathcal{S}_{items}$ | $\mathcal{S}_{LR}$ | $\mathcal{S}_{SAR}$ | $\mathcal{S}_{SIR}$ |
|---|---|---|---|---|---|
| $User_1$ | 2314 | 613 | 22 | 1690 | 250 |
| $User_2$ | 944 | 415 | 13 | 527 | 186 |
| $User_3$ | 2012 | 792 | 27 | 2089 | 390 |
| $User_4$ | 3052 | 915 | 19 | 2527 | 673 |
| $User_5$ | 802 | 223 | 6 | 539 | 93 |

(3)SIR+SAR. It takes the items with SIR or SAR to $I'$ as results. (4)SIR+SAR+LR. It takes the items with SIR or SAR or LR to $I'$ as results.

For each user, we select ten "representative" documents from her access logs. "Representative" means that the selections should not only include "lightweight" personal documents, but also contain some "heavyweight" documents, which cost users more energy, such as paper drafts, presentation slides and etc. These documents often have more Context-based References. We deliver the selected documents to participants and ask them give "standard answer" to each document.

To test the effectiveness of our methods for identifying CR relationship, we take recall and precision to evaluate. We take each selected document as input, and assume that each user $U_i$ submits 10 C-Queries. By comparing the results produced by our algorithms and the right answer given by users, we can compute the recall and precision of each algorithm. We take traditional F-measure method to compute F-score of each method and give evaluation of them.

Figure 4 shows the results of our experiments. Figure 4(a) compares recall of the four methods, Figure 4(b) compares their precision, and Figure 4(c) compare their F-score based on F-measures. The results show that the SIR+SAR+LR method has the best effectiveness. It shows that although we take a relax method, the average precision still reach 60%, and the average recall is more than 90%.

# 6. CONCLUSIONS

In this paper we propose a new semantic relationship *Context-based reference(CR)* and present a new type of query of PDS *Context-based Query(C-Query)*. We also propose an efficient method to identify CR relationship based on user operation logs, and present the processing method of C-Query. This is only a preliminary work on supporting context-based query in personal dataspace. In the future, we will try to improve the precision of identifying CR relationship by considering more user-related information, and will study the ranking approaches of C-Query results.

# 7. ACKNOWLEDGMENTS

This research was partially supported by the grants from the National High-Tech Research and Development Plan of China (No:2007AA01Z155, 2009AA011904); the Natural Science Foundation of China under grant number 60833005. And the authors would like thank Xin Dong, Xiaodong Zhou and Wei Lu for some discussions regarding this work.

# 8. REFERENCES

[1] W. Jones and H. Bruce. A Report on the NSF-Sponsored Workshop on Personal Information Management. PIM worshop 2005.

[2] M. J. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: A New Abstraction for Information Management. SIGMOD Record, 34(4):27-33, 2005.

[3] L. Catledge. and J. Pitkow. Characterizing browsing strategies in the World Wide Web. Computer Networks and ISDN Systems, 27(6), 1065-1073, 1995.

[4] B. McKenzie and A Cockburn. An empirical analysis of web page revisitation. In Proceedings of the 34th International Conference on System Science (HICSS34), 2001.

[5] L. Tauscher and S. Greenberg. How people revisit Web pages: Empirical findings and implications for the design of history systems. International Journal of Human Computer Studies, 47(1), 97-138,1997.

[6] J.-P. Dittrich and M.A.V. Salles. iDM: A Unified and Versatile Data Model for Personal Dataspace Management. VLDB 2006: 367-378

[7] X. Dong and A.Y. Halevy. A Platform for Personal Information Management and Integration. CIDR 2005:119-130.

[8] M.A.V. Salles, J.-P. Dittrich, S.K. Karakashian, O.R. Girard and L. Blunschi. iTrails: Pay-as-you-go Information Integration in Dataspaces, VLDB 2007: 663-674.

[9] X. Dong and A. Halevy. Indexing Dataspace. SIGMOD 2007: 43-54.

[10] P.-A. Chirita and W. Nejdl. Analyzing User Behavior to Rank Desktop Items. SPIRE 2006: 86-97.

[11] C. Peery, W. Wang, A. Marian and T. D. Nguyen. Multi-Dimensional Search for Personal Information Management Systems. EDBT 2008.

2009 Fifth International Conference on Semantics, Knowledge and Grid

# Exploring Personal CoreSpace for DataSpace Management

Yukun Li and Xiaofeng Meng

*School of Information, Renmin University of China, Beijing, China*
{liyukun,xfmeng}@ruc.edu.cn

***Abstract*— With rapid increment of personal data amount, how to efficiently search Personal DataSpace(PDS) becomes an interesting and promising research topic. Popular methods include folder explorer, desktop search tools, and etc. Because these methods ignore user features, they fail to work well in some cases. For example, sometimes users expect to relocate a personal document based on some fuzzy memory clues, such as its type, access time, and so on. These queries can't be supported well by current personal data management tools.**

**The aim of this paper is to discover effective methods to help users search Personal DataSpace. We take Semantic Link Network(SLN) to describe PDS, and divide the semantic links of PDS into two classes: Objective Semantic Link(OSL) and Memory-based Semantic Link(MSL). Base on MSL, we propose a concept *Personal CoreSpace(PCS)*, which is a classification view of personal resources and is specified as a n-dimensional space based on Resource Space Model(RSM). Furthermore we design an ontology of PCS based on user behavior features, and propose a method to design facet search interfaces for users to explore PCS efficiently. We validate the effectiveness of our methods by implementing a prototype system for PCS exploring.**

## I. Introduction

With rapid increment of personal data amount, people have to manage more and more personal data resources in limited time. Therefore how to efficiently relocate expected items in Personal DataSpace(PDS) [1], [2], [3] becomes an important and interesting research topic. Taking personal desktop resources management for an example, folder explorer is the most popular way [4] for users to refind expected files, but it demands users have knowledge about the path of the files. It is a challenge for ordinary users to remember the exact path of the long-time-unvisited files. Desktop search is another popular tool for relocating expected files, but it works well only when users can recall exact keywords in the aimed documents. Neither of these methods works well to the following queries.

**Query 1:** Find me the picture I developed for MDM2008 one year ago, which type is jpg or vsd, but I can't remember its path and name exactly.

**Query 2:** Find me a file of ppt type about dataspace I copied from others, which was stored in D: of my desktop computer, and I have visited it in the last month.

We often meet the queries like the two examples in everyday life, and the existing methods can't help us to finish them efficiently. These queries have the following characters. First, users can't remember the exact values of their attributes, such as file names, pathes, keywords, and so on. Second, users can only recall some fuzzy clues for querying. For example, "which type is jpg or vsd", "I have visited it in the last month", and etc. So we must discover a method to specify these features to support exploring personal dataspace based on these fuzzy clues, which is the focus of this paper.

### A. Related work

Related works include Personal DataSpace(PDS), Personal Information Management(PIM), Resource Space Model, theories of cognitive psychology, and so on.

Franklin et al. [1], [2] introduced the DataSpace Support Platforms (DSSPs) for next generation data management to fit the new characteristics of data, such as large scale, heterogeneity, evolutionary, and so on. These studies on Personal DataSpace include data model [3], indexing, querying [5] and prototype systems, such as iMemex [6], [7], Semex [8] and so on. In Personal Information Management(PIM) area, there are also some research works and systems, like Haystack [9], MyLifeBits[10], and etc. Desktop search engine [11] is also an important effort to tackle the problem, which allows users to relocate expected items by keywords. All these works concentrate on describing the objective data world, but pay little attention to the role of user features in improving efficiency of data operations, thus they fail to work well to the example queries.

According to cognitive psychology, there are some rules on user memory, which play an important role in managing PDS [12], [13], [14], [15]. For example, different from web search, PDS searches are known-item search, which means the user knows the existence of the searched items, and only want to relocate them for reuse. These knowledge can be utilized to improve the efficiency of searching PDS. Reference[16] proposes a user-centered framework of personal dataspace management systems, and presents to take CoreSpace to describe the set of frequently-accessed personal data items. We extend this concept in this paper so as to allow users access PDS more effectively.

There are also some works on modeling semantic association of data resources, such as Semantic Link Network(SLN) [17], Resource Space Model(RSM) [18], [19], [20], which are general methods on data resources specifi-

978-0-7695-3810-5/09 $26.00 © 2009 IEEE
DOI 10.1109/SKG.2009.46

cation. In this paper, we focus on utilizing these knowledge and techniques to produce an effective approach for searching PDS.

### B. Contribution Summary

In this paper, our contributions can be summarized as below:

The first, we propose to take SLN to represent PDS, and divide the semantic links among PDS into two classes: Objective Semantic Link(OSL) and subjective Memory-based Semantic Link(MSL), further we propose a new concept Personal CoreSpace(PCS), which is a classification view of personal resources and is specified as a n-dimensional space based on Resource Space Model(RSM).

The second, we discover several types of MSLs which can be utilized to help users relocate expected items, and design an ontology of PCS based on user experiences and memory rules, and illustrate its axis and coordinates.

The third, we propose a facet-based search interface of PDS, which can help users perform complex semantic query, and propose a method to translate the PCS ontology into a facet-based search interface. We validate the effectiveness of our methods by implementing a prototype system.

The paper is organized as below, Session II gives an overview of Personal CoreSpace. Session III introduces the design of PCS and an ontology of PCS. Session IV introduces the implementation of our methods. Session V concludes the paper.

## II. Personal CoreSpace Overview

In this section, we overview CoreSpace from the following aspects: features of personal data and operation, Resource Space Model overview and Personal CoreSpace Model.

### A. Features of Personal Data

Different from other data management domain, there are many special features of personal data.

*1) Versatile and heterogeneous:* Personal data come from different data sources, such as Web, email systems, desktop file systems, and so on. All the data items are stored in multi places(Desktop, web, laptop, cellphone, and etc.). So we need to take an uniform method to specify the heterogeneous data sources.

*2) Personalization:* Because of the differences of position, knowledge background, and experience of using computer, persons usually have different habits on organizing personal data resources, such as the style of naming files, the policies for classification, and so on.

*3) Complex structure:* In RDBMS, relations are based on class(table) level. But in PDS, the relations are based on entity(tuple) level. Each entity may have relations with others. for example, a people is both an author of a paper and a sender of an email. It is a challenge for people to design a completive schema to describe these entity-based associations. Therefore PDS demands a more flexible schema.

*4) Evolutionary:* Traditional data management systems are business-oriented, and the updates of data is implemented by manual measures. For example, to a student management system, users only need to update the records(insert new records or update existed records) according to the practical conditions. But in PDS, it is impossible for users to manually update PDS every time when he makes a modification on his data resources, thus PDS needs an evolution mechanism to automatically update PDS with user interactions.

*5) Disordered:* Although people try their best to classify personal data items well, it is a challenge for them to make and maintain a good category of personal data resources because of limitation of people memory and the random of encountering new entities. For example, when a user finds a new useful paper (e.g. *indexing dataspace*) and want to keep it, it isn't easy for him to select a "right" folder to place it.

### B. Features of personal data operation

Because of the data features, there are many differences about data operations between PDS and traditional DBMS.

*1) PayGo Integration:* The construction of traditional database is mostly a manual process, and the data items are always input by users one by one. In PDS, it should be an automatic process to lessen user burdens. There should be a self-learning engine to get the knowledge on what should be kept and how to keep them efficiently. Thus PDS construction is a pay-as-you-go process, which means with user interactions, PDS will become more and more optimized and provide users more and more efficient services.

*2) Known-item relocation:* In PDS, the aim that people keep data items is to reuse them in the future, therefore most data access in PDS is known-item based relocation, which means to relocate one or some existing items, and it is different from Web search and structure-based queries. To web search, people don't know if there are expected results. To RDBMS query, users know the existence of the expected entities, and know exact clues for relocating it, like product ID, student name, and etc. In PDS, because of the limitation of people memory, users often can not remember exact information on expected data items, like the two query examples given in section 1.

*3) Multi query methods:* In traditional RDBMS, because there are stable data schema, there is always a predesigned query interface. For example, a student management system provides query measures based on student ID or student name. But in PDS, there are many distinct query scenarios, which demand different query methods. For example, to relocate a frequently-used items, users may expect a simple browsing way to refind it; To relocate a file untouched for a long time, users need to relocate it by keyword search tools. So PDS systems should provide users multi query methods.

*4) Simple interface:* The aim of Personal DataSpace Management Systems(PDSMS) is to help users efficiently manage personal data resources. Comparing with traditional DBMS, there is no specific administrator, and the user is both system manager and end user. Because most ordinary users haven't

special knowledge on data management, the interface of PDSMS should be as simple as possible.

### C. Resource Space Model Overview

A fundamental problem of personal dataspace management is classification, which means how to effectively classify personal data resources. Thus the first task is to find an effective method to specify classifications of personal data resources.

Resource Space Model(RSM) is proposed by Hai Zhuge [18], [19], [20], which is based on theories of cognitive psychology, focus on specifying, sharing and managing versatile resources with a universal resource view, and forms a completed theory systems. RSM is inspired by the fact that Internet makes web becoming a global data repository, and how to specify and share these sea-size data become a challenge problem. The personal data resources have similar features with web resources, such as versatile, distributed, heterogeneous, evolutionary and so on. Based on it, we propose to take RSM to specify the global view of PDS. Firstly we give an overview on main concepts of RSM [18], [19], [20].

(1) *A resource space* is a n-dimensional space where every point uniquely determines one resource or a set of interrelated resources, denoted as $RS(X1; X2; ...; Xn)$ or just by name $RS$ in simple. $X_i$ is the name of an axis. $X_i = (C_{i1}; C_{i2}; ...; C_{in})$ represents an axis with its coordinates and the order between them. $C$ denotes the coordinate name in form of a noun or a noun phrase. Any name corresponds to a formal or an informal semantic definition in its domain ontology.

(2) *A coordinate C* represents a class of resources, denoted as $R(C)$, a coordinate $C$ is called independent from another coordinate $C'$ if $C$ is neither the synonym nor the near-synonym of $C'$ in the discussion domain ontology.

(3) Two axes are called the same if their names are the same and the names of all the corresponding coordinates are the same in a discussion domain ontology.

(4) If two axes $X_1 = (C_{11}; C_{12}; ...; C_{1n})$ and $X_2 = (C_{21}; C_{22}; ...; C_{2m})$ have the same axis name but have different coordinates, they can be merged into one: $X = X_1 \bigcup X_2 = (C_{11}; C_{12}; ...; C_{1n}; C_{21}; C_{22}; ...; C_{2m})$ whose order consists with the order of $(C_{11}; C_{12}; ...; C_{1n})$ and $(C_{21}; C_{22}; ...; C_{2m})$.

(5) An axis $X$ can be split into two axes $X'$ and $X''$ by dividing the coordinate set of $X$ into two: the coordinate set of $X'$ and that of $X''$, such that $X = X \bigcup X''$.

**Definition 1.** Let $X = (C_1; C_2; ...; C_n)$ be an axis and $C'_i$ be a coordinate at another axis $X'$, we say that $X$ fine classifies $C'_i$ (denoted as $C'_i/X$) if and only if:

1. $R(C_1) \bigcap R(C'_i) \neq NULL, R(C_2) \bigcap R(C'_i) \neq NULL), ...,$ *and* $R(C_n) \bigcap R(C'_i) \neq NULL$;
2. $(R(C_1) \bigcap R(C'_i)) \bigcap (R(C_2) \bigcap R(C'_i)) \bigcap ... \bigcap (R(C_n) \bigcap R(C'_i)) = NULL$;
3. $(R(C_1) \bigcap R(C'_i)) \bigcup (R(C_2) \bigcap R(C'_i)) \bigcup ... \bigcup (R(C_n) \bigcap R(C'_i)) = R(C'_i)$;

As the result of the fine classification, $R(C')$ is classified into n categories: $R(C'_i/X) = \{R(C_1) \bigcap R(C'_i), R(C_2) \bigcap R(C'_i), ..., R(C_n) \bigcap R(C'_i)\}$.

**Definition 2.** For two axes $X = (C_1; C_2; ...; C_n)$ and $X' = (C'_1, C'_2, ..., C'_m)$, we say that $X$ fine classifies $X'$ (denoted as $X'/X$) if and only if $X$ fine classifies $C'_1; C'_2; ...; C'_m$.

**Definition 3.** Two axes $X$ and $X'$ are called orthogonal with each other (denoted as $X \perp X'$) if $X$ fine classifies $X'$ and vice versa, i.e., both $X'/X$ and $X/X'$ hold.

Three normal forms of the resource space are defined for designing resource space. The first-normal-form of a resource space is a resource space and there does not exist name duplication between coordinates at any axis. The second-normal- form of a resource space is a first-normal-form and for any axis, any two coordinates are independent each other. The third-normal-form of a resource space is a second-normal-form and any two axes of it are orthogonal with each other.

These concepts show that RSM focuses on taking a coordinate-based method to describe the classification view of resource space.

### D. Personal CoreSpace Model

RSM provides an effective method to specify personal data resources. but how to decide if a resource belongs to PDS and how to decide which class it belongs to become two main problems. In this section, we will analyze the relationships between users and normal data sources, and propose a user-centered personal resource classification model *CoreSpace*. Firstly we give the following concepts.

*An entity* is an "object" in the real world that is distinguishable from all other objects. In personal dataspace, an entity can be an email, a file, a picture, and so on.

*An attribute* is descriptive properties possessed by an entity, and an entity is represented as a set of attributes. For example, a file A is an entity, and file name, access time, size are its attributes.

*The owner* is an specific entity of personal dataspace, which is both the administrator and end user of PDS. Each personal dataspace only has one owner entity, and we call other entities of PDS *normal entity*.

*A relationship* is an association among PDS entities. For example, we define a relationship *AttachedBy* to represent that a file is attached by an email. Because Owner is a special entity of PDS, we divide relationships of PDS into two classes: (1)relationship between PDS owner and normal entities and (2) relationship among normal entities. There are many owner-entity relationships, such as a document is *written by* owner, an email $E$ is accessed by the owner, etc. We define a basic owner-entity relationship: $knownBy(O, E)$, which means the owner $O$ has seen the entity and known some information on it, which is the precondition of relocation.

*A Personal CoreSpace* is an ordered collection of personal known entities. A Personal CoreSpace is a n-dimensional space where every point uniquely determines one personal entity or a set of personal entities, we denote it as $PCS(X_1; X_2; ...; X_n)$ or just by name $PCS$ in simple. $X_i$

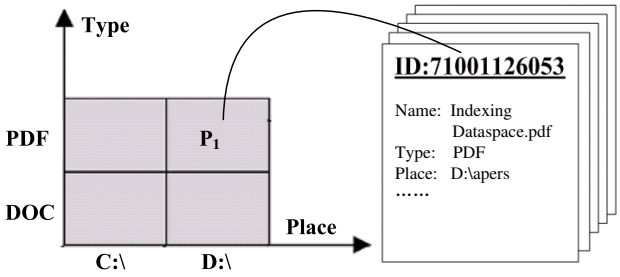



Fig. 1. An example of personal CoreSpace

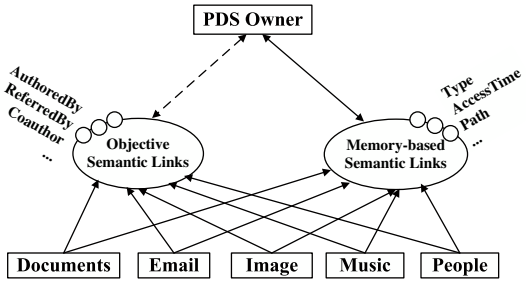



Fig. 2. Semantic links of personal CoreSpace

is the name of an axis. $X_i = (C_{i1}; C_{i2}; ...; C_{in})$ represents an axis with its coordinates and the order between them. $C$ denotes the coordinate name in form of a noun or a noun phrase.

PCS has following meanings. The first, it only includes the personal known entities, which identify a clear boundary of personal data set. The second, it is not a disordered entity set, but has a logical structure of n-dimensional space, where each coordinate can be taken as a classification on the personal data set. The third, CoreSpace is not a storage physical structure, but a set of views on personal data classification.

Figure 1 shows an example of PCS, which has two dimensions: $X_1$ is entity type, and $X_2$ is storage place, where $X_1 = \{PDF, DOC\}$, and $X_2 = \{C :, D :\}$. We can see $X_1$ and $X_2$ partition the personal resources into four parts. The part $P_1$ represents a set of personal files of pdf type located in $D$ : drive. Of course this is only a simple example, the practical $PCS$ will be complex much more.

## III. Personal CoreSpace Design

[18] proposes some principles and methods for designing RSM, based on which, we design PCS by the following steps: analyzing features of SLN of PDS; Design a PCS ontology.

### A. User-centered Semantic Link Network

Semantic Link Network [17] is an effective method to specify resources and their associations. We take it to describe the semantic associations among entities of PCS, and divide the entity associations into two classes. One is the objective associations. For example, the "referenceOf" relationship between two papers, the "authorOf" relationship between a person and a paper, and etc. The other is the associations between objective entities and PDS owner, which depend on the people's memory features, and we call it subjective associations. For example, a user remembers he has accessed file $F'$ in may, 2009, or he remembers the type of $F'$ is "doc" or "txt". There are many works on how to improve effectiveness of PDS query by highlighting objective associations among entities. For example, we can find "referenceOf" relationship of two papers and "co-author" relationship of two persons by entity extraction and entity identify techniques in IR area [21], [22].

According to the two associations of PCS listed above, we divide semantic links of PCS into two classes: Objective Semantic Link(OSL) and Memory-based Semantic Links(MSL), as shown in figure 2. OSL means the objective associations between two objective entities, which is not focus of this paper. MSL is a type of semantic link we define for specifying the associations between objective entities and user memory. For example, let $D'$ be a document of user $U'$, and $U'$ remembers $D'$ is a file of doc type, and is developed by $U'$ in 2008. So we think there are three MSLs $U' \xrightarrow{\alpha} D'$ , $U' \xrightarrow{\beta} D'$ and $U' \xrightarrow{\gamma} D'$, where $\alpha$ means "$U'$ $rememers$ $D'$ $is$ $a$ $doc$ $file$", $\beta$ means "$D'$ $is$ $developed$ $by$ $U'$" and $\gamma$ means "$U'$ $has$ $accessed$ $D'$ $in$ 2008".

Comparing with OSL, MSL is a type of subjective semantic links, which depends on memory of PDS owner. The followings are our selected attributes for MSL according to people's everyday experiences or research results of cognitive psychology.

*1) Natural attributes:* Natural attributes means the objective attributes of an entity, which are independent with user characters, and can be obtained directly. For example, as to a personal desktop document, its file name, path, size are natural attributes. According to experiences of persons, the natural attributes are often used to relocate expected items. For example, users usually relocate expected items by exploring folders and ordering the files based on name, access time, type, and so on.

*2) User-based attributes:* Besides the natural attributes, there are also some user-based attributes, which can't be gotten directly, but is useful for relocating personal documents. According to popular experience, people sometimes expect to refind a document based on some fuzzy clues. For example, when an user want to recall a document, he only remember it hasn't been accessed for a long time, or it is created for a specific task. These attributes must be discovered by analyzing personal behaviors.

Figure 3 shows a classification tree derived from analyzing attributes of personal resources, which is based on classifying personal desktop files. If more data resources are considered, such as emails, web pages, and so on, more attributes should be considered for classification.

### B. Personal CoreSpace Ontology

According to the classification of figure 3, we design a PCS ontology based on RSM, which is a 9-dimensional space. and the axis includes: name, type, access time, size, path, source,

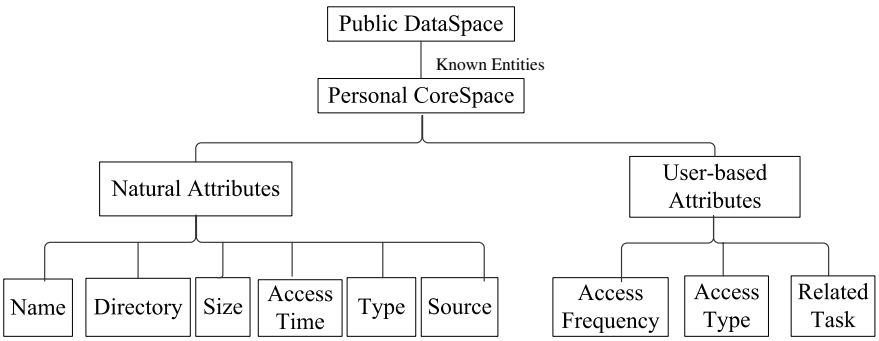



Fig. 3. A classification structure of Personal CoreSpace

access frequency, access type and related task. After deciding the axis, we should consider how to decide the coordinates of each axis. The naive method is to enumerate all possible values for each axis. This method has the following disadvantages. The first, there may be a great number of values for a specific axis, and it will make users inconvenient to explore them. For example, type is an axis, if we list all possible values of it, there may be hundreds of types, but most of which haven't been used by users and aren't helpful for relocating expected items. In fact, an user always prefer to a small number of values to each axis, and their memory on the values has certain rules. So we should select coordinates for each axis based on users' memory rules. We take the personal desktop resources for example to design a PCS ontology, of course it can be extended to more data resources. The axis and their coordinates is described as below.

*Type:* We specify it with a 2-level classification structure, the first level is {Email, Web pages, Picture, Documents}. As to the second level, picture includes jpg, vsd, gif, etc, and the documents includes doc, txt, pdf, and so on, we don't fine classify email and web pages any more. The concrete coordinates of each subclasses depend on the owner features, we construct it based on user access activities. For example, if the owner is a programmer, the document type values may include cpp, java and so on; if the owner is a researcher, the document type values may include pdf, tex, doc, and etc.

*Access Time:* Each desktop file has three time attributes: create time, modify time and access time. According to popular experience, it is a challenge for users to remember the exact create time or modify time on a file, but they can remember approximate information on access time. For example, he possibly remembers "I haven't accessed it in last month". Based on the rule "users' memory on an entity decreases with time going", we classify access time axis as "Today","Yesterday","Last week","Last month","Last year","One year ago".

*Directory:* Directory tree is a natural structure for users to organize personal resources, and path is an important clue for users to relocate expected files. We take the natural directory tree as a axis of PCS, and its coordinates are the actual values of the directory tree.

*Size:* People seldom remember the exact size of a document, but they can remember its possible size. For example, people can remember "it is only a doc file with one page and has a small size". So it is not meaningful to make a strict classification on file size. Similar to the search tools of Windows Systems, we classify it as $\{(0, 10K], (10K, 100K], (100K, 1M], (1M, 10M], (10M, \infty)\}$.

*Source:* There are many data sources devoted to PCS. A document of PCS may be downloaded from web, copied from other people or developed by himself. It isn't practical to ask the PDS owner to precisely mark source of each personal resource. For example, to a file $A$ and $B$, where $A$ is copied from a friend, and $B$ is downloaded from a web site, It is uneasy to automatically distinct them without user interactions. Here we make a rough classification policy. We divide them into two classes: self-developed and cloned from other sources.

*Access Frequency:* It is uneasy for a people to remember exact times a file has been accessed, but he can remember "I frequently access this file" or "I seldom access this file". Therefore we make a partition based on the access times as below: $\{(1, 5], (6, 10], (11, 15], (15, 20], (20, \infty]\}$, where the numbers mean access times.

*Access Type:* A user sometimes wants to explore documents created or modified by himself. When a person relocates a document, maybe he can not remember exact access time of it, but he can recall if he modified it or read it only. Therefore we classify access type as read-only or modified.

*Related tasks:* Task is an important factor for people to query PDS. We define a task as a set of personal documents, and take each task as a coordinate of this axis. How to efficiently identify personal tasks and the tasks' related files are challenge problems, and we will tackle them by another work.

Based on the classification as shown in figure 3 and the coordinates we decide for each axis, we can construct a PCS ontology. Of course, it is only an initial one based on personal desktop documents, if more data resources(Email, web pages, etc) are considered, it will have more dimensions. And with time going, it will be added new axis and new coordinates

increasingly.

### C. Personal CoreSpace Features

From the definition of PCS, we can see it has three features.

*1) User dependency:* Different persons may have different positions, habits and experiences on using computer, therefore their PCSs are different. The differences are shown by the following respects: PDS size, complexity of associations, number of data sources, and so on. So their PCSs should include different axis and coordinates.

*2) Extendability:* Both the axis and coordinates of PCS can be extended easily. With time going, the size of personal data set will rise increasingly, and a more detailed classification may be needed, when the new axis and coordinates can be added easily.

*3) Evolutionary:* The aim of proposing PCS is to release burden of users, therefore its aim is to reduce user's interactions as more as possible. Different from traditional DBMS, the construction and update of PCS are mostly automatic processes. Based on user access activities, new personal entities will be kept into CoreSpace automatically, and the PCS view can evolved with time going.

## IV. CoreSpace Implementation

To evaluate the effectiveness of PCS model, we develop a PCS prototype system based on personal desktop resources. Figure 4 shows the framework of it, which mainly includes the following parts: User behavior monitor, Item identifier and Query processer.

### A. User Behavior Monitor

User Behavior Monitor(UBM) is responsible for discovering the changes of personal desktop. When a user reads or modifies a file, this operation results in changes of its attributes, UBM is to find these changes in time. If these changes can't be found, it will lead to inconsistency of personal resources, and bring problems to queries.

The monitoring program is developed by us to monitor desktop changes by scanning the recent folder. According to our investigation, most users active the "recent folder" option, and we can easily find the the latest update of desktop resources. We can directly get some natural attributes of accessed file, like name, access time, directory, size, and etc. But we can't directly get the attributes of access type, access frequency and related task. As to access type, we compute it by comparing its modify time with the original one recorded in PDS. Access frequency is the access times, and can be easily gotten. How to identify related task of a personal document is another interesting topic, and is not the focus of this work.

### B. PayGo Evolution of Personal CoreSpace

Personal CoreSpace draws a beautiful picture for us to easily relocate expected items. But how to efficiently construct and update PCS are two important problems. As discussed above, it is impossible to ask people update it manually each time when there is change in desktop, so we take a pay-as-you-go method to build PCS and update it efficiently. The related

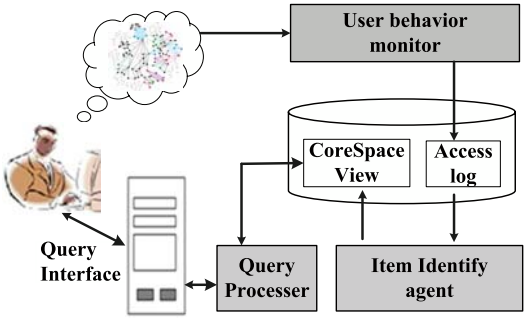



Fig. 4. CoreSpace Implementation Framework

problems on constructing CoreSpace includes: Identify known entities among public dataspace; Identify the Coordinates of each axis of PCS; construct the classification structure.

*1) Identify personal known items:* There are mainly three methods. The naive method is to monitor user behaviors to find new personal resources and keep it into PDS. But this method can't build a PCS soon. To tackle the problem, we propose a method to automatically construct initial PCS, it mines some user preferences features based on recent-accessed files and automatically construct a classifier to identify the known entities among desktop resources. This method is not focus of this paper, and is not introduced in details.

*2) Construct Personal CoreSpace:* Let $S = \{X_1;\ X_2;\ ...;\ X_n\}$, which is the set of personal known items, where $n$ is the number of known items, and each $X_i$ includes the following natural attributes: file name, file size, directory, access time, type, and etc. We construct PCS as below. The first, we summarize the coordinates for each axis based on $S$. For example, based on the type attributes, we can get the set of file types cared by the user, and take it as the coordinates of type axis. By this method, we can decide coordinates for other axis based on natural attributes.

To the attributes that can not be directly obtained by analyzing the entities themselves, we must got them by analyzing user access activities. For example, access frequency and personal task must be computed based on user access activities.

*3) PayGo Evolution of Personal CoreSpace:* Once built, the CoreSpace should have the ability to evolve with users' operations, which means, with people's interactions(pay) PCS should be optimized adaptively(go). By the monitor, PCS system can automatically find the changes of desktop. Once monitoring a user activity, it will do the following operations.

If the file operated is a new one, it will be added into PCS and linked into the proper classes of PCS. For example, if a new file $D : \backslash PIM\backslash PIMsurvey.doc$ is found to be created, and we assume the current time is 2009-01-20, and the size of it is 5K, then it will be linked to the following coordinates: $Type\backslash Documents\backslash Doc$, $Path\backslash D : \backslash PIM$, $Time\backslash 2009-01-20$, $Size\backslash (0, 10K]$, $AccessFrequency\backslash 1$, $AccessType\backslash Modify$, $Task\backslash\{PIM, Surveye\}$, etc.

If UMB finds the operated file is an existing one, the

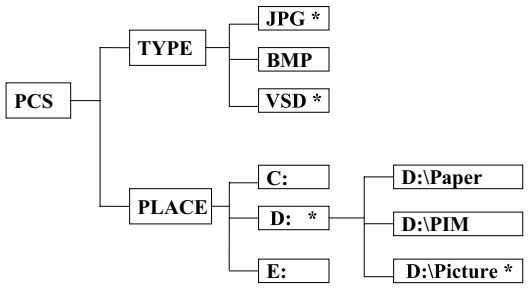



Fig. 5. An example of facet-based query expression

attributes of it will be updated, such as access time, access frequency, related task, and so on.

Through monitoring users behavior, PCS can be automatically updated. We also call it pay-as-you-go integration of personal data resources.

### C. From CoreSpace to Facet Search

PCS is a tree structure, and based on it we can easily design a facet-based hierarchical structure. The method is simple, we take each coordinate $X_i$ as a facet $F_i$, and take its coordinates as the options of facet $F_i$. Based on the hierarchical structure of PCS, we can easily construct a facet-based search interface.

Facet-based search interface provides users a simple and easy method for relocation. Keyword search and structured query language are two popular query methods. The former can't support semantic query, and the latter can't be used easily by average users because its complexity. In PDS, users need a lightweight search method and a friendly search interface, which doesn't only support semantic query, but also is as simple as possible. We leverage it and make a good tradeoff based on facet search. We propose a simple and visualized method to represent simple conditional expression.

Let $X$ and $X'$ are two selected nodes of facet tree, and they can be regarded as two conditional expressions. For example, if X means node "$Type\backslash Doc$" , which represents expression $x.type =' Doc'$. Based on it, we define the following query algebra: If $X$ is brother of $X'$, it means $X \vee X'$; else it means $X \wedge X'$.

Figure 5 shows an example of facet query. The nodes marked with star represent the options selected by user, according to the query algebra we defined above, we can get the following logical expression: $R = \{Xi|(Xi.type = JPG \vee Xi.type = VSD) \wedge Xi.place = "D:\backslash Picture"\}$. It means the user want to relocate some personal data items, which lie in "$D:\backslash Picture$" and have type of vsd or jpg.

To fit users expectations, we design a combined query interface, which support keyword search, facet-based CoreSpace explorer and folder explorer.

### D. CoreSpace Search Implementation

To evaluate the effectiveness of our PCS model, we develop a prototype system CoreSpace. Figure 6 is the query interface of it. The left of it is an area where users input query conditions, the right part is for displaying query results, and the bottom of right area is the input form for keyword search. The system has the following features.

*1) CoreSpace Explorer:* In the default case, the right area displays the files of PCS, which support users rehandle them more. Users can rank them based on access time, type, name and so on.

*2) Facet-based Filter:* Because the result set is often large in the default cases, users can refine the results by filtering them based on facet search interface. We can input simple semantic expression. For example, if we can remember the expected item is a pdf or text file, we can filter all doc and text files of PCS by selecting the "$type \setminus pdf$" option and the "$type \setminus txt$" option, which make it easier for users to relocate what are expected.

*3) Combined Search:* A friendly search interface should support multi query requirements. As shown in figure 6, we design a search interface which combines multi query methods, instead of single CoreSpace explorer. Through the interface, a user can combine keyword search and PCS explorer for relocating more effectively.

### E. Illustration of PCS query

In this section, we utilize the two query examples given in section 1 to illustrate the query process based on our prototype system.

**Query 1:** Find me the picture I developed for MDM2008 one year ago, which type is jpg or vsd, but I can't remember its path and name exactly.

The user can finish it by the following steps. Firstly he searches it by inputting keyword "MDM", there may be a lot of files returned. Then he can filter the results by selecting "$type\backslash jpg$" and "$type\backslash vsd$". If there are still too many items returned, he can further filter the results by selecting "$Operation\backslash\ Have\ modified$" and "$Lastaccess\backslash one - year - ago$" in facet search window. Therefore he can relocate it more easily.

**Query 2:** Find me a file of ppt type about Dataspace I copied from others, which was stored in D: of my desktop computer, and I have visited it in the last month.

To perform this query, the user firstly select "$type\backslash ppt$", then he can select "$Location\backslash D$: ", and select the "$Last\ Access\backslash Last\ month$". Then he can relocate it by exploring the files according to the conditions. If the results size is still large, he can refine it by inputing keyword "Dataspace".

## V. CONCLUSIONS

By utilizing Resource Space Model(RSM) in personal dataspace management area, we propose a personal resource space model: Personal CoreSpace, which is a n-dimensional space of personal resources based on memory rules of people. We define nine axis and their coordinates, and based on it we propose a PCS ontology. Further we present a framework of PCS systems and introduce a prototype system implemented

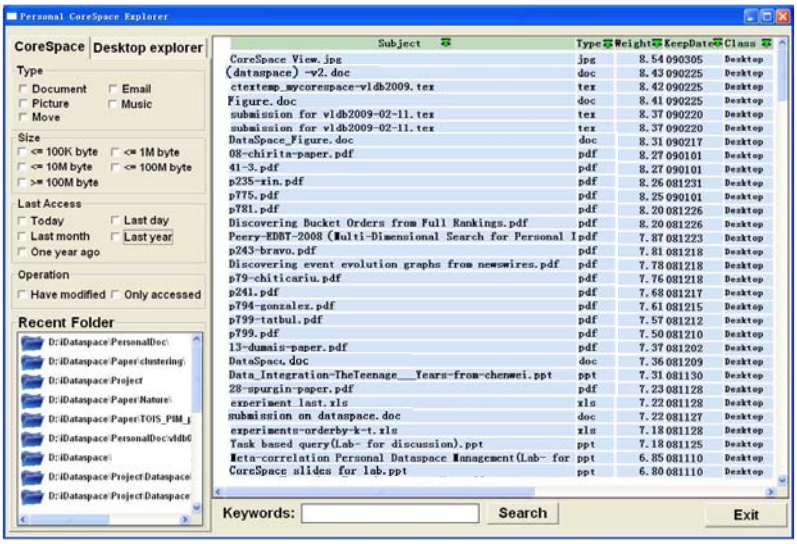



Fig. 6. CoreSpace Search Interface

based on PCS framework, which validates the effectiveness of our idea about exploring PDS based on RSM.

This work highlights some interesting research topics. In the future, we plan to do the following works. The first, in PCS, the coordinates of some axis(such as personal task, and etc.)are not a partition of personal resource set, but a coverage of it, therefore we will try to extend RSM to make it fit the characters of PCS better. The second, we will try to find more rules on user activities to make the PCS ontology more completed. In addition, we will take more personal resources(Email, web pages, and etc.) into consideration in the PDS ontology and our methods and systems.

## ACKNOWLEDGMENTS

This research was partially supported by the grants from National High-Tech Research and Development Plan of China (No: 2007AA01Z155 ,2009AA011904); China National Basic Research and Development Program's Semantic Grid Project (No: 2003CB317000).

## REFERENCES

[1] M. Franklin, A. Halevy, D. Maier. From Databases to Dataspaces: A New Abstraction for Information Management, SIGMOD Record, 34(4):27-33, 2005.

[2] A. Halevy, M. Franklin, D. Maier. Principles of dataspace systems, In *ACM PODS*, 2006, pp.1-9.

[3] J. P. Dittrich and M.A.V. Salles. iDM: A unified and versatile data model for personal dataspace management. In *VLDB*, 2006, pp. 367-378.

[4] O. Bergman, R. Beyth-Marom, R. Nachmias, N. Gradovitch, S. Whittaker. Improved Search Engines and Navigation Preference in Personal Information Management.ACM Transactions on Information Systems, Vol. 26, No. 4, Article 20, Publication date: September 2008.

[5] X. Dong, A. Halevey. Indexing Dataspace. In *ACM SIGMOD*, 2007, pp.43-54.

[6] L. Blunschi, J.-P. Dittrich, O. R. Girard, S. K. Karakashian and M. A. V. Salles. A Dataspace Odyssey: The iMeMex Personal Dataspace Management System(Demo). In *CIDR*, 2007, pp.114-119.

[7] J. P. Dittrich. iMeMex: A Platform for Personal Dataspace Management. In *SIGIR PIM Workshop*, 2006.

[8] X. Dong and A. Halevy. A Platform for Personal Information Management and Integration. In *CIDR*, 2005, pp.119-130.

[9] D. R. Karger. Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In *CIDR*, 2005, pp.13-26.

[10] J. Gemmell, G. Bell, R. Lueder, S.M. Drucker, C. Wong. MyLifeBits: fulfilling the Memex vision. ACM Multimedia 2002: 235-238.

[11] http://desktop.google.com/en/

[12] D. Elsweiler, M. Baillte, and I. Ruthven. Exploring Memory in Email Refinding. ACM Transactions on Information Systems, Vol. 26, No. 4, Article 21, Publication date: September 2008.

[13] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *ACM SIGIR*, 2003, pp.143-150.

[14] J.H. Lee, A. Renear and L.C. Smith. Known-Item search: Variations on a concept. In *ASIST*, 2006.

[15] C. Macdonald and I. Ounis. Combining fields in known-item email search. In *ACM SIGIR*, 2006, pp. 675-676.

[16] Y. Li, X.Meng: Research on Personal Dataspace Management, In *ACM SIGMOD PhD Workshop (IDAR)*, 2008, pp. 7-12.

[17] H.Zhuge, Communities and Emerging Semantics in Semantic Link Network: Discovery and Learning, IEEE Transactions on Knowledge and Data Engineering, vol.21, no.6, 2009, pp. 785-799.

[18] H. Zhuge. Resource space model, its design method and applications. The Journal of Systems and Software 72 (2004) 71-81.

[19] H.Zhuge, The Web Resource Space Model, Springer, 2008.

[20] H.Zhuge, Y.Xing and P.Shi, Resource Space Model, OWL and Database: Mapping and Integration, ACM Transactions on Internet Technology, 8/4, 2008.

[21] A. McCallum and B. Wellner. Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *NIPS*, 2004.

[22] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *ACM SIGKDD*, 2002, pp.350-359.

# ViDE: A Vision-Based Approach for Deep Web Data Extraction

Wei Liu, Xiaofeng Meng, *Member*, *IEEE*, and Weiyi Meng, *Member*, *IEEE*

**Abstract**—Deep Web contents are accessed by queries submitted to Web databases and the returned data records are enwrapped in dynamically generated Web pages (they will be called *deep Web pages* in this paper). Extracting structured data from deep Web pages is a challenging problem due to the underlying intricate structures of such pages. Until now, a large number of techniques have been proposed to address this problem, but all of them have inherent limitations because they are Web-page-programming-language-dependent. As the popular two-dimensional media, the contents on Web pages are always displayed regularly for users to browse. This motivates us to seek a different way for deep Web data extraction to overcome the limitations of previous works by utilizing some interesting common visual features on the deep Web pages. In this paper, a novel vision-based approach that is Web-page-programming-language-independent is proposed. This approach primarily utilizes the visual features on the deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction. We also propose a new evaluation measure *revision* to capture the amount of human effort needed to produce perfect extraction. Our experiments on a large set of Web databases show that the proposed vision-based approach is highly effective for deep Web data extraction.

**Index Terms**—Web mining, Web data extraction, visual features of deep Web pages, wrapper generation.

## 1 Introduction

The World Wide Web has more and more online Web databases which can be searched through their Web query interfaces. The number of Web databases has reached 25 millions according to a recent survey [21]. All the Web databases make up the deep Web (hidden Web or invisible Web). Often the retrieved information (query results) is enwrapped in Web pages in the form of data records. These special Web pages are generated dynamically and are hard to index by traditional crawler-based search engines, such as Google and Yahoo. In this paper, we call this kind of special Web pages *deep Web pages*. Each data record on the deep Web pages corresponds to an object. For instance, Fig. 1 shows a typical deep Web page from Amazon.com. On this page, the books are presented in the form of data records, and each data record contains some data items such as title, author, etc. In order to ease the consumption by human users, most Web databases display data records and data items regularly on Web browsers.

However, to make the data records and data items in them machine processable, which is needed in many applications such as deep Web crawling and metasearching, the structured data need to be extracted from the deep Web pages. In this paper, we study the problem of automatically extracting the structured data, including data records and data items, from the deep Web pages.

The problem of Web data extraction has received a lot of attention in recent years and most of the proposed solutions are based on analyzing the HTML source code or the tag trees of the Web pages (see Section 2 for a review of these works). These solutions have the following main limitations: First, they are Web-page-programming-language-dependent, or more precisely, HTML-dependent. As most Web pages are written in HTML, it is not surprising that all previous solutions are based on analyzing the HTML source code of Web pages. However, HTML itself is still evolving (from version 2.0 to the current version 4.01, and version 5.0 is being drafted [14]) and when new versions or new tags are introduced, the previous works will have to be amended repeatedly to adapt to new versions or new tags. Furthermore, HTML is no longer the exclusive Web page programming language, and other languages have been introduced, such as XHTML and XML (combined with XSLT and CSS). The previous solutions now face the following dilemma: should they be significantly revised or even abandoned? Or should other approaches be proposed to accommodate the new languages? Second, they are incapable of handling the ever-increasing complexity of HTML source code of Web pages. Most previous works have not considered the scripts, such as JavaScript and CSS, in the HTML files. In order to make Web pages vivid and colorful, Web page designers are using more and more complex JavaScript and CSS. Based on our observation from a large number of real Web pages, especially deep Web pages, the underlying structure of current Web pages is more complicated than ever and is far different from their layouts on Web browsers. This makes it more difficult for existing solutions to infer the regularity of the structure of Web pages by only analyzing the tag structures.

Meanwhile, to ease human users' consumption of the information retrieved from search engines, good template

- *W. Liu is with the School of Information, Renmin University of China, Beijing 100872, China. E-mail: gue1976@gmail.com*
- *X. Meng is with the School of Information, Renmin University of China, Beijing 100872, China. E-mail: xfmeng@ruc.edu.cn.*
- *W. Meng is with the Department of Computer Science, Watson School of Engineering, Binghamton University, Binghamton, NY 13902. E-mail: meng@cs.binghamton.edu.*

*Manuscript received 30 Dec. 2007; revised 12 Aug. 2008; accepted 16 Feb. 2009; published online 17 Apr. 2009.*
*Recommended for acceptance by V. Ganti.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-12-0629.*
*Digital Object Identifier no. 10.1109/TKDE.2009.109.*

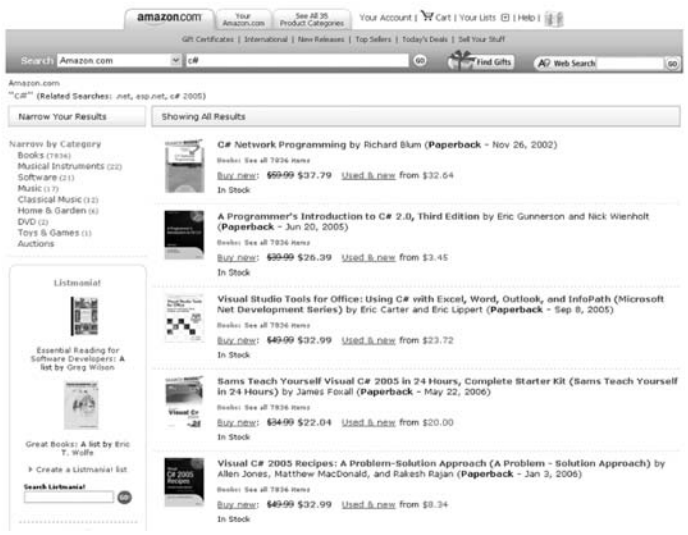



Fig. 1. An example deep Web page from Amazon.com.

designers of deep Web pages always arrange the data records and the data items with visual regularity to meet the reading habits of human beings. For example, all the data records in Fig. 1 are clearly separated, and the data items of the same semantic in different data records are similar on layout and font.

In this paper, we explore the visual regularity of the data records and data items on deep Web pages and propose a novel vision-based approach, Vision-based Data Extractor (ViDE), to extract structured results from deep Web pages automatically. ViDE is primarily based on the visual features human users can capture on the deep Web pages while also utilizing some simple nonvisual information such as data types and frequent symbols to make the solution more robust. ViDE consists of two main components, Vision-based Data Record extractor (ViDRE) and Vision-based Data Item extractor (ViDIE). By using visual features for data extraction, ViDE avoids the limitations of those solutions that need to analyze complex Web page source files.

Our approach employs a four-step strategy. First, given a sample deep Web page from a Web database, obtain its visual representation and transform it into a Visual Block tree which will be introduced later; second, extract data records from the Visual Block tree; third, partition extracted data records into data items and align the data items of the same semantic together; and fourth, generate visual wrappers (a set of visual extraction rules) for the Web database based on sample deep Web pages such that both data record extraction and data item extraction for new deep Web pages that are from the same Web database can be carried out more efficiently using the visual wrappers.

To our best knowledge, although there are already some works [3], [4], [23], [26], [28] that pay attention to the visual information on Web pages, our work is the first to perform deep Web data extraction using primarily visual features. Our approach is independent of any specific Web page programming language. Although our current implementation uses the VIPS algorithm [4] to obtain a deep Web page's Visual Block tree and VIPS needs to analyze the HTML source code of the page, our solution is independent of any specific method used to obtain the Visual Block tree in the sense that any tool that can segment the Web pages into a tree structure based on the visual information, not HTML source code, can be used to replace VIPS in the implementation of ViDE.

In this paper, we also propose a new measure, *revision*, to evaluate the performance of Web data extraction tools. It is the percentage of the Web databases whose data records or data items cannot be perfectly extracted (i.e., at least one of the precision and recall is not 100 percent). For these Web databases, manual revision of the extraction rules is needed to achieve perfect extraction.

In summary, this paper has the following contributions: 1) A novel technique is proposed to perform data extraction from deep Web pages using primarily visual features. We open a promising research direction where the visual features are utilized to extract deep Web data automatically. 2) A new performance measure, *revision*, is proposed to evaluate Web data extraction tools. This measure reflects how likely a tool will fail to generate a perfect wrapper for a site. 3) A large data set consisting of 1,000 Web databases across 42 domains is used in our experimental study. In contrast, the data sets used in previous works seldom had more than 100 Web databases. Our experimental results indicate that our approach is very effective.

The rest of the paper is organized as follows: Related works are reviewed in Section 2. Visual representation of deep Web pages and visual features on deep Web pages are introduced in Section 3. Our solutions to data record extraction and data item extraction are described in Sections 4 and 5, respectively. Wrapper generation is discussed in Section 6. Experimental results are reported in Section 7. Finally, concluding remarks are given in Section 8.

## 2 Related Work

A number of approaches have been reported in the literature for extracting information from Web pages. Good surveys about previous works on Web data extraction can be found in [16] and [5]. In this section, we briefly review previous works based on the degree of automation in Web data extraction, and compare our approach with fully automated solutions since our approach belongs to this category.

### 2.1 Manual Approaches

The earliest approaches are the manual approaches in which languages were designed to assist programmer in constructing wrappers to identify and extract all the desired data items/fields. Some of the best known tools that adopt manual approaches are Minerva [7], TSIMMIS [11], and Web-OQL [1]. Obviously, they have low efficiency and are not scalable.

### 2.2 Semiautomatic Approaches

Semiautomatic techniques can be classified into sequence-based and tree-based. The former, such as WIEN [15], Soft-Mealy [12], and Stalker [22], represents documents as sequences of tokens or characters, and generates delimiter-based extraction rules through a set of training examples. The latter, such as W4F [24] and XWrap [19], parses the document into a hierarchical tree (DOM tree), based on which they perform the extraction process. These approaches require manual efforts, for example, labeling some sample pages, which is labor-intensive and time-consuming.

### 2.3 Automatic Approaches

In order to improve the efficiency and reduce manual efforts, most recent researches focus on automatic approaches instead of manual or semiautomatic ones. Some representative automatic approaches are Omini [2], RoadRunner [8], IEPAD [6], MDR [17], DEPTA [29], and the method in [9]. Some of these approaches perform only data record extraction but not data item extraction, such as Omini and the method in [9]. RoadRunner, IEPAD, MDR, DEPTA, Omini, and the method in [9] do not generate wrappers, i.e., they identify patterns and perform extraction for each Web page directly without using previously derived extraction rules. The techniques of these works have been discussed and compared in [5], and we do not discuss them any further here. Note that all of them mainly depend on analyzing the source code of Web pages. As a result, they cannot avoid the inherent limitations described in Section 1. In addition, there are several works (DeLa [27], DEPTA, and the method in [20]) on data item extraction, which is a preparation step for holistic data annotation, i.e., assigning meaningful labels to data items. DeLa utilizes HTML tag information to construct regular expression wrapper and extract data items into a table. Similar to DeLa, DEPTA also operates on HTML tag tree structures to first align data items in a pair of data records that can be matched with certainty. The remaining data items are then incrementally added. However, both data alignment techniques are mainly based on HTML tag tree structures, not visual information. The automatic data alignment method in [20] proposes a clustering approach to perform alignment based on five features of data items, including font of text. However, this approach is primarily text-based and tag-structure-based, while our method is primarily visual-information-based.

The only works that we are aware of that utilize some visual information to extract Web data are ViNTS [30], ViPERS [25], HCRF [32], and VENTex [10]. ViNTs use the visual content features on the query result pages to capture content regularities denoted as Content Lines, and then, utilize the HTML tag structures to combine them. ViPER also incorporates visual information on a Web page for data records extraction with the help of a global multiple sequence alignment technique. However, in the two approaches, tag structures are still the primary information utilized, while visual information plays a small role. In addition, both of them only focus on data record extraction, without considering data item extraction. HCRF is a probabilistic model for both data record extraction and attribute labeling. Compared to our solution, it also uses VIPS algorithm [4] to represent Web pages, but the tag information is still an important feature in HCRF. And furthermore, it is implemented under an ideal assumption that every record corresponds to one block in the Visual Block tree, but this assumption is not always correct according to our observation to the real Web pages (about 20 percent of deep Web pages do not meet this assumption). VENTex implements the information extraction from Web tables based on a variation of the CSS2 visual box model. So, it can be regarded as the only related work using pure visual features. The main difference between our approach and VENTex is their objectives. VENTex aims to extract various forms of tables that are embedded in common pages, whereas our approach focuses on extracting regularly arranged data records and data items from deep Web pages.

TABLE 1
Font Attributes and Examples

| Font factor | Example | Font factor | Example |
|---|---|---|---|
| Size | A (10pt) | underline | A |
| face | A(Sans Serif) | italic | *A* |
| color | A (red) | weight | **A** |
| strikethrough | ~~A~~ | frame | A |

## 3 Visual Block Tree and Visual Features

Before the main techniques of our approach are presented, we describe the basic concepts and visual features that our approach needs.

### 3.1 Visual Information of Web Pages

The information on Web pages consists of both texts and images (static pictures, flash, video, etc.). The visual information of Web pages used in this paper includes mostly information related to *Web page layout* (location and size) and *font*.

#### 3.1.1 Web Page Layout

A coordinate system can be built for every Web page. The origin locates at the top left corner of the Web page. The X-axis is horizontal left-right, and the Y-axis is vertical top-down. Suppose each text/image is contained in a minimum bounding rectangle with sides parallel to the axes. Then, a text/image can have an exact coordinate ($x$, $y$) on the Web page. Here, $x$ refers to the horizontal distance between the origin and the left side of its corresponding rectangle, while $y$ refers to the vertical distance between the origin and the upper side of its corresponding box. The size of a text/image is its height and width.

The coordinates and sizes of texts/images on the Web page make up the Web page layout.

#### 3.1.2 Font

The fonts of the texts on a Web page are also very useful visual information, which are determined by many attributes as shown in Table 1. Two fonts are considered to be the same only if they have the same value under each attribute.

### 3.2 Deep Web Page Representation

The visual information of Web pages, which has been introduced above, can be obtained through the programming interface provided by Web browsers (i.e., IE). In this paper, we employ the VIPS algorithm [4] to transform a deep Web page into a Visual Block tree and extract the visual information. A Visual Block tree is actually a segmentation of a Web page. The root block represents the whole page, and each block in the tree corresponds to a rectangular region on

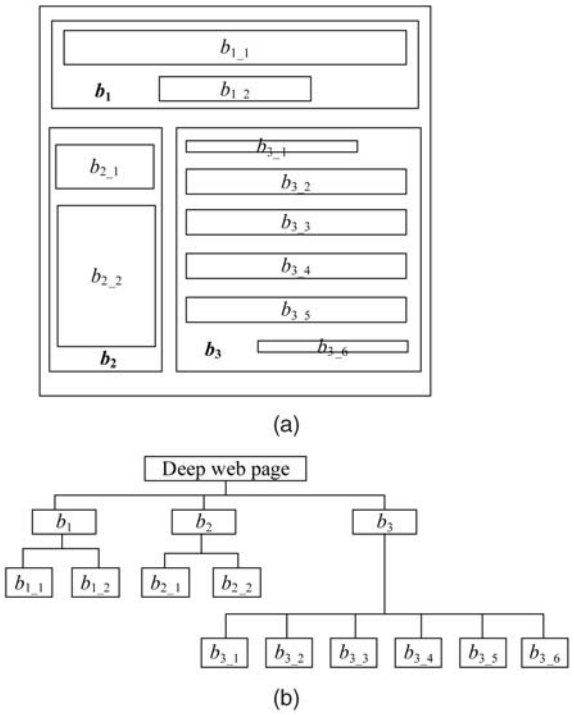



Fig. 2. (a) The presentation structure and (b) its Visual Block tree.

the Web page. The leaf blocks are the blocks that cannot be segmented further, and they represent the minimum semantic units, such as continuous texts or images. Fig. 2a shows a popular presentation structure of deep Web pages and Fig. 2b gives its corresponding Visual Block tree. The technical details of building Visual Block trees can be found in [4]. An actual Visual Block tree of a deep Web page may contain hundreds even thousands of blocks.

Visual Block tree has three interesting properties. First, block $a$ contains block $b$ if $a$ is an ancestor of $b$. Second, $a$ and $b$ do not overlap if they do not satisfy property one. Third, the blocks with the same parent are arranged in the tree according to the order of the corresponding nodes appearing on the page. These three properties are illustrated by the example in Fig. 2. The formal representations for internal blocks and leaf blocks in our approach are given below. Each internal block $a$ is represented as $a = (CS, P, S, FS, IS)$, where $CS$ is the set containing its child blocks (note that the order of blocks is also kept), $P$ is the position of $a$ (its coordinates on the Web page), $S$ is its size (height and width), $FS$ is the set of the fonts appearing in a, and $IS$ is the number of images in $a$. Each leaf block $b$ is represented as $\mathrm{b} = (P, S, F, L, I, C)$, where the meanings of $P$ and $S$ are the same as those of an inner block, $F$ is the font it uses, $L$ denotes whether it is a hyperlink text, $I$ denotes whether it is an image, and $C$ is its content if it is a text.

### 3.3 Visual Features of Deep Web Pages

Web pages are used to publish information to users, similar to other kinds of media, such as newspaper and TV. The designers often associate different types of information with distinct visual characteristics (such as font, position, etc.) to make the information on Web pages easy to understand. As a result, visual features are important for identifying special information on Web pages. Deep Web pages are special Web pages that contain data records retrieved from Web databases, and we hypothesize that there are some distinct visual features for data records and data items. Our observation based on a large number of deep Web pages is consistent with this hypothesis. We describe the main visual features in this section and show the statistics about the accuracy of these features at the end of this Section 3.3.

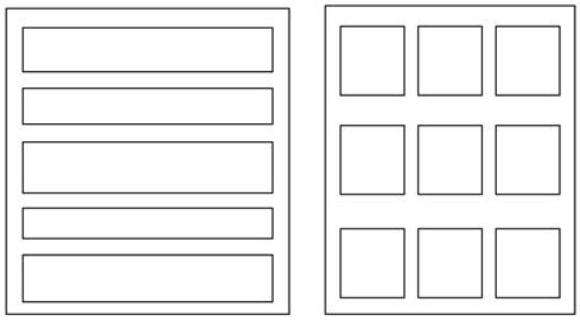


Fig. 3. Layout models of data records on deep Web pages.

**Position features ($PF$s).** These features indicate the location of the data region on a deep Web page.

- $PF1$: Data regions are always centered horizontally.
- $PF2$: The size of the data region is usually large relative to the area size of the whole page.

Since the data records are the contents in focus on deep Web pages, Web page designers always have the region containing the data records centrally and conspicuously placed on pages to capture the user's attention. By investigating a large number of deep Web pages, we found two interesting facts. First, data regions are always located in the middle section horizontally on deep Web pages. Second, the size of a data region is usually large when there are enough data records in the data region. The actual size of a data region may change greatly because it is not only influenced by the number of data records retrieved, but also by what information is included in each data record. Therefore, our approach uses the ratio of the size of the data region to the size of whole deep Web page instead of the actual size. In our experiments in Section 7, the threshold of the ratio is set at 0.4, that is, if the ratio of the horizontally centered region is greater than or equal to 0.4, then the region is recognized as the data region.

**Layout features ($LF$s).** These features indicate how the data records in the data region are typically arranged.

- $LF1$: The data records are usually aligned flush left in the data region.
- $LF2$: All data records are adjoining.
- $LF3$: Adjoining data records do not overlap, and the space between any two adjoining records is the same.

Data records are usually presented in one of the two layout models shown in Fig. 3. In Model 1, the data records are arranged in a single column evenly, though they may be different in width and height. LF1 implies that the data records have the same distance to the left boundary of the data region. In Model 2, data records are arranged in

TABLE 2
Relevant Visual Information about the Top Five Data Records in Fig. 1

| | Images (pixel) | plain texts | | link texts | |
|---|---|---|---|---|---|
| | | Total font number | Shared font number | Total font number | Shared font number |
| record1 | 115*115 | 5 | 5 | 2 | 2 |
| record2 | 115*115 | 5 | 5 | 2 | 2 |
| record3 | 115*110 | 5 | 5 | 2 | 2 |
| record4 | 115*115 | 5 | 5 | 2 | 2 |
| record5 | 115*115 | 5 | 5 | 2 | 2 |

multiple columns, and the data records in the same column have the same distance to the left boundary of the data region. Because most deep Web pages follow the first model, we only focus on the first model in this paper, and the second model can be addressed with minor implementation expansion to our current approach. In addition, data records do not overlap, which means that the regions of different data records can be separated.

**Appearance features ($AF$s).** These features capture the visual features within data records.

- $AF1$: Data records are very similar in their appearances, and the similarity includes the sizes of the images they contain and the fonts they use.
- $AF2$: The data items of the same semantic in different data records have similar presentations with respect to position, size (image data item), and font (text data item).
- $AF3$: The neighboring text data items of different semantics often (not always) use distinguishable fonts.

$AF1$ describes the visual similarity at the data record level. Generally, there are three types of data contents in data records, i.e., images, plain texts (the texts without hyperlinks), and link texts (the texts with hyperlinks). Table 2 shows the information on the three aspects for the data records in Fig. 1. We can see that these five data records are very close on the three aspects. $AF2$ and $AF3$ describe the visual similarity at the data item level. The text data items of the same semantic always use the same font, and the image data items of the same semantic are often similar in size. The positions of data items in their respective data records can be classified into two kinds: *absolute position* and *relative position*. The former means that the positions of the data items of certain semantic are fixed in the line they belong to, while the latter refers to the position of a data item relative to the data item ahead of it. Furthermore, the items of the same semantic from different data records share the same kind of position. $AF3$ indicates that the neighboring text data items of different semantics often use distinguishable fonts. However, $AF3$ is not a robust feature because some neighboring data items may use the same font. Neighboring data items with the same font are treated as a *composite data item*. Composite data items have very simple string patterns and the real data items in them can often be separated by a limited number of symbols, such as ",", "/," etc. In addition,


Fig. 4. Illustrating visual features of deep Web pages.

the composite data items of the same semantics share the same string pattern. Hence, it's easy to break composite data items into real data items using some predefined separating symbols. For example, in Fig. 4, four data items, such as publisher, publishing date, edition, and ISBN, form a composite data item, and they are separated by commas. According to our observation to deep Web pages, the granularity of the data items extracted is not larger than what HTML tags can separate, because a composite data item is always included in one leaf node in the tag tree.

**Content feature ($CF$).** These features hint the regularity of the contents in data records.

- $CF1$: The first data item in each data record is always of a mandatory type.
- $CF2$: The presentation of data items in data records follows a fixed order.
- $CF3$: There are often some fixed static texts in data records, which are not from the underlying Web database.

The data records correspond to the entities in real world, and they consist of data items with different semantics that describe the attribute values of the entities. The data items can be classified into two kinds: *mandatory* and *optional*. Mandatory data items appear in all data records. For example, if every data record must have a title, then titles are mandatory data items. In contrast, optional items may be missing in some data records. For example, "discounted price" for products is likely an optional unit. The order of different types of data items from the same Web database is always fixed in data records. For example, the order of attributes of data records from Bookpool.com in Fig. 4 is "title," "author," "publisher," "publish time," "edition," "ISBN," "discount price," "save money," "availability," etc. Fixed static texts refer to the texts that appear in every data record. Most of them are meaningful labels that can help users understand the semantics of data items, such as "Buy new" in Fig. 4. We call these static texts static items, which are part of the record template.

Our deep Web data extraction solution is developed mainly based on the above four types of visual features. $PF$ is used to locate the region containing all the data records on a deep Web page; $LF$ and $AF$ are combined together to extract the data records and data items.

**Statistics on the visual features.** To verify the robustness of these visual features we observed, we examined these features on 1,000 deep Web pages of different Web databases from the General Data Set (GDS) used in our

TABLE 3
The Statistics on the Visual Features

| Feature type | | Statistics | Feature type | | Statistics |
|---|---|---|---|---|---|
| Position Features | *PF1* | 99.9% | Appearance Features | *AF1* | 99.5% |
| | *PF2* | 99.9% | | *AF2* | 100% |
| | | | | *AF3* | 92.8% |
| Layout Features | *LF1* | 99.3% | Content Features | *CF1* | 100% |
| | *LF2* | 100% | | *CF2* | 100% |
| | *LF3* | 100% | | *CF3* | 6.5% |

experiments (see Section 7 for more information about GDS). The results are shown in Table 3. For most features (except $AF3$ and $CF3$), their corresponding statistics are the percentages of the deep Web pages that satisfy them. For example, the statistics of 99.9 percent for $PF1$ means that for 99.9 percent of the deep Web pages, $PF1$ feature "data regions are always centered horizontally" is true. From the statistics, we can conclude that these visual features are very robust and can be reliably applied to general deep Web pages. For $AF3$, 92.8 percent is the percentage of the data items that have different font from their following data items. For $CF3$, 6.5 percent is the percentage of the static data items over all data items.

We should point out that when a feature is not satisfied by a page, it does not mean that ViDE will fail to process this page. For example, our experiments using the data sets to be described in Section 7 show that among the pages that violate $LF3$, 71.4 percent can still be processed successfully by ViDE, and among the pages that violate $AF1$, 80 percent can still be correctly processed.

### 3.4 Special Supplementary Information

Several types of simple nonvisual information are also used in our approach in this paper. They are *same text*, *frequent symbol*, and *data type*, as explained in Table 4.

Obviously, the above information is very useful to determine whether the data items in different data records from the same Web database belong to the same semantic. The above information can be captured easily from the Web pages using some simple heuristic rules without the need to analyze the HTML source code or the tag trees of the Web pages. Furthermore, they are specific language (i.e., English, French, etc.) independent.

TABLE 4
Nonvisual Information Used

| Special complementary information | Remarks |
|---|---|
| *Same text* | Given two texts, we can determine whether or not they are the same. |
| *Frequent symbol* | Given the deep web pages of a web database, if some symbols/words (e.g., ISBN, $) appear in all the data items of an attribute, they are called frequent symbols. |
| *Data type* | They are predefined, including image, text, number, date, price, email, etc |

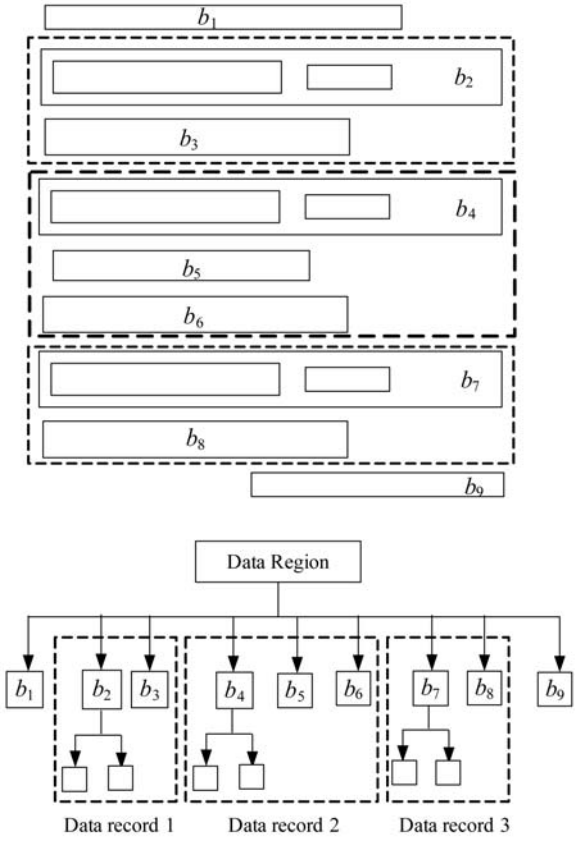



Fig. 5. A general case of data region.

## 4 Data Records Extraction

Data record extraction aims to discover the boundary of data records and extract them from the deep Web pages. An ideal record extractor should achieve the following: 1) all data records in the data region are extracted and 2) for each extracted data record, no data item is missed and no incorrect data item is included.

Instead of extracting data records from the deep Web page directly, we first locate the data region, and then, extract data records from the data region. $PF1$ and $PF2$ indicate that the data records are the primary content on the deep Web pages and the data region is centrally located on these pages. The data region corresponds to a block in the Visual Block tree. We locate the data region by finding the block that satisfies the two position features. Each feature can be considered as a rule or a requirement. The first rule can be applied directly, while the second rule can be represented by $(area_b/area_{page}) > T_{region}$, where $area_b$ is the area of block $b$, $area_{page}$ is the area of the whole deep Web page, and $T_{region}$ is a threshold. The threshold is trained from sample deep Web pages. If more than one block satisfies both rules, we select the block with the smallest area. Though very simple, this method can find the data region in the Visual Block tree accurately and efficiently.

Each data record corresponds to one or more subtrees in the Visual Block tree, which are just the child blocks of the data region, as Fig. 5 shows. So, we only need to focus on the child blocks of the data region. In order to extract data

records from the data region accurately, two facts must be considered. First, there may be blocks that do not belong to any data record, such as the statistical information (e.g., about 2,038 matching results for java) and annotation about data records (e.g., 1, 2, 3, 4, 5 (Next)). These blocks are called *noise blocks* in this paper. Noise blocks may appear in the data region because they are often close to the data records. According to $LF2$, noise blocks cannot appear between data records. They always appear at the top or the bottom of the data region. Second, one data record may correspond to one or more blocks in the Visual Block tree, and the total number of blocks in which one data record contains is not fixed. In Fig. 5, block $b_1$ (statistical information) and $b_9$ (annotation) are noise blocks; there are three data records ($b_2$ and $b_3$ form data record 1; $b_4$, $b_5$, and $b_6$ form data record 2; $b_7$ and $b_8$ form data record 3), and the dashed boxes are the boundaries of data records.

Data record extraction is to discover the boundary of data records based on the $LF$ and $AF$ features. That is, we attempt to determine which blocks belong to the same data record. We achieve this in the following three phases:

1. Phase 1: Filter out some noise blocks.
2. Phase 2: Cluster the remaining blocks by computing their appearance similarity.
3. Phase 3: Discover data record boundary by regrouping blocks.

### 4.1 Phase 1: Noise Blocks Filtering

Because noise blocks are always at the top or bottom, we check the blocks located at the two positions according to $LF1$. If a block at these positions is not aligned flush left, it will be removed as a noise block. This step does not guarantee the removal of all noise blocks. For example, in Fig. 5, block $b_9$ can be removed in this step, while block $b_1$ cannot be removed.

### 4.2 Phase 2: Blocks Clustering

The remaining blocks in the data region are clustered based on their appearance similarity. Since there may be three kinds of information in data records, i.e., images, plain text, and link text, the appearance similarity between blocks is computed from the three aspects. For images, we care about the size; for plain text and link text, we care about the shared fonts. Intuitively, if two blocks are more similar on image size and font, they should be more similar in appearance. The formula for computing the appearance similarity between two blocks $b_1$ and $b_2$ is given below:

$$sim(b_1, b_2) = w_i * simIMG(b_1, b_2) + w_{pt} * simPT(b_1, b_2) + w_{lt} * simLT(b_1, b_2), \quad (1)$$

where $simIMG(b_1, b_2)$, $simIMG(b_1, b_2)$, and $simLT(b_1, b_2)$ are the similarities based on image size, plain text font, and link text font, respectively. And $w_i$, $w_{pt}$, and $w_{lt}$ are the weights of these similarities, respectively. Table 5 gives the formulas to compute the component similarities and the weights in different cases. The weight of one type of contents is proportional to their total size relative to the total size of the two blocks.

A simple one-pass clustering algorithm is applied. First, starting from an arbitrary order of all the input blocks, take the first block from the list and use it to form a cluster. Next, for each of the remaining blocks, say $b$, compute its similarity with each existing cluster. Let $C$ be the cluster that has the largest similarity with $A$. If $sim(b, C) > T_{as}$ for some threshold $T_{as}$, which is to be trained by sample pages (generally, $T_{as}$ is set to 0.8), then add $b$ to $C$; otherwise, form a new cluster based on $b$. Function $sim(b, C)$ is defined to be the average of the similarities between $b$ and all blocks in $C$ computed using (1). As an example, by applying this method to the blocks in Fig. 1, the blocks containing the titles of the data records are clustered together, so are the blocks containing the prices and so on.

TABLE 5
Formulas and Remarks

| formulas | remarks |
|---|---|
| $simIMG(b_1,b_2)=\frac{Min\{sa_i(b_1),sa_i(b_2)\}}{Max\{sa_i(b_1),sa_i(b_2)\}}$ | $sa_i(b)$ is the total area of images in block $b$. $sa_b(b)$ is the total area of block $b$. $fn_{pt}(b)$ is the total number of fonts of the plain texts in block $b$. $sa_{pt}(b)$ is the total area of the plain texts in block $b$. $fn_{lt}(b)$ is the total number of fonts of the link texts in block $b$. $sa_{lt}(b)$ is the total area of the link texts in block $b$. |
| $w_i=\frac{sa_i(b_1)+sa_i(b_2)}{sa_b(b_1)+sa_b(b_2)}$ | |
| $simPT(b_1,b_2)=\frac{Min\{fn_{pt}(b_1),fn_{pt}(b_2)\}}{Max\{fn_{pt}(b_1),fn_{pt}(b_2)\}}$ | |
| $w_{pt}=\frac{sa_{pt}(b_1)+sa_{pt}(b_2)}{sa_b(b_1)+sa_b(b_2)}$ | |
| $simLT(b_1,b_2)=\frac{Min\{fn_{lt}(b_1),fn_{lt}(b_2)\}}{Max\{fn_{lti}(b_1),fn_{lt}(b_2)\}}$ | |
| $w_{lt}=\frac{sa_{lt}(b_1)+sa_{lt}(b_2)}{sa_b(b_1)+sa_b(b_2)}$ | |

### 4.3 Phase 3: Blocks Regrouping

The clusters obtained in the previous step do not correspond to data records. On the contrary, the blocks in the same cluster all come from different data records. According to $AF2$, the blocks in the same cluster have the same type of contents of the data records.

The blocks need to be regrouped such that the blocks belonging to the same data record form a group. Our basic idea of blocks regrouping is as follows: According to $CF1$, the first data item in each data record is always mandatory. Clearly, the cluster that contains the blocks for the first items has the maximum number of blocks possible; let $n$ be this maximum number. It is easy to see that if a cluster contains n blocks, these blocks contain mandatory data items. Our regrouping method first selects a cluster with $n$ blocks and uses these blocks as seeds to form data records. Next, given a block $b$, we determine which record $b$ belongs to according to $CF2$. For example, suppose we know that title is ahead of author in each record and they belong to different blocks (say an author block and a title block). Each author block should relate to the nearest title block that is ahead of it. In order to determine the order of different semantic blocks, a minimum bounding rectangle is

**Algorithm block regrouping**
Input: $C_1,C_2,\ldots,C_m$: a group of clusters generated by blocks clustering from a given sample deep web page $P$
Output: $G_1,G_2,\ldots,G_n$: each of them corresponds to a data record on $P$
Begin
//Step 1. sort the blocks in $C_i$ according to their positions in the page (from top to bottom and then from left to right)
1 for each cluster $C_i$ do
2 for any two blocks $b_{i,j}$ and $b_{i,k}$ in $C_i$ //$1 \le j < k \le |C_i|$
3 if $b_{i,j}$ and $b_{i,k}$ are in different lines on $P$, and $b_{i,k}$ is above $b_{i,j}$
4 $b_{i,j} \leftrightarrow b_{i,k}$; //exchange their orders in $C_i$;
5 else if $b_{i,j}$ and $b_{i,k}$ are in the same line on $P$, and $b_{i,k}$ is in front of $b_{i,j}$
6 $b_{i,j} \leftrightarrow b_{i,k}$;
7 end until no exchange occurs;
8 form the minimum-bounding rectangle $Rec_i$ for $C_i$;
//Step 2. initialize $n$ groups, and $n$ is the number of data records on $P$
9 $C_{max}=\{C_i \mid |C_i|=\max\{|C_1|,|C_2|,\ldots\ldots,|C_m|\}\}$; // n=$|C_{max}|$
10 for each block $b_{max,i}$ in $C_{max}$
11 Initialize group $G_i$;
12 put $b_{max,i}$ into $G_i$;
//Step 3. put the blocks into the right groups, and each group corresponds to a data record
13 for each cluster $C_i$
14 if $Rec_i$ overlaps with $Rec_{max}$ on $P$
15 if $Rec_i$ is ahead of (behind) $Rec_{max}$
16 for each block $b_{i,j}$ in $C_i$
17 find the nearest block $b_{max,k}$ in $C_{max}$ that is behind (ahead of) $b_{i,j}$ on the web page;
18 place $b_{i,j}$ into group $G_k$;
End

Fig. 6. The algorithm of blocks regrouping.

formed for each cluster on the page. By comparing the positions of these rectangles on the page, we can infer the order of the semantics. For example, if the rectangle enclosing all title blocks is higher than the rectangle enclosing the author blocks, then title must be ahead of its corresponding author. Based on this idea, the algorithm of block regrouping is developed as shown in Fig. 6.

This algorithm consists of three steps. Step 1 rearranges the blocks in each cluster based on their appearance order on the Web page, i.e., from left to right and from top to bottom (lines 1-7). In addition, a minimum bounding rectangle is formed for each cluster on the page (line 8). In Step 2, $n$ groups are initialized with a seed block in each group as discussed earlier, where n is the number of blocks in a maximum cluster, denoted as $C_{max}$. According to $CF1$, we always choose the cluster that contains the first mandatory data item of each record as $C_{max}$. Let $b_{max,k}$ denote the seed block in each initial group Gk. Step 3 determines to which group each block belongs. If block $b_{i,j}$ (in $C_i$, $C_i$ is not $C_{max}$) and block $b_{max,k}$ (in $C_{max}$) are in the same data record, then $b_{i,j}$ should be put into the same group $b_{max,k}$ belongs to. According to $LF3$, no two adjoining data records overlap. So, for $b_{max,k}$ in $C_{max}$, the blocks that belong to the same data record with $b_{max,k}$ must be below $b_{max,k-1}$ and above $b_{max,k+1}$. For each $C_i$, if data record $R_i$ is ahead of $R_{max}$, then the block on top of $R_i$ is ahead of (behind) the block on top of $R_{max}$. Here, "ahead of" means "on the left of" or "above," and "behind" means "on the right of" or "below." According to $CF2$, $b_{i,j}$ is ahead of

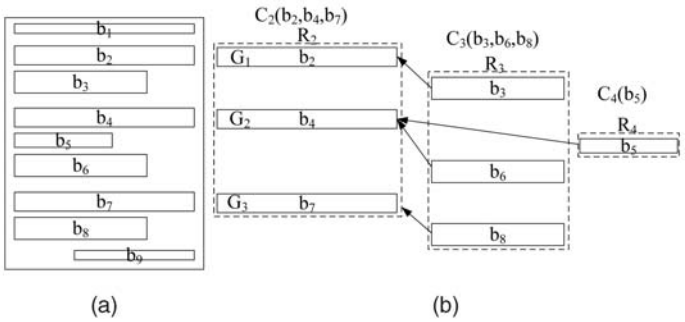



Fig. 7. An illustration of data record extraction.

$b_{max,k}$ if they belong to the same data record. So, we can conclude that if $b_{max,k}$ is the nearest block behind $b_{i,j}$, then $b_{i,j}$ should be put into the group $b_{max,k}$ belongs to. Obviously, the complexity of this algorithm is $O(n^2)$, where $n$ is the number of data records in the sample page.

**Example for data record extraction.** Fig. 7 illustrates the case in Fig. 5. First, $b_9$ is removed according to $LF1$. Then, the blocks on the left in Fig. 7b are clustered using (1). Altogether, four clusters are formed and the blocks in them are also rearranged: $C_1\{b_1\}$, $C_2\{b_2, b_4, b_7\}$, $C_3\{b_3, b_6, b_8\}$, and $C_4\{b_5\}$. Next, $C_2$ is $C_{max}$, and $b_2$, $b_4$, and $b_7$ form three initial groups $G_1, G_2$, and $G_3$, respectively. Since $R_3$ and $R_4$ overlap with $R_2$ and $R_3$ is below $R_2$, we group $b_3$, $b_6$, and $b_8$ with $b_2$, $b_4$, and $b_7$ (the nearest block above it in $C_2$), respectively. At last, $G_1$ is $\{b_2, b_3\}$, $G_2$ is $\{b_4, b_5, b_6\}$, and $G_3$ is $\{b_7, b_8\}$. Each group forms a complete data record.

## 5 Data Item Extraction

A data record can be regarded as the description of its corresponding object, which consists of a group of data items and some static template texts. In real applications, these extracted structured data records are stored (often in relational tables) at data item level and the data items of the same semantic must be placed under the same column. When introducing $CF$, we mentioned that there are three types of data items in data records: mandatory data items, optional data items, and static data items. We extract all three types of data items. Note that static data items are often annotations to data and are useful for future applications, such as Web data annotation. Below, we focus on the problems of segmenting the data records into a sequence of data items and aligning the data items of the same semantics together.

Note that data item extraction is different from data record extraction; the former focuses on the leaf nodes of the Visual Block tree, while the latter focuses on the child blocks of the data region in the Visual Block tree.

### 5.1 Data Record Segmentation

$AF3$ indicates that composite data items cannot be segmented any more in the Visual Block tree. So, given a data record, we can collect its leaf nodes in the Visual Block tree in left to right order to carry out data record segmentation. Each composite data item also corresponds to a leaf node. We can treat it as a regular data item initially, and then, segment it into the real data items with the heuristic rules mentioned in $AF3$ after the initial data item alignment.

```
Algorithm data item matching
Input: item1, item2: two data items
Output: matched or unmatched: the match result (Boolean)
Begin
1 if (font(item1) ≠ font(item2))
2     Return unmatched;
3 if (position(item1) = position(item2))
4     return matched;
5 if (itemp1 and itemp2 are matched)     // itemp1 and itemp2 are the data
items immediately in front of item1 and item2 respectively
6         return matched;
7 else
      return unmatched;
End
```

Fig. 8. The algorithm of data item matching.

## 5.2 Data Item Alignment

$CF1$ indicates that we cannot align data items directly due to the existence of optional data items. It is natural for data records to miss some data items in some domains. For example, some books have discount price, while some do not.

Every data record has been turned into a sequence of data items through data record segmentation. Data item alignment focuses on the problem of how to align the data items of the same semantic together and also keep the order of the data items in each data record. In the following, we first define visual matching of data items, and then, propose an algorithm for data item alignment.

### 5.2.1 Visual Matching of Data Items

$AF2$ indicates that if two data items from different data records belong to the same semantic, they must have consistent font and position, including both absolute position and relative position. In Fig. 8, a simple algorithm to match two visually similar data items from different data records is described.

The first four lines of the algorithm say that two data items are matched only if they have the same absolute position in addition to having the same font. Here, absolute position is the distance between the left side of the data region and the left side of a data item. When two data items do not have the same absolute position, they can still be matched if they have the same relative position. For match on relative position, the data items immediately before the two input data items should be matched (from line 5 to line 6). As an example, for the two records in Fig. 4, the titles can be matched based on the absolute positions and the authors on the relative positions.

Because two data items of different semantics can also be visually similar, $AF2$ cannot really determine whether two data items belong to the same semantic. But the fixed order of the data items in the same data record ($CF2$) can help us remedy this limitation. So, we further propose an effective algorithm for data item alignment that utilizes both $CF2$ and $AF2$.

### 5.2.2 Algorithm for Data Item Alignment

$CF2$ says that the order of data items in data records is fixed. Thus, each data record can be treated as a sequence of data items. We can utilize this feature to align data items. Our goal is to place the data items of the same semantic in the same column. If an optional data item does not appear in a data record, we will fill the vacant position with a predefined blank item. Based on this insight, we propose a multialignment algorithm that can process all extracted data records holistically step by step. The basic idea of this algorithm is described as follows: Initially, all the data items are unaligned. We align data items by the order in their corresponding data records. When we encounter optional data items that do not appear in some data records, these vacant positions will be filled with the predefined blank item. This ensures that all data records are aligned and have the same number of data items at the end. Our data item alignment algorithm is shown in Fig. 9.

```
Algorithm data item alignment
Input: a set of extracted data records {ri | 1≤i≤n}
Output: a set of data records {ri | 1≤i≤n} with all the data items aligned
Begin
1  currentItemSet=ϕ;
2  currentCluster=ϕ;
//put the first unaligned data item of each ri into currentItemSet:
// Itemi^U(i) refers to the first unaligned item of the ith data record
3   currentItemSet ↔ Itemi^U(i) (1≤i≤n);
4   while currentItemSet≠ϕ
5       use the data item matching algorithm to group the data items
in currentItemSet into k clusters {Ci | 1≤i≤k} (k≤n);
6       for each cluster Ci
7         for each rj that does not have a data item in Ci
8           if Itemj^U(j)+k is matched with data items in Ci
9                   Log position k;
10          else
11                  Log position 0;
12  Pi = max value of these logged positions for Ci;
      /*Till now, each cluster Ci has a position Pi */
13    if any PL==0
14      currentCluster=CL;
15    else
16      currentCluster=CL whose PL is max {P1, P2,...,PK};
17    for each rj whose Itemj^U(j) is in currentCluster CL
18      remove Itemj^U(j) from currentItemSet;
19      if Itemj^U(j)+1 exists in rj
20              put Itemj^U(j)+1 into currentItemSet;
21    for each rj that has no item in currentCluster CL
22      insert a blank item ahead of Itemj^U(j) in rj ;
23    U(j)++;
End
```

Fig. 9. The algorithm of data item alignment.

The input is n data records $\{r_1, r_2, \ldots, r_n\}$, and each data record $r_i$ is denoted as a sequence of data items $\{item_i^1, item_i^2, \ldots, item_i^m\}$. Any data item has a unique position in its corresponding sequence according to the semantic order. In each iteration, we only process the next unaligned data item of every data record and decide which ones should be ahead of all others. The complexity of this algorithm is $O(n^2 * m)$, where $n$ is the number of data records in the sample page and $m$ is the average number of data items per data record.

**Example for data item alignment.** The example shown in Fig. 10 explains the process of data item alignment.

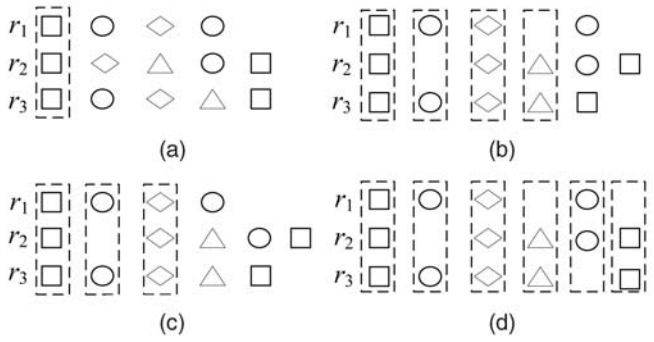



Fig. 10. An example of data item alignment.

Suppose there are three data records $\{r_1, r_2, r_3\}$ and each row is a data record. We use simple geometric shapes (rectangle, circle, triangle, etc.) to denote the data items. The data items represented by the same shape are visually matched data items. We also use $item_i^j$ to denote the $j$th data item of the $i$th data record. Initially (Fig. 10a), all current unaligned data items $\{item_1^1, item_2^1, item_3^1\}$ of the input data records are placed into one cluster, i.e., they are aligned as the first column. Next (Fig. 10b), the current unaligned data items $item_1^2, item_2^2, item_3^2$ are matched into two clusters $C_1 = \{item_1^2, item_3^2\}$ and $C_2 = \{item_2^2\}$ (line 5 in Fig. 9). Thus, we need to further decide which cluster should form the next column. The data items in $C_1$ can match $item_2^4$, and the position value 2 is logged (lines 6-12), which means that $item_2^4$ is the third of the unaligned data items of $r_2$. The data items in $C_2$ can match $item_1^3$ and $item_3^3$, and the position value 1 is logged (lines 6-12). Because 1 is smaller than 2 (line 16), the data items in $C_1$ should be ahead of the data items in $C_2$ and form the next column by inserting the blank item into other records at the current positions (lines 21-22). The remaining data items can be aligned in the same way (Figs. 10c and 10d).

## 6 Visual Wrapper Generation

ViDE has two components: ViDRE and ViDIE. There are two problems with them. First, the complex extraction processes are too slow in supporting real-time applications. Second, the extraction processes would fail if there is only one data record on the page. Since all deep Web pages from the same Web database share the same visual template, once the data records and data items on a deep Web page have been extracted, we can use these extracted data records and data items to generate the extraction wrapper for the Web database so that new deep Web pages from the same Web database can be processed using the wrappers quickly without reapplying the entire extraction process.

Our wrappers include data record wrapper and data item wrapper. They are the programs that do data record extraction and data item extraction with a set of parameter obtained from sample pages. For each Web database, we use a normal deep Web page containing the maximum number of data records to generate the wrappers. The wrappers of previous works mainly depend on the structures or the locations of the data records and data items in the tag tree, such as tag path. In contrast, we mainly use the visual information to generate our wrappers. Note that some other kinds of information are also utilized to enhance the performances of the wrappers, such as the data types of the data items and the frequent symbols appearing in the data items. But they are easy to obtain from the Web pages. We describe the basic ideas of our wrappers below.

TABLE 6
Explanation for $(f, l, d)$

| Parameter | Value | Remarks |
|---|---|---|
| $f$ | font | the font used by the data items of this attribute |
| $l$ | Boolean | *True* denotes that the data items of this attribute are link texts |
| $d$ | image, text, number, date, email, etc | the data type of this attribute |

### 6.1 Vision-Based Data Record Wrapper

Given a deep Web page, vision-based data record wrapper first locates the data region in the Visual Block tree, and then, extracts the data records from the child blocks of the data region.

**Data region location.** After the data region $R$ on a sample deep Web page $P$ from site $S$ is located by ViDRE, we save five parameters values $(x, y, w, h, l)$, where $(x, y)$ form the coordinate of $R$ on $P$, $w$ and $h$ are the width and height of $R$, and $l$ is the level of $R$ in the Visual Block tree.

Given a new deep Web page $P^*$ from $S$, we first check the blocks at level $l$ in the Visual Block tree for $P^*$. The data region on $P^*$ should be the block with the largest area overlap with $R$ on $P^*$. The overlap area can be computed using the coordinates and width/height information.

**Data record extraction.** For each record, our visual data record wrapper aims to find the first block of each record and the last block of the last data record (denoted as $b_{last}$).

To achieve this goal, we save the visual information (the same as the information used in (1)) of the first block of each data record extracted from the sample page and the distance (denoted as $d$) between two data records. For the child blocks of the data region in a new page, we find the first block of each data record by the visual similarity with the saved visual information. Next, $b_{last}$ on the new page needs to be located. Based on our observation, in order to help the users differentiate data records easily, the vertical distance between any two neighboring blocks in one data record is always smaller than $d$ and the vertical distance between $b_{last}$ and its next block is not smaller than $d$. Therefore, we recognize the first block whose distance with its next block is larger than $d$ as $b_{last}$.

### 6.2 Vision-Based Data Item Wrapper

The data alignment algorithm groups data items from different data records into columns or attributes such that data items under the same column have the same semantic. Table 6 lists useful information about each attribute obtained from the sample page that can help determine which attribute a data item belongs to.

The basic idea of our vision-based data item wrapper is described as follows: Given a sequence of attributes

$\{a_1, a_2, \ldots, a_n\}$ obtained from the sample page and a sequence of data items $\{item_1, item_2, \ldots, item_m\}$ obtained from a new data record, the wrapper processes the data items in order to decide which attribute the current data item can be matched to. For $item_i$ and $a_j$, if they are the same on $f$, $l$, and $d$, their match is recognized. The wrapper then judges whether $item_{i+1}$ and $a_{j+1}$ are matched next, and if not, it judges $item_i$ and $a_{j+1}$. Repeat this process until all data items are matched to their right attributes.

Note that if an attribute on a new page did not appear on the sample page, the data item of the attribute cannot be matched to any attribute. To avoid such a problem, several sample pages may be used to generate the wrapper. This can increase the chance that every attribute appears on at least one of these sample pages.

This process is much faster than the process of wrap-per generation. The complexity of data records extraction with the wrapper is $O(n)$, where $n$ is the number of data records in the page. The complexity of data items extraction with the wrapper is $O(n * m)$, where $n$ is the number of data records in the test page and $m$ is the average number of data items per data record.

## 7 Experiments

We have implemented an operational deep Web data extraction system for ViDE based on the techniques we proposed. Our experiments are done on a Pentium 4 1.9 GH, 512 MB PC. In this section, we first describe the data sets used in our experiments, and then, introduce the performance measures used. At last, we evaluate both ViDRE and ViDIE. We also choose MDR [17] and DEPTA [29] to compare with ViDRE and ViDIE, respectively. MDR and DEPTA are the recent works on Web data record extraction and data item extraction, and they are both HTML-based approaches.

### 7.1 Data Sets

Most performance studies of previous works used small data sets, which are inadequate in assuring the impartiality of the experimental results. In our work, we use a large data set to carry out the experiments.

**GDS.** This data set is collected from CompletePlanet (www.completeplanet.com), which is currently the largest deep Web repository with more than 70,000 entries of Web databases. These Web databases are classified into 42 categories covering most domains in the real world. GDS contains 1,000 available Web databases. For each Web database, we submit five queries and gather five deep Web pages with each containing at least three data records.

**Special data set (SDS).** During the process of obtaining GDS, we noticed that the data records from two-thirds of the Web databases have less than five data items on average. To test the robustness of our approaches, we select 100 Web databases whose data records contain more than 10 data items from GDS as SDS.

Note that the deep Web pages collected in the testbed are the ones that can be correctly displayed by the Web browser we used. An example where a page is not correctly displayed is when some images are displayed as small red crosses. This will cause the positions of the texts on the result page to shift.

TABLE 7
Performance Measures Used in the Evaluation of ViDE

| | *precision* | *recall* | *revision* |
|---|---|---|---|
| ViDRE | $\frac{DR_c}{DR_e}$ | $\frac{DR_c}{DR_r}$ | $\frac{WDB_t - WDB_c}{WDB_t}$ |
| ViDIE | $\frac{DI_c}{DI_e}$ | $\frac{DI_c}{DI_r}$ | |

### 7.2 Performance Measures

All previous works use *precision* and *recall* to evaluate their experimental results (some also include F-measure, which is the weighted harmonic mean of *precision* and *recall*). These measures are also used in our evaluation.

In this paper, we propose a new metric, *revision*, to measure the performance of an automated extraction algorithm. It is defined to be the percentage of the Web databases whose data records or data items are not perfectly extracted, i.e., either *precision* or *recall* is not 100 percent. This measure indicates the percentage of Web databases the automated solution fails to achieve perfect extraction, and manual revision of the solution is needed to fix this. An example is used to illustrate the significance of this measure. Suppose there are three approaches (A1, A2, and A3) which can extract structured data records from deep Web pages, and they use the same data set (five Web databases and 10 data records in each Web database). A1 extracts nine records for each site and they are all correct. So, the average *precision* and *recall* of A1 are 100 and 90 percent, respectively. A2 extracts 11 records for each site and 10 are correct. So, the average *precision* and *recall* of A2 are 90.9 and 100 percent, respectively. A3 extracts 10 records for four of the five databases and they are all correct. For the fifth site, A3 extracts no records. So, the average *precision* and *recall* of A3 are both 80 percent. Based on average *precision* and *recall*, A1 and A2 are better than A3. But in real applications, A3 may be the best choice. To make *precision* and *recall* 100 percent, all wrappers generated by A1 and A2 have to be manually tuned/adjusted, while only one wrapper generated by A3 needs to be manually tuned. In other words, A3 needs the minimum manual intervention.

Because our experiments include data record extraction and data item extraction, we define *precision*, *recall*, and *revision* for them separately.

In Table 7, $DR_c$ is the total number of correctly extracted data records, $DR_r$ is the total number of data records, $DR_e$ is the total number of data records extracted, $DI_c$ is the total number of correctly extracted data items, $DI_r$ is the total number of data items, and $DI_e$ is the total number of data items extracted; $WDB_c$ is the total number of Web databases whose *precision* and *recall* are both 100 percent and $WDB_t$ is the total number of Web databases processed.

### 7.3 Experimental Results on ViDRE

In this part, we evaluate ViDRE and also compare it with MDR. MDR has a similarity threshold, which is set at the default value (60 percent) in our test, based on the suggestion of the authors of MDR. Our ViDRE also has a

TABLE 8
Comparison Results between ViDRE and MDR

| | *dataset* | *precision* | *recall* | *revision* |
|---|---|---|---|---|
| ViDRE | GDS | 98.7% | 97.2% | 12.4% |
| | SDS | 98.5% | 97.8% | 10.9% |
| MDR | GDS | 85.3% | 53.2% | 55.2% |
| | SDS | 78.7% | 47.3% | 63.8% |

TABLE 9
Comparison Results between ViDIE and DEPTA

| | *dataset* | *precision* | *recall* | *revision* |
|---|---|---|---|---|
| ViDIE | GDS | 96.3% | 97.2% | 14.1% |
| | SDS | 95.6% | 98.4% | 11.6% |
| DEPTA | GDS | 75.3% | 71.6% | 32.8% |
| | SDS | 66.1% | 54.1% | 37.6% |

similarity threshold, which is set at 0.8. In this experiment, the input to ViDRE and MDR contains the deep Web pages and the output contains data records extracted. For ViDRE, one sample result page containing the most data records is used to generate the data record wrapper for each Web database. Table 8 shows the experimental results on both GDS and SDS. Based on our experiment, it takes approximately 1 second to generate the data record wrapper for each page and less than half second to use the wrapper for data record extraction.

From Table 8, we can make the following three observations. First, ViDRE performs significantly better than MDR on both GDS and SDS. Second, ViDRE is far better than MDR on revision. ViDRE needs only to revise slightly over 10 percent of the wrappers, while MDR has to revise almost five times more wrappers than ViDRE. Third, the precision and recall of ViDRE are steady on both SDS and GDS, but for MDR, they drop noticeably for SDS. Our analysis indicates that: for *precision* of ViDRE, most errors are caused by failing to exclude noise blocks that are very similar to the correct ones in appearance; for *recall* of ViDRE, most errors are caused by mistaking some top or bottom data records as the noise blocks; for MDR, its performance is inversely proportional to the complexity of the data records, especially data records with many optional data items.

### 7.4 Experimental Results on ViDIE

In this part, we evaluate ViDIE and compare it with DEPTA. DEPTA can be considered as the follow-up work for MDR, and its authors also called it MDRII. Only correct data records from ViDRE are used to evaluate ViDIE and DEPTA. For ViDIE, two sample result pages are used to generate the data item wrapper for each Web database. Table 9 shows the experimental results of ViDIE and DEPTA on both GDS and SDS. Our experiments indicate that it takes between 0.5 and 1.5 seconds to generate the data item wrapper for each page and less than half second to use the wrapper for data item extraction.

From Table 9, we can see that the observations we made in comparing the results of ViDRE and MDR remain basically valid for comparing ViDIE and DEPTA. In addition, we also found that DEPTA often misaligns two data items of different semantics if they are close in the tag tree and have the same tag path, and this leads to the misalignment of all the data items in the same data record that follow the misaligned data items. In contrast, ViDIE can easily distinguish them due to their different fonts or positions.

We also tried to use one sample page and three sample pages to generate the data item wrapper for each Web database. When one page is used, the performance is much lower; for example, for SDS, the precision, recall, and revision are 91.7, 95, and 32.3 percent, respectively. This is caused by the absence of some optional data items from all the data records in the sample page used. When more sample pages are used, the likelihood that this will happen is significantly reduced. When three pages are used, the results are essentially the same as shown in Table 9, where two sample pages are used. This suggests that using two sample pages to generate the data item wrapper for each Web database is sufficient.

We also conducted experiments based on the data sets used in [30] and provided by [13], and the results are similar to those shown in Tables 8 and 9. These results are not shown in this paper due to space consideration.

## 8 Conclusions and Future Works

With the flourish of the deep Web, users have a great opportunity to benefit from such abundant information in it. In general, the desired information is embedded in the deep Web pages in the form of data records returned by Web databases when they respond to users' queries. Therefore, it is an important task to extract the structured data from the deep Web pages for later processing. In this paper, we focused on the structured Web data extraction problem, including data record extraction and data item extraction. First, we surveyed previous works on Web data extraction and investigated their inherent limitations. Meanwhile, we found that the visual information of Web pages can help us implement Web data extraction. Based on our observations of a large number of deep Web pages, we identified a set of interesting common visual features that are useful for deep Web data extraction. Based on these visual features, we proposed a novel vision-based approach to extract structured data from deep Web pages, which can avoid the limitations of previous works. The main trait of this vision-based approach is that it primarily utilizes the visual features of deep Web pages.

Our approach consists of four primary steps: Visual Block tree building, data record extraction, data item extraction, and visual wrapper generation. Visual Block tree building is to build the Visual Block tree for a given sample deep page using the VIPS algorithm. With the Visual Block tree, data record extraction and data item extraction are carried out based on our proposed visual features. Visual wrapper generation is to generate the

wrappers that can improve the efficiency of both data record extraction and data item extraction. Highly accurate experimental results provide strong evidence that rich visual features on deep Web pages can be used as the basis to design highly effective data extraction algorithms.

However, there are still some remaining issues and we plan to address them in the future. First, ViDE can only process deep Web pages containing one data region, while there is significant number of multidata-region deep Web pages. Though Zhao et al. [31] have attempted to address this problem, their solution is HTML-dependent and its performance has a large room for improvement. We intend to propose a vision-based approach to tackle this problem. Second, the efficiency of ViDE can be improved. In the current ViDE, the visual information of Web pages is obtained by calling the programming APIs of IE, which is a time-consuming process. To address this problem, we intend to develop a set of new APIs to obtain the visual information directly from the Web pages.

## ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation of China under grant 60833005, the National High Technology Research and Development Program of China (863 Program) under grant 2007AA01Z155 and 2009AA011904, the Doctoral Fund of Ministry of Education of China under grant 200800020002, the China Postdoctoral Science Foundation funded project under grant 20080440256 and 200902014, and US National Science Foundation (NSF) grants IIS-0414981 and CNS-0454298. The authors would also like to express their gratitude to the anonymous reviewers for providing some very helpful suggestions.

## REFERENCES

[1] G.O. Arocena and A.O. Mendelzon, "WebOQL: Restructuring Documents, Databases, and Webs," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 24-33, 1998.
[2] D. Buttler, L. Liu, and C. Pu, "A Fully Automated Object Extraction System for the World Wide Web," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 361-370, 2001.
[3] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma, "Block-Level Link Analysis," *Proc. SIGIR,* pp. 440-447, 2004.
[4] D. Cai, S. Yu, J. Wen, and W. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," *Proc. Asia Pacific Web Conf. (APWeb),* pp. 406-417, 2003.
[5] C.-H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan, "A Survey of Web Information Extraction Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 10, pp. 1411-1428, Oct. 2006.
[6] C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic Information Extraction from Semi-Structured Web Pages by Pattern Discovery," *Decision Support Systems,* vol. 35, no. 1, pp. 129-147, 2003.
[7] V. Crescenzi and G. Mecca, "Grammars Have Exceptions," *Information Systems,* vol. 23, no. 8, pp. 539-565, 1998.
[8] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 109-118, 2001.
[9] D.W. Embley, Y.S. Jiang, and Y.-K. Ng, "Record-Boundary Discovery in Web Documents," *Proc. ACM SIGMOD,* pp. 467-478, 1999.
[10] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krpl, and B. Pollak, "Towards Domain Independent Information Extraction from Web Tables," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 71-80, 2007.
[11] J. Hammer, J. McHugh, and H. Garcia-Molina, "Semistructured Data: The TSIMMIS Experience," *Proc. East-European Workshop Advances in Databases and Information Systems (ADBIS),* pp. 1-8, 1997.
[12] C.-N. Hsu and M.-T. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," *Information Systems,* vol. 23, no. 8, pp. 521-538, 1998.
[13] http://daisen.cc.kyushu-u.ac.jp/TBDW/, 2009.
[14] http://www.w3.org/html/wg/html5/, 2009.
[15] N. Kushmerick, "Wrapper Induction: Efficiency and Expressiveness," *Artificial Intelligence,* vol. 118, nos. 1/2, pp. 15-68, 2000.
[16] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira, "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record,* vol. 31, no. 2, pp. 84-93, 2002.
[17] B. Liu, R.L. Grossman, and Y. Zhai, "Mining Data Records in Web Pages," *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 601-606, 2003.
[18] W. Liu, X. Meng, and W. Meng, "Vision-Based Web Data Records Extraction," *Proc. Int'l Workshop Web and Databases (WebDB '06),* pp. 20-25, June 2006.
[19] L. Liu, C. Pu, and W. Han, "XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 611-621, 2000.
[20] Y. Lu, H. He, H. Zhao, W. Meng, and C.T. Yu, "Annotating Structured Data of the Deep Web," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 376-385, 2007.
[21] J. Madhavan, S.R. Jeffery, S. Cohen, X.L. Dong, D. Ko, C. Yu, and A. Halevy, "Web-Scale Data Integration: You Can Only Afford to Pay As You Go," *Proc. Conf. Innovative Data Systems Research (CIDR),* pp. 342-350, 2007.
[22] I. Muslea, S. Minton, and C.A. Knoblock, "Hierarchical Wrapper Induction for Semi-Structured Information Sources," *Autonomous Agents and Multi-Agent Systems,* vol. 4, nos. 1/2, pp. 93-114, 2001.
[23] Z. Nie, J.-R. Wen, and W.-Y. Ma, "Object-Level Vertical Search," *Proc. Conf. Innovative Data Systems Research (CIDR),* pp. 235-246, 2007.
[24] A. Sahuguet and F. Azavant, "Building Intelligent Web Applications Using Lightweight Wrappers," *Data and Knowledge Eng.,* vol. 36, no. 3, pp. 283-316, 2001.
[25] K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual Perceptions," *Proc. Conf. Information and Knowledge Management (CIKM),* pp. 381-388, 2005.
[26] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, "Learning Block Importance Models for Web Pages," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 203-211, 2004.
[27] J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 187-196, 2003.
[28] X. Xie, G. Miao, R. Song, J.-R. Wen, and W.-Y. Ma, "Efficient Browsing of Web Search Results on Mobile Devices Based on Block Importance Model," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PerCom),* pp. 17-26, 2005.
[29] Y. Zhai and B. Liu, "Web Data Extraction Based on Partial Tree Alignment," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 76-85, 2005.
[30] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C.T. Yu, "Fully Automatic Wrapper Generation for Search Engines," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 66-75, 2005.
[31] H. Zhao, W. Meng, and C.T. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 989-1000, 2006.
[32] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma, "Simultaneous Record Detection and Attribute Labeling in Web Data Extraction," *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 494-503, 2006.

**Wei Liu** received the BS and MS degrees in computer science from Shandong University in 1998 and 2004, respectively, and the PhD degree in computer science from Renmin University of China in 2008. Since 2008, he has been a postdoctoral fellow in computer science in the Institute of Computer Science and Technology of Peking University. His research interests include Web data extraction and deep Web data integration.

**Xiaofeng Meng** received the BS degree from Hebei University, the MS degree from Remin University of China, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, all in computer science. He is currently a professor in the School of Information, Renmin University of China. His research interests include Web data integration, native XML databases, mobile data management, and flash-based databases. He is the secretary general of Database Society of the China Computer Federation (CCF DBS). He has published more than 100 technical papers. He is a member of the IEEE.

**Weiyi Meng** received the BS degree in mathematics from Sichuan University, China, in 1982, and the MS and PhD degrees in computer science from the University of Illinois at Chicago, in 1988 and 1992, respectively. He is currently a professor in the Department of Computer Science at the State University of New York at Binghamton. His research interests include Web-based information retrieval, metasearch engines, and Web database integration. He is a coauthor of a database book *Principles of Database Query Processing for Advanced Applications*. He has published more than 100 technical papers. He is a member of the IEEE.

In Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA 2009):595-600, April 21-23, 2009, Brisbane, Australia

# Selectivity Estimation for Exclusive Query Translation in Deep Web Data Integration

Fangjiao Jiang[1,2], Weiyi Meng[3], and Xiaofeng Meng[1]

[1]School of Information, Renmin University of China
{jiangfj, xfmeng2006}@gmail.com
[2]College of Physics and Electronic Engineering, Xuzhou Normal University
[3]Computer Science Dept, SUNY at Binghamton, meng@cs.binghamton.edu

**Abstract.** In Deep Web data integration, some Web database interfaces express exclusive predicates of the form $Q_e = P_i(P_i \in P_1, P_2, \ldots, P_m)$, which permits only one predicate to be selected at a time. Accurately and efficiently estimating the selectivity of each $Q_e$ is of critical importance to optimal query translation. In this paper, we mainly focus on the selectivity estimation on infinite-value attribute which is more difficult than that on key attribute and categorical attribute. Firstly, we compute the attribute correlation and retrieve approximate random attribute-level samples through submitting queries on the least correlative attribute to the actual Web database. Then we estimate Zipf equation based on the word rank of the sample and the actual selectivity of several words from the actual Web database. Finally, the selectivity of any word on the infinite-value attribute can be derived by the Zipf equation. An experimental evaluation of the proposed selectivity estimation method is provided and experimental results are highly accurate.

## 1 Introduction

The Deep Web continues to grow rapidly [1], which makes exploiting useful information a remarkable challenge. Metaquerier, which provides a uniform integrated interface to the users and can query multiple databases simultaneously, is becoming the main trend for Deep Web data integration.

Query translation plays an important role in a metaquerier. However, due to the large-scale, heterogeneity and autonomy of the Web databases, automatic query translation is challenging. One of the important aspects is that Web database interfaces may express different predicate logics. The integrated query interface and many Web database interfaces express conjunctive predicates of the form $Q_c = P_1 \wedge P_2 \wedge \ldots \wedge P_m$, where $P_i$ is a simple predicate on single attribute. While some Web database interfaces express exclusive predicates of the form $Q_e = P_i(P_i \in P_1, P_2, \ldots, P_m)$, which means any given query can only include one of these predicates. Exclusive attributes are often represented on a Web database interface as a selection list of attribute names or a group of radio buttons each of which is an attribute. A very interesting problem is, among all the $Q_e s$ on an interface, which one has the lowest selectivity? It is of critical importance to optimal query translation. In this paper, we mainly focus on the selectivity estimation of infinite-value attribute for exclusive query translation.

Before we carry out our study, we have two important observations: 1) there exist different correlations between different attribute pairs, and 2) the word frequency of the values on an infinite-value attribute usually has a Zipf-like distribution. Based on these observations, we propose a correlation-based sampling approach to obtain the approximate random attribute-level sample and a Zipf-based approach that can estimate the selectivity of any word by Zipf equation.

The rest of paper is organized as follows. Section 2 gives the overview of query selectivity estimation. Section 3 proposes the correlation-based sampling approach. Section 4 proposes a Zipf-based selectivity estimation approach. Section 5 reports the results of experiments. Section 6 introduces the related work. Section 7 concludes the paper.

## 2 An Overview of Query Selectivity Estimation

The overall flow chart of our approach is given in Fig.1.

**Attribute correlation calculation for a domain.** For any given domain (e.g., Books), we first calculate attribute correlation for each pair of attributes (***Attribute Correlation calculation)*** and identify the least correlative attribute $Attr_i$ for each specific attribute $Attr_u$. Because attribute correlation of each attribute pair in a domain is usually independent of the Web databases, the attribute correlation can be used for all the Web databases in the same domain.

**Selectivity estimation for a Web database.** Given an infinite-value attribute $Attr_u$ and a specific Web database, we use a series of query probes on $Attr_i$ in the Web database interface to obtain an approximate random attribute-level sample on $Attr_u$ (***Correlation-based sampling)***. The word rank on $Attr_u$ can be calculated from the sample, which is viewed as the actual word rank on $Attr_u$ of the Web database due to the randomness of the sample. Then several words on $Attr_u$ are used to probe the actual Web database and the frequencies of these words are returned (***Word frequency probing)***. Zipf equation can be estimated using the word ranks and the actual frequencies of several words (***Zipf equation calculation)***. Finally, for any word on $Attr_u$, we can estimate its frequency by the Zipf equation and its rank (***Selectivity estimation)***.

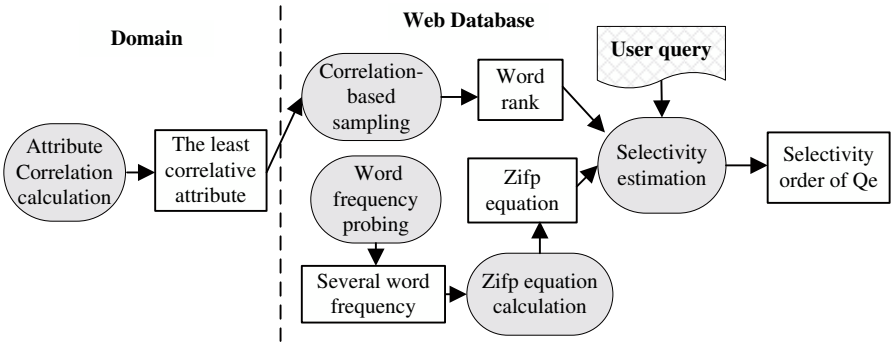



**Fig. 1.** The processing flow of our approach.

## 3 Correlation-based Sampling for Word Rank

In this paper, we use *Attribute Word Distribution* of different attributes to define the concept of attribute correlation.

**Definition 1 Attribute Word Distribution (AWD)**. *Given all the words $w_1$, $w_2$, ..., $w_m$ of the values of attribute A in a database D, the Attribute Word Distribution for A is a vector $\overrightarrow{v}(v_1, v_2, \ldots, v_m)$, each component of which $v_i$ is the frequency of the word $w_i$. Under the assumption that no word appears more than once in an attribute value, the frequency of the word $w_i$ is the number of tuples returned by the query $\sigma_{A=w_i} D$.*

**Definition 2 Attribute Correlation**. *Attribute Correlation is the dependence between any attribute pair($Attr_u$, $Attr_v$) and is measured by the difference of the Attribute Word Distributions of the returned results on an attribute ($Attr_u$).*

A measure of the distribution difference is Kullback-Leibler(KL) divergence. If we submit different queries $Q_1$, $Q_2$, ..., $Q_s$ on $Attr_v$, we will gain the corresponding result sets $S_1$, $S_2$, ..., $S_s$ on $Attr_u$. Suppose that $S$ is the union of $S_1$, $S_2$, ..., $S_s$ and $S$ consists of a set of words $w_1$, $w_2$, ..., $w_k$. Then the KL-divergence of $Attr_u$ from $S$ to $S_j$ is:

$$D_{KL}(S||S_j) = \sum_{l=1}^{k} prob(Attr_u = w_l|S) log \frac{prob(Attr_u = w_l|S)}{prob(Attr_u = w_l|S_j)}$$

where prob($Attr_u$ =$w_l$| $S$) refers to the probability that $Attr_u$=$w_l$ in S and prob($Attr_u$=$w_l$| $S_j$) refers to the probability that $Attr_u$=$w_l$ in $S_j$.

Attribute correlation is the average of the KL divergence of $Attr_u$ from $S$ to $S_j$:

$$Correlation(Attr_u, Attr_v) = \frac{1}{s} \sum_{j=1}^{s} D_{KL}(S||S_j)$$

After discovering the least correlative attribute $Attr_i$, we submit some query probes on $Attr_i$ to the Web databases and collect the returned results on attribute $Attr_u$ as the attribute-level sample of $Attr_u$, which is the approximate random sample. Then we order the words of the sample by their frequencies and the word rank can be viewed as the actual one due to the randomness of the sample.

## 4 Zipf-based Selectivity Estimation

It is well known that English words of a general corpus satisfy the Zipf distribution. However, it is not clear if the words of text attributes in different domains also follow this distribution. Our experiments indicate that they do.

Zipf distribution can be represented by $N = P(r + p)^{-E}$ [9], where $N$ represents the frequency of the word, $r$ represents the rank of the word and $P$, $p$ and $E$ are the positive parameters. As Fig.2 shows, we submit word $i$, word $j$ to the Web database and obtain their frequencies $F_{wi}$(i.e., $N_i$) and $F_{wj}$(i.e., $N_j$), respectively. And we know the ranks of these two words (i.e., $r_i$ and $r_j$) from the sample obtained in section 3. Then, we can estimate the parameters $P$, $p$ and $E$ as follows.

– Equation Transformation: After the logarithm transformation, the Zipf equation is changed to $ln(N) = lnP - Eln(r+p)$. Because the parameter $p$ $(0 < p < 1)$ is usually much smaller than word rank $r$(i.e., some applications even assume $p = 0$), the parameter $E$ is approximately viewed as the slope of the line $ln(N) = lnP - Eln(r)$ as shown in Fig.3.
– Parameter $E$: $E$ can be calculate by the equation $E \approx \frac{ln(N_i)-ln(N_j)}{ln(r_j)-ln(r_i)}$.
– Parameter $p$: When $E$ is estimated, parameter $p$ can be derived from the equation $\frac{N_i}{N_j} = \frac{P*(r_i+p)^{-E}}{P*(r_j+p)^{-E}}$. So we have $p \approx \frac{r_j - r_i * e^m}{e^m - 1}$ $(m = \frac{1}{E} * \ln\frac{N_i}{N_j})$.
– Parameter $P$: Finally, the parameter $P$ is derived. $P \approx N_j * (r_j + p)^E$.

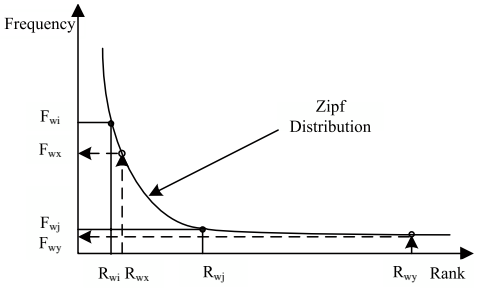



**Fig. 2.** Zipf-based Selectivity Estimation.

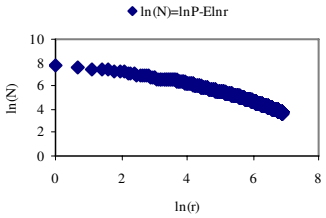



**Fig. 3.** Distribution transformation

Consequently, we can use the Zipf equation and the word ranks to compute the selectivity of any word on the attribute.

It is worth noticing that the parameters $P$, $p$ and $E$ are not unique. We study the relationships among the precision, word ranks and rank distances. The results show that the precision will go down when the rank increases and to keep the precision stable, the distance of two word ranks should be increased with the increase of the word ranks.

## 5 Experiments

We evaluate our approach with the precision measure which is defined as follows.

$$Precision = \frac{1}{N} \sum_{n} \left| \frac{N_r - E_s}{N_r} \right|$$

where $N_r$ is the number of results when submitting the word on the attribute to the actual Web database, $E_s$ is the selectivity of the word on the same attribute estimated by our approach, and $n$ is the number of the words that we test in the experiments.

We select the top 100 words on *Title*, *Conference* attribute of Libra, *Title*, *Director* attribute of IMDb and submit them to actual Web databases. Meanwhile, we estimate the selectivity of these words using our approach. Overall, as we can see from Fig.4, the precision of our approach is generally good.

However, there is still some deviation on estimation values. The reasons are that any two attributes are somehow correlative with each other and the words on some infinite-value attributes do not satisfy Zipf distribution perfectly.

Given that our approach can cope with selectivity estimation of all the infinite-value attributes and it is domain independent, it is generally feasible to be applied in query translation for exclusive query interface.

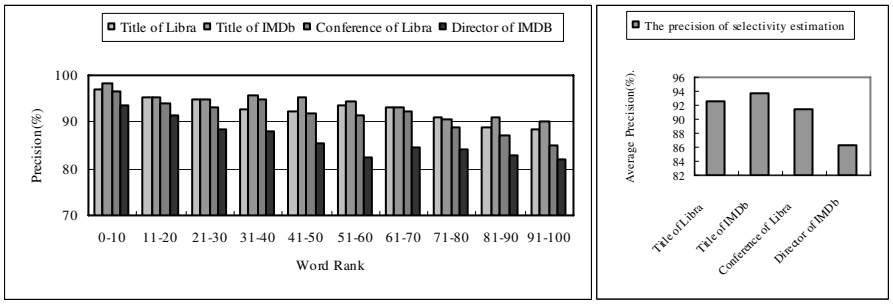



**Fig. 4.** The precision of selectivity estimation.

## 6 Related Works

The problem of selectivity estimation through uniform random sampling has received considerable attention [2, 6]. [2] cannot be applied as we do not have full access to the Web databases. [6] proposes a random walk approach to sampling the hidden databases, which is a database-level sampling and relatively complex compared with our attribute-level sampling. [3] focuses on the selectivity estimation of the text type attribute with several constraints (e.g., *any*, *all* or *exactly*, etc.) in Web database interfaces.

## 7 Conclusions

In this paper, we study the query translation problem of the exclusive query interface and present a novel Zipf-based selectivity estimation approach for infinite-value attribute. Experimental results on several large-scale Web databases indicate that our approach can achieve high precision on selectivity estimation of infinite-value attribute for exclusive query translation.

**Acknowledgments.** This work is supported in part by China 863 High-Tech Program(No.2007AA01Z155); NSF of China (No.60833005, 60573091); China National Basic Research and Development Program's Semantic Grid Project (No.2003CB317000). US NSF grants IIS-0414981 and CNS-0454298.

## References

1. The Deep Web: Surfacing Hidden Value. http://www.completeplanet.com/Tutorials/
2. Oliken. F.: Random Sampling from databases. PhD Thesis, University of California, Berkeley (1993).
3. Zhang. Z., He. B., Chang. K. C. C.: On-the-fly Constraint Mapping across Web Query Interfaces. In: IIWEB (2004).
4. Mandelbrot. B. B.: Fractal Geometry of Nature. W. H. Freeman and Co. (1988).
5. Si. L., Callan. J. P. : Relevant Document Distribution Estimation Method for Resource Selection. In: SIGIR. (2003) 298-305.
6. Dasgupta. A., Das. G., Mannila. H.: A random walk approach to sampling hidden databases. In: SIGMOD. (2007) 629-640.

ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
E-mail:fcst@public2.bta..net.cn
http://www.ceaj.org
Tel: +86-10-51616056

# C-Rank: 一种 Deep Web 数据记录可信度评估方法*

艾静， 王仲远， 孟小峰+

中国人民大学 信息学院，北京 100872

# C-Rank: A Credibility Evaluation Method for Deep Web Records

AI Jing, WANG Zhong-yuan, MENG Xiao-feng

School of Information, Renmin University of China, Beijing 100872, China

+ Corresponding author: E-mail: xfmeng@ruc.edu.cn

**Abstract:** How to identify and evaluate information credibility ranking has become an increasing important problem. To address the issue, an effective credibility evaluation method called C-Rank to compute trust values of records in Deep Web databases is proposed, which constructs an S-R Credibility Graph for each record. The graph contains 2 types of vertices and 3 types of edges. Firstly, each vertex's local trust value is computed by using out-degrees based on the idea of trust propagation. Then, the weight of Record vertex is computed by using its in-degree and adjacent Site vertices' local trust values. Lastly the global trust value of the S-R Credibility Graph is computed, which denotes the record's credibility in the whole Web. Experiment results show C-Rank can evaluate credibility rankings of records appropriately and discriminate false information effectively. This method is generally applicable to all domains of Deep Web.

**Key words:** Deep Web; Web Information Credibility; S-R Credibility Graph; Trust Propagation

**摘 要**：针对 Web 信息可信度问题，提出了一种为 Deep Web 数据记录计算可信度的有效方法 C-Rank。该方法为每一条记录构造一个 S-R 可信度网络，包含两种类型顶点及三种类型边。首先基于可信度传播的思想，利用顶点出度为每一个顶点计算其局部可信度值；再利用 Record 顶点入度及相邻 Site 顶点的可信度值，为该 Record 顶点计算权值；继而求得整个 S-R 网络的全局可信度值。实验证明，C-Rank 方法能够合理而有效地评价数据记录的可信度，从而达到甄别虚假信息，为用户推荐可信数据记录的目的。该方法普遍适用于 Deep Web 的各个领域。
**关键词**： 深层网络; Web 信息可信度; S-R 可信度网络; 可信度传播
**文献标识码**：A　　**中国分类号**：TP391

## 1 引 言

随着网络与通信技术的迅速发展，Web 上的信息呈现爆炸性增长，互联网上存储了海量信息。其中，面向领域的 Deep Web[1]数据源是网络上一种非常重要的数据源，包含极为丰富的结构化数据，成为人们获取信息的一个重要途径。

以工作信息领域为例，越来越多的用户开始依赖互联网来获取招聘信息，寻找理想的工作。但是，面对纷繁复杂的数据源（如招聘网站、博客、论坛等），以及每日成千上万条新发布的招聘信息，用户无法对所有招聘信息的可信度进行正确辨别。而这些信息的可信度可能会涉及到如下两个重要问题：

（1）用户的隐私泄露问题

通常应聘者的简历上包含许多个人信息。因此，一些不法分子通过发布虚假的招聘信息，收集用户的简历，从而获取他的个人信息，并将其转让给第三方，

*the grants from the Natural Science Foundation of China under Grant No.60833005, 60573091(国家自然科学基金项目); National High-Tech Research and Development Plan of China under Grant No.2007AA01Z155，2009AA011904(国家863高技术研究发展计划); the Ph.D. Programs Foundation of Ministry of Education of China No.200800020002(教育部博士点基金项目)
Received 2009-07, Accepted 2009-

从而导致用户隐私泄露问题。更有甚者，利用简历中的个人信息去填写信用卡申请表，然后刷卡或提现，给应聘者造成巨大的损失。

因此，这是一个不容忽视的问题，也是为 Deep Web 数据记录计算可信度的迫切需求。

（2）数据记录最优选择问题

在择业时，招聘公司的数量远远超过一名应聘者所能够了解的能力范围。面对一家未知的公司，应聘者常常需要花费大量的时间和精力，来查询这家公司是否是一家与其描述相符的公司。例如，其有可能是一家刚进入中国不久的世界 500 强企业，也可能是一个随时都会倒闭的“皮包公司”。用户只有了解了这条招聘信息的真实可信度，才能够根据自己的实际情况进行正确的选择。

在其它领域，如网上购书、预订机票、转让二手物品等，其数据记录的可信度问题也是关系到用户切身利益的一个重要问题。因此，数据记录的可信度问题在 Deep Web 的许多领域里都广泛存在。

为了解决数据记录的可信度问题，通过对工作领域数据记录的观察我们发现，Web 上的招聘信息，其可信度具有如下两个重要特点：

（1）某条招聘信息所发布的网站的可信度越高，通常这条招聘信息的可信度就越高。如图 1 所示，是一条 Google 招聘工程师的信息，它在多个网站上发布。由于这条招聘信息出现在了 Google 的网站上，以及高校招生就业网等高可信度的网站，所以这条招聘信息的可信度也比较高。

（2）某条招聘信息被转载的次数越多，通常这条招聘信息的可信度就越高。这是由于用户在转载这条招聘信息的时候，通常会进行一些判断。也就是说，在转载的时候，这条信息就包含了一定的人类的知识在其中。在图 1 中，这条 Google 的招聘信息，被转载到了论坛、博客等网站。

忽视上述任何一点，都无法完全断定一条数据记录的可信度。例如，一则招聘信息虽然有可能只发布在一家著名公司的网站上，但是这条招聘信息的可信度就极高；而如果一个恶意的信息发布者，即使他在许多论坛上重复发布、转载同一条招聘信息，这条信息仍然是不可信的。因此，为了真实展现一条 Deep Web 上的招聘记录的可信度，只有综合上述两个特点，才能够计算得出正确的可信度值。

通过如上两点在工作信息领域的观察，本文提出了一种为 Deep Web 中的数据记录自动构建一个可信度网络，通过计算局部可信度值以及全局可信度值，为


**Fig.1** A recruitment record of Google on the Web

**图 1** Web 上关于 Google 的一条招聘信息

用户评价一条 Deep Web 数据记录可信度的方法。

这种方法，通过设置传播率参数以及采取适当的全局可信度计算方法，易于扩展到 Deep Web 的各个领域，具有较强的适用性。

## 2 相关工作

关于 Web 上的信息可信性研究，是目前的一个研究热点问题。文献[2]是早期的工作之一，提出了一种将名声、信誉、口碑等信任管理的社会机制引入计算中网络中，辨别信息可信度的方法。

在 Web 上的信息可信性问题上，研究者们通常比较关注新闻网站[3,4]以及有评论系统网站[5,6]上的信息可信度。这些工作一般都是以网页为基本单元，通过各种方法将该网页上信息的可信度计算出来。这些网页，绝大多数是 Surface Web 页面。而 Deep Web（即 Web 数据库）中的记录的可信性问题，就较少被研究者关注。而本文研究的正是工作信息领域中的招聘记录的可信性问题。

Deep Web 数据库中的记录之间并不是完全孤立隔离开的。根据记录与网站之间的链入和链出关系，以及不同的 Deep Web 数据库中记录重复出现的关系，可以形成网状结构，该网络的节点不再是通常的网页，而是一条条的记录。而在网状结构上辨别和维护节点的可信度，最常用的方法是信任值传播机制。在文献[7]提出的方法中，每个节点只需维护一个与它有直接关联的邻居节点的可信度值向量，网络中其他节点的可信度值均可根据节点信任向量的共享机制以及传播法得到。文献[8,9]也是传播法和信任值传递模型的典型代

表。然而，在由 Deep Web 记录组成的网络中，并不是所有的节点之间都是相异的。关于同一领域的不同网站的 Deep Web 数据库之间经常会有重叠部分，即同一条记录可能会在网上重复出现多次。因此网络中可能有多个节点表示同一条记录。这是我们的研究问题和上述文献中的问题的不同之处，在目前已有的可信性研究相关文献中均未涉及到如何在这种情况下进行信任值的传播与计算。

在这种网络上做节点可信度计算和维护的研究，还需要解决表示同一条记录节点之间的信任值相互影响的问题，以及它们最终合并的问题。在这方面，Deep Web 上的实体识别[10]是一个值得借鉴的工作，该文献提出了一种在两个 Deep Web 数据库之间识别出相同记录的方法，对于我们辨别哪些记录节点之间有这种关系以及后续的信任值影响研究有很大的帮助。

## 3 问题分析

Deep Web 数据库中记录的可信度，与它所在的网站的可信度值有关，也与链接指向它的网站的可信度值有关。

首先，设想这样一个例子：如果 Google 在它的网站上发布了一条“招聘工程师”的记录，显然地，这条记录的可信度一定比较高。如图 2 所示，这条记录发布在 google.com 网站上，而 google.com 本身是一个可信度值非常高的网站，因此它发布的信息的可信度也比较高。


**Fig.2** Relationships among recruitment records on the Deep Web

**图 2** Deep Web 上招聘信息链接关系

而与此对应的是，专门的招聘信息发布网站，如 51job，智联招聘等网站，包含成千上万条招聘信息。与上面例子中 Google 发布招聘信息不同，专门的招聘信息发布网站包含的记录并不是由这个网站发布的，网站只是给发布招聘信息的公司提供一个平台。即使 51job 网站本身的可信度值较高（确实是一家发布工作信息的专门网站），也并不能说明它上面所有招聘记录的可信度值都比较高。

还有一些公司，如中国移动研究院，并不把招聘的详细信息放在自己的网站上，而是外包给 51job 这样的平台，在 51job 上发布完整的招聘记录。而这样的记录，一般都会带有超链接，指向中国移动研究院的网站。而中国移动研究院的网站上也有指向这条招聘记录的链接，如图 2 所示。这种记录的可信度值也是比较高的。

此外，还有一些数据记录会被不断转载，如图 1 所示，Google 招聘工程师的记录被好几个高校的就业网转载，也被 51job、智联招聘、中华英才网等工作信息网站转载，而每一次转载，都相当于经过了转载者（或网站的建设者）的一次对其可信度的鉴别，所以如果一条记录被多个网站所指向，或重复出现在多个网站中，也能说明它的可信度较高。这相当于是综合了多个转载用户的集体智慧和判断。

通过以上分析，我们可以得出在 Deep Web 中为数据记录计算数据可信度需要考虑如下特点：

(1) 可信度值越高的网站，其发布的数据记录的可信度值也越高；

(2) Web2.0 信息共享平台、Deep Web 数据发布平台需要与专业机构等传统的网站平台区分开；

(3) 不同数据源之间的可信度值可以通过链接相互传递；

(4) 同一数据记录在不同数据源出现次数越多，其可信度值越高。

综上所述，一条招聘记录的可信度是多种因素共同作用的结果。基于工作信息领域的大量观察，我们设计了一个算法，引入这些影响因素，为 Deep Web 中的数据记录计算可信度。本文将在下一章节进行重点介绍。

## 4 S-R可信度网络构建与计算

### 4.1 面向Deep Web的S-R可信度网络

在 Deep Web 中，每一个领域都存在着大量的数据源。信息在数据源中以记录的形式存在。通常情况下，同一个领域的不同数据源之间是相互隔绝的，不存在直接的联系。然而，同一条记录可能会重复出现在多个不同的数据源中，因此数据源之间可能会有内容重叠。而且，不同数据源中的记录之间虽然没有直接的关联，但是通过网站之间的关联关系，也可以为记录建立间接的联系。根据数据源之间的重叠以及记录与网站之间的关系，本文提出了一种针对某一条记录构建 S-R 可信度网络的方法。

**定义 1（S-R 可信度网络）**针对 Deep Web 中某一条记录而构造的一个包含两种类型顶点、三种类型边的网络。

在 S-R 可信度网络中，顶点包含两类：

(1)**Site 顶点($v^s$)**：含有数据记录的网站。例如，在招聘信息领域，这类顶点指的就是一个个网站，包括专业工作信息发布网站，例如中华英才网、51job 等；公司或机构的网站；论坛、博客、bbs 以及各大高校的就业网站等。

(2)**Record 顶点($v^r$)**：各个网站上的数据记录。在一个 S-R 可信度网络中的所有 Record 顶点，代表的都是同一内容的数据，只是它们可能散布在不同的数据源里。例如，在招聘信息领域，这类顶点指的就是不同网站的同一条招聘记录。

S-R 可信度网络的边包含三类：

(1)**内部链接边($e^i$)**：如果一个记录是属于某一个数据源的，那么在 S-R 网络中，这条记录对应的 Record 顶点与这个数据源对应的 Site 顶点之间存在一条有向边，边的方向是从 Site 顶点指向 Record 顶点。这种有向边表示网站与它包含的记录之间的关系，因此被称为内部链接边。

(2)**外部链接边($e^o$)**：记录与记录，以及记录与外部数据源之间的链接关系用外部链接边表示。如果一条记录中包含一个指向外部数据源的链接，或某个数据源里有一个链接指向其他数据库中的一条记录，那么在 S-R 网络中，其对应的顶点间存在一条有向边，边的方向与链接指向的方向相同；

(3)**实体识别边($e^r$)**：如果属于不同数据源的两条记录经过实体识别技术[10]验证，被认为是表示同一实体，则这两个 Record 顶点之间存在一条无向边。

这样，按照如上定义，我们就可以为 Deep Web 中的某一条记录，通过链接关系以及实体识别技术，为其构造出一个 S-R 可信度网络，如图 3 所示。

在一个 S-R 网络中，所有 Record 顶点（用椭圆形表示）在理论上应该都是表示同一条记录。因此每一条记录都有一个它自己的 S-R 网络图。目前，两两数据源之间的实体识别能够达到较高的准确率，而大规模数据源之间的实体识别工作仍未取得有效成果，因此，本文所构造的 S-R 网络中，所有 Record 顶点之间并不是一个完全子图。

S-R 网络中的多个 Record 顶点表示同一条记录在整个 Web 上重复出现在不同数据源中的情况。矩形节点表示与这些记录有关联（内部链接边和外部链接边）的网站。

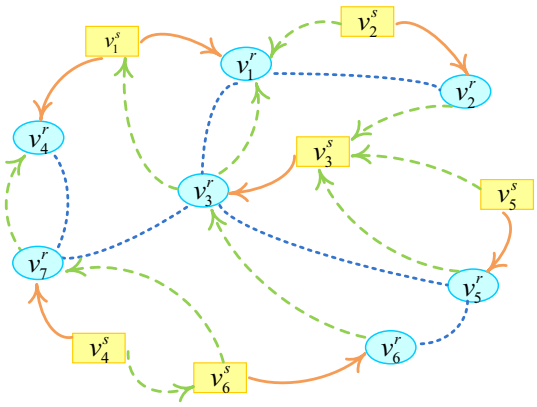



**Fig.3** S-R credibility network

**图 3** S-R 可信度网络

在开始计算这条记录的可信度值之前，要为网络中的每一个顶点赋予一个初始可信度值。构造初始可信度值的方法有很多种(比如 Site 顶点取 PR 值，Record 顶点为 0)，但是这不是本文所关注的重点，本文的重点在于可信度值传播算法，因此这部分不展开叙述。关于顶点初始可信度值的设置我们会在实验中有进一步的说明。

本文用传播迭代法为 S-R 网络中的每一个节点计算可信度值。一个顶点的可信度值可以通过有向边以及无向边传递给其相邻顶点。这种传播行为在整个 S-R 网络的所有节点之间进行。然后，将此 S-R 网络中所有顶点的可信度值以合适的方式合并起来，计算出整个网络的全局可信度值，就是这条 Deep Web 中的记录的可信度值。

S-R 可信度网络可以应用于 Deep Web 中的各个领域。只是在不同的领域中，可信度传播方法以及全局可信度值的计算可能会有所不同。

以下，将着重讨论一种在 Deep Web 中，利用 S-R 可信度网络来计算某一条数据记录的可信度的方法：C-Rank。

### 4.2 S-R可信度网络局部可信度值计算

在构造 S-R 可信度网络之后，本文利用可信度传播的思想，利用顶点的出度，在 S-R 可信度网络中迭代计算可信度传播率，从而得到 Record 顶点的局部可信度值。

**定义 2（局部可信度值）**在 S-R 可信度网络中，每一个顶点的可信度值称为局部可信度值。

不同于以往的网络，本文所提出的 S-R 网络包含两类顶点、三类边。因此不同类型边的可信度的传播率也不相同。

如图 4 所示，图中的黄色实线箭头表示内部链接边 $e^i$，绿色虚线箭头表示外部链接边 $e^o$，蓝色虚线边表示实体识别边 $e^r$。

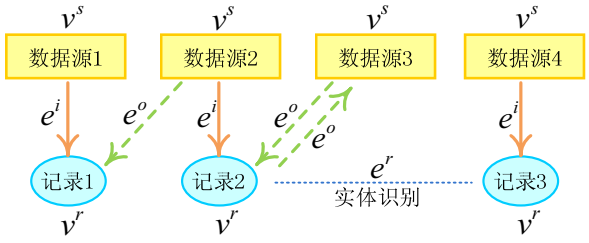



**Fig.4** Credibility propagation graph

**图 4** 可信度传播数据图

三种类型的边拥有不同的含义，因而拥有不同的可信度传播率。我们为这三种类型的边设置三种传播率类型：

(1)内部链接边的传播率类型为 $\alpha(e_G^i)$；

(2)外部链接边的传播率类型为 $\beta(e_G^o)$；

(3)实体识别边的传播率类型为 $\gamma(e_G^r)$。

对于 S-R 图中的每一条边，首先判断它是属于哪种类型的边，根据传播率类型和相应的边的出度，再计算这条边的实际传播率。例如，如果边 $e_k^i=(u\to v)$ 是 S-R 图中的第 k 条边，它属于 $e^i$ 边，其传播率类型为 $\alpha(e_G^i)$，那么这条边的传播率为：

$$\alpha(e_k^i)=\begin{cases}\dfrac{\alpha(e_G^i)}{OutDeg(u,e_G^i)}, & OutDeg(u,e_G^i)>0\\ 0, & OutDeg(u,e_G^i)=0\end{cases} \quad (1)$$

其中，$OutDeg(u,e_G^i)$ 表示顶点 $u$ 发出的 $e^i$ 类型边的数量。类似的，我们可以求出属于 $e^o$ 类型边以及属于 $e^r$ 类型边的传播率。

需要注意的是，由于 $e^r=(u,v)$ 是一个无向边，我们将其看成是两个单向边，即 $u\to v$ 的边以及 $v\to u$ 的边，然后分别处理这两条边。

然后，我们利用 PageRank[11]以及 ObjectRank[12]的基本思想，根据 S-R 网络的特点加以改进，经过 n 次可信度值传播后的顶点可信度值，来迭代计算第 n+1 次传播后顶点的可信度值：

$$R_{loc}^{(n+1)}=dAR_{loc}^{(n)}+\frac{(1-d)}{|S|}s \quad (2)$$

其中，$R_{loc}^{(n)}$ 表示经过 n 次可信度值传播后的顶点可信度值向量，它其中的每一个元素 $r_{loc}^{(n)}(v_i)$ 表示经过 n 次可信度值传播后的顶点 $v_i$ 的局部可信度值；$A$ 是一个 $m\times m$ 的矩阵，$A$ 中的每一个元素 $A_{ij}=\alpha(e^i)$；d 为阻尼系数；$S$ 是所有顶点的一个任意子集；而 $s=(s_0,\ldots,s_i,\ldots,s_m)^T$ 为基本向量，如果一个顶点 $v_i\in S$，则 $s_i$ 等于 1，否则为 0。

根据文献[13]，这样的迭代计算必然会趋向收敛。设定一个任意小的向量 $\varepsilon$，当符合如下条件时，迭代停止：

$$|R_{loc}^{(n+1)}-R_{loc}^{(n)}|<\varepsilon \quad (3)$$

至此，我们为 S-R 可信度网络中的每一个顶点求得了它的局部可信度值。

### 4.3 S-R可信度网络全局可信度值计算

正如前文所提到的，整个 S-R 可信度网络实际上是关于一条记录而构建的一个网络，因此，当我们求得所有顶点的局部可信度值后，需要综合考虑所有 Record 顶点的局部可信度值，求得全局可信度值。

**定义 3（全局可信度值）**整个 S-R 可信度网络的可信度值，它代表了此 S-R 网络对应的招聘记录在 Web 上的总体可信度值。

为了计算整个网络的全局可信度值，我们采用几种不同的方法综合 S-R 网络中所有 Record 顶点的局部可信度值。

**方法一：求和法**。将所有 Record 顶点的局部可信度值求算术和。如公式（4）所示：

$$C\text{-}Rank(S\text{-}R)=\sum r_{loc}(v_i) \quad (4)$$

这种方法反映了一条记录重复出现次数越多，可信度值越高的情况。

**方法二：最大值法**。在所有的 Record 顶点中，取局部可信度值最大的作为这条记录的全局可信度值。如公式（5）所示：

$$C\text{-}Rank(S\text{-}R)=\max\{r_{loc}(v_i)\,|\,i=1,\ldots,m\} \quad (5)$$

最大值法的合理之处在于：同一条记录在不同网站上多次出现，如果有一次能够被证明可信度是非常高的，那么这条记录应该也是非常可信的。

**方法三：顶点加权法**。上述两种方法虽然在一定程度上可行，但是无法正确处理虚假信息恶意转载以及中小型公司的招聘信息可信度问题。

根据问题分析中所提到的关于信息可信度的观察，在 Deep Web 中一条记录的可信度与其发布的数据源的可信度以及重复出现的次数均相关。因此，在根据局部可信度值计算全局可信度值时，需要考虑不同的 Record 顶点具有不同的权值。

$$\omega(v_i)=\sum InDeg(v_i^{\{i,o,r\}})\times\{\overline{\alpha}(e_{ji}^{i}),\overline{\beta}(e_{ji}^{o}),\overline{\gamma}(e_{ji}^{r})\mid\exists e_{ji}=v_j\rightarrow v_i\}$$
$$+\sum r(v_j^{s})\times\{\alpha(e_{ji}^{i}),\beta(e_{ji}^{o}),\gamma(e_{ji}^{r})\mid\exists e_{ji}=v_j\rightarrow v_i\wedge v_j\in v^{s}\} \quad (6)$$

在公式(6)中，$InDeg(v_i^{\{i,o,r\}})$表示指向顶点$v_i$的$e^i$、$e^o$以及$e^r$类型的边的数量(对于$e^r$边，将其看成两个单向边)；$\overline{\alpha}(e_{ji}^{i}),\overline{\beta}(e_{ji}^{o}),\overline{\gamma}(e_{ji}^{r})$表示顶点$v_i$属于$\alpha(e_G^i)$、$\beta(e_G^o)$和$\gamma(e_G^r)$传播率类型的边的平均传播率值；$r(v_j^s)$表示与顶点$v_i$相邻的 Site 顶点的可信度值；而$\alpha(e_{ji}^{i}),\beta(e_{ji}^{o}),\gamma(e_{ji}^{r})$表示的是 Record 顶点$v_i$与 Site 顶点$v_j$的边的传播率。

综合考虑了顶点的入度以及相邻 Site 顶点的可信度值，我们可以得到 S-R 图中的每一个 Record 顶点的权值，使用公式(7)将顶点的权值进行标准化：

$$\omega_{nor}(v_i)=\frac{\omega(v_i)}{\max\{\omega(v_k),k=0,\ldots,m\}} \quad (7)$$

在得到所有 Record 顶点的局部可信度值及其对应的权重后，我们通过加权计算，得到整个 S-R 网络的可信度值：

$$C\text{-}Rank(S\text{-}R)=\sum\omega_{nor}(v_i)r_{loc}(v_i) \quad (8)$$

顶点加权值法可以有效地防止恶意的信息发布者在多个网站上重复发布、转载同一条虚假招聘信息。如果恶意发布者想通过这种方法使记录的出现次数增多来达到骗取一定的可信度值的目的，也不可能获得较高的全局可信度值。同时，这种方法也能够为中小型公司的招聘信息计算出一个符合实际情况的可信度值。

## 5 实验

目前，在 Deep Web 上，没有为数据记录计算可信度值的相关工作，因此本文实验的主要目的在于验证 C-Rank 方法的有效性与合理性。在包含 1000 条招聘记录的数据集上，使用 C-Rank 方法为每一条记录进行了可信度值的计算，并邀请了用户为实验效果进行打分评价。在本节中给出了实验结果及用户反馈情况。

### 5.1 数据集

本实验的数据集需要从工作信息领域获取。我们使用 Jobtong[14]（一个工作信息领域的数据集成原型系统）从 Deep Web 数据源中爬取招聘信息记录。我们首先用 Jobtong 取到任意的 900 条不同的招聘记录，然后手工加入 100 条不可信的招聘记录。这 100 条招聘记录是我们从网上手动收集的，是由网友揭发出来的骗简历的虚假招聘信息，或完全名不符实的“皮包公司”。

该数据集的构造方法使它具有如下特点：第一，实验者知道哪些记录必定是不可信的，可以验证可信度值计算方法是否能够有效地将这些记录鉴别出来；第二，所有记录均从结构化数据源中获取，信息完备；第三，数据集中的所有招聘记录均为随机选取。基于这三个特点，该数据集可保证实验结果的客观性。

### 5.2 实验设置

在实验开始之前，使用 4.1 节中的方法为每一条招聘记录自动构造它的 S-R 可信度网络。Site 顶点的初始可信度值设为该网站的 PR 值。Record 顶点的初始可信度值为 0。如果某个网站是属于论坛、博客、信息集成系统这类允许自由发布信息的开放式平台，则将其初始可信度置为 PR 值与一个折扣系数的乘积。折扣系数可以由统计实验得到，也可以由人工根据经验设定。折扣系数是一个[0,1]区间内的值。

### 5.3 实验分析

我们使用 4.2 节中的方法，为这 1000 条记录的 S-R 网络中的每个顶点计算了局部可信度值，然后使用顶点加权法（经实验证明，求和法与最大值法的效果不如顶点加权法）计算记录的全局可信度值，并将其标准化为[0,1]区间内的值。实验结果如图 5 所示，横坐标表示每一条招聘记录的唯一标号，纵坐标表示这条记录对应的可信度值。

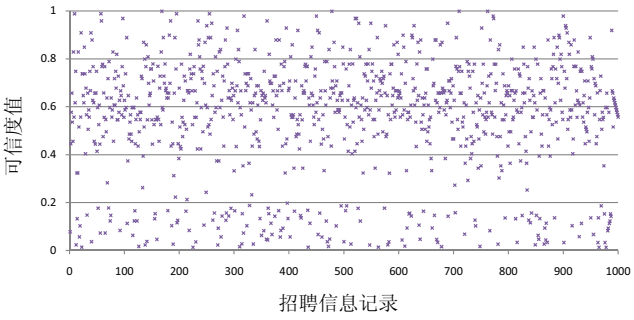



**Fig.5** Scatter diagram of records credibility values distribution

**图 5** 记录可信度分布散点图

经过确认，我们发现，在实验中人工加入的 100 条不可信记录全都分布在可信度值为[0, 0.2)的区间内。同时，从图 5 可以看出，可信度值的分布呈现一定的聚集效果。为了进一步量化这种聚集规律，我们将可信度值平均分为 5 个区间：[0, 0.2), [0.2, 0.4),……, [0.8, 1]。这 5 个区间代表 5 个可信度等级，等级越高越可信。

可信度等级 1～5 级中包含的记录数量如图 6 所示，分别为 156 条、45 条、294 条、390 条，115 条。

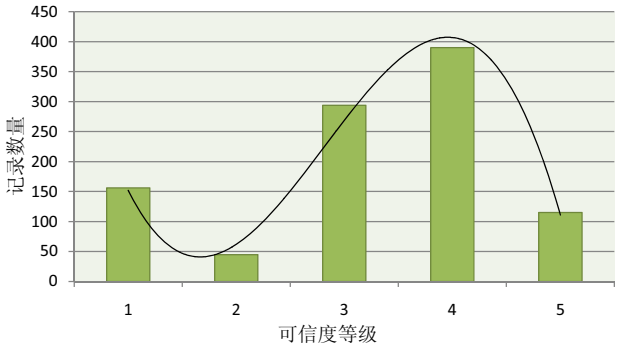



**Fig.6** Record numbers of different credibility levels

**图 6** 不同可信度等级的记录数

从图 6 中的趋势线可以看出，不可信记录全部被识别出来（经人工确认，第一等级[0, 0.2)区间中的 156 条记录完全包含了人工加入的 100 条不可信记录）。并且，第二等级的记录数相对较少，是整个趋势线的波谷，这说明了 C-Rank 方法能够有效地将可信记录与不可信记录区分开来。而第二等级到第五等级之间的记录数目呈现正态分布的趋势，这是符合真实的网络环境中招聘记录的可信度的分布情况的，这也说明了我们的方法能够对招聘记录的可信度给出合理而合适的评价值。

为了验证本文方法对于可信度的评价值是否准确，我们邀请了 10 名用户对实验结果进行评价。系统随机为每一名用户分配 100 条招聘记录（包括招聘职位、内容描述、记录来源等信息），以及用 C-Rank 方法为其计算的一个可信度值。用户根据人工判断，对每条记录的可信度值是否合理进行评价，评价的内容包括：合理、偏高、偏低、不合理。用户评价结果如图 7 所示。

从图 7 中的结果可以看出，用户认为使用 C-Rank 方法计算招聘信息可信度具有较高的合理性。10 名用户评价的平均合理率达到 94.2%，而认为偏高或偏低的记录只占 1.8%及 2.7%，认为不合理的记录仅占 1.3%。经过确认，用户认为偏高、偏低或不合理的记录多是容易导致主观性较强的一些招聘信息，例如一名用户以私人邮箱为公司招聘技术开发人员，但内容比较详细，所以用户仅凭内容分析无法做出正确判断。从这个意义上来说，本文提出的方法计算出的可信度值也能够为这些用户提供有价值的参考信息。

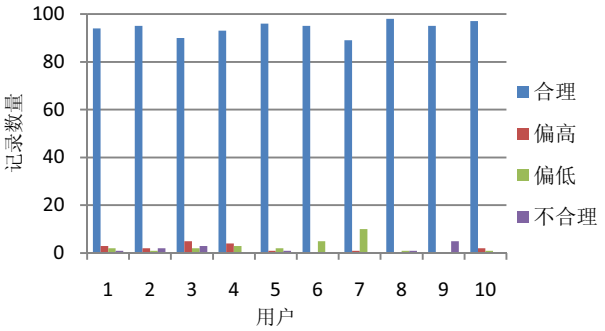



**Fig.7** Users' feedbacks for records' credibility values

**图 7** 用户对于记录可信度分值的评价结果

## 6 结束语

本文提出了一种基于传播机制的 Deep Web 数据记录可信度评估方法。该方法为每一条招聘记录构造一个 S-R 可信度网络，利用 S-R 网络中顶点的出度以及可信度传播机制为每一个 Record 顶点计算一个局部可信度值，然后使用入度及相邻 Site 顶点的可信度值为每一个 Record 顶点计算权重，求得关于这条记录的 S-R 网络的整体可信度值，作为该记录的全局可信度值。通过在 Deep Web 工作信息领域的实验证明，C-Rank 方法能够合理而有效地评价招聘记录的可信度，从而达到甄别虚假信息，为用户推荐可信的、更有价值的招聘记录的目的。根据用户对于计算结果的评价，证明了用 C-Rank 方法计算出的记录可信度值符合实际情况。

Deep Web 上的数据记录可信度问题具有很强的现实意义。本文所提出的解决办法具有实际操作性，且易于扩展到其它领域。

**References:**

[1] He B, Patel M, Zhang Z, et al. Accessing the Deep Web: A Survey[J]. Communications of the ACM (CACM), 2007, 50(5): 94-101.

[2] Alfarez A-R, Hailes S. Relying On Trust to Find Reliable Information[C]//Proceedings of International Symposium on Database, Web and Cooperative Systems (DWA-COS'99), 1999. [2009-04-25].http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.7934

[3] Lee R, Kitayama D, Sumiya K. Web-based Evidence Excavation to Explore the Authenticity of Local Events[C]//Proceedings of WICOW 2008, California: ACM, 2008: 63-66.

[4] Kawai Y, Fujita Y, Kumamoto T. Using a Sentiment Map for Visualizing Credibility of News Sites on the Web[C]//Proceedings of WICOW 2008, California: ACM, 2008:53-58.

[5] Wanas N, El-Saban M, Ashour H, et al. Automatic Scoring of Online Discussion Posts[C]//Proceedings of WICOW 2008, California: ACM, 2008:19-25.
[6] Staddon J, Chow R. Detecting Reviewer Bias through Web-Based Association Mining[C]//Proceedings of WICOW 2008, California: ACM, 2008:5-9.
[7] Kamvar S D, Schlosser M T, Garcia-Molina H. The EigenTrust Algorithm for Reputation Management in P2P Networks[C]//Proceedings of the 12th World Wide Web Conference, Budapest: ACM, 2003: 640-651.
[8] Ziegler C-N, Lausen G. Propagation Models for Trust and Distrust in Social Networks[J]. Information Systems Frontiers, 2005, 7:4/5: 337–358.
[9] Guha R, Kumar R, Raghavan P, et al. Propagation of Trust and Distrust[C]//Proceedings of the 13th World Wide Web Conference, New York: ACM, 2004: 403-412.
[10] Elmagarmid A K, Ipeirotis P G, Verykios V S. Duplicate Record Detection: A Survey[J]. IEEE Transactions on knowledge and Data Engineering (TKDE), 2007, 19(1): 1-16.
[11] Brin S, Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine[C]//Proceedings of the 7th World Wide Web Conference. Brisbane: ACM, 1998: 107-117.
[12] Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based Keyword Search in Databases[C]// Proceedings of the 30th International Conference on Very Large Data Bases(VLDB'04), Toronto: ACM, 2004: 564-575.
[13] Motwani R, Raghavan P. Randomized Algorithms[M]. United Kingdom: Cambridge University Press, 1995.
[14] Jobtong. [2009-4-25]. http://www.jobtong.cn

## 作者简介

**AI Jing** was born in 1985. She received the B.S. degree from Renmin University of China in 2007, and now is a M.S. candidate in Computer Application and Technology at Renmin University. Her research interest focuses on Web data management.

艾静(1985- )，女，北京人，2007 年于中国人民大学获计算机科学与技术专业工学学士学位，目前是中国人民大学计算机应用技术专业硕士研究生，主要研究领域为 Web 数据管理。

**WANG Zhong-yuan** was born in 1985. He received the B.S. degree from Renmin University of China in 2007, and now is a M.S. candidate in Computer Application and Technology at Renmin University. His research interest focuses on Web data management.

王仲远(1985- )，男，福建仙游人，2007 年于中国人民大学获计算机科学与技术专业工学学士学位，目前是中国人民大学计算机应用技术专业硕士研究生，主要研究领域为 Web 数据管理。

**MENG Xiao-feng** was born in 1964. He received the Ph.D degree in Computer Application and Technology from Institute of Computing Technology Chinese Academy of Sciences in 1999. Now he is a professor and doctoral supervisor at Renmin University. His research interests include web data management, native XML databases, mobile data management, etc.

孟小峰(1964- ), 男, 河北邯郸人, 1999 年于中国科学院获计算机应用技术专业工学博士学位, 目前是中国人民大学教授, 博士生导师, 主要研究领域为 Web 数据管理、XML 数据库、移动数据管理。

In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009), page 315-324, November 2-6, 2009, Hong Kong, China

# Efficient Algorithms for Approximate Member Extraction Using Signature-based Inverted Lists

Jiaheng Lu Jialong Han Xiaofeng Meng
School of Information, Renmin University of China Beijing 100872, China
Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education, China
jiahenglu@ruc.edu.cn jialonghan@gmail.com xfmeng@ruc.edu.cn

## ABSTRACT

We study the problem of approximate membership extraction (AME), i.e., how to efficiently extract substrings in a text document that approximately match some strings in a given dictionary. This problem is important in a variety of applications such as named entity recognition and data cleaning. We solve this problem in two steps. In the first step, for each substring in the text, we filter away the strings in the dictionary that are very different from the substring. In the second step, each candidate string is verified to decide whether the substring should be extracted. We develop an incremental algorithm using signature-based inverted lists to minimize the duplicate list-scan operations of overlapping windows in the text. Our experimental study of the proposed algorithms on real and synthetic datasets showed that our solutions significantly outperform existing methods in the literature.

## Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems – *Textual Databases*

## General Terms

Algorithms

## Keywords

Approximate Member Extraction, Filtration-verification, Approximate string matching, Incremental Computation

## 1. INTRODUCTION

In this paper we study the problem of finding substrings in a text document M that approximately match (e.g. having similarity scores above a given threshold $\delta$) some strings in a given dictionary R of strings. This problem, called AME (short for ***A**pproximate **M**ember **E**xtraction*), arises in many applications, as illustrated by the following examples.

*Named Entity Recognition*: With a given document, we want to locate pre-defined entities such as person names, conference names, and company names. We want this extraction to be *approximate*, i.e., we allow slight mismatches in the substrings. For instance, as shown in Figure 1, suppose we have a collection of conference names, such as "ACM SIGMOD" and "CIKM Conference."


*CIKM'09*, November 2-6, 2009, Hong Kong, China.

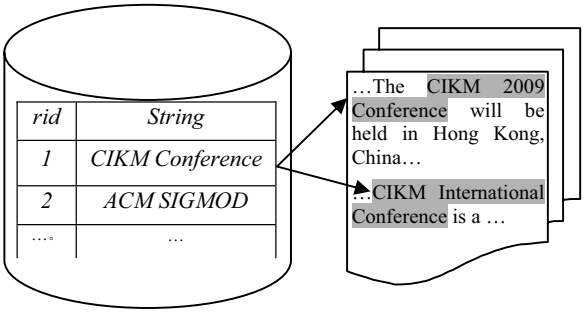




**Figure 1. Approximate member extraction**

We want to extract all conference names from a given document. We want to find matches such as "CIKM 2009 Conference" and "CIKM International Conference", even though they do not match the string "CIKM Conference" in the dictionary exactly.

*Data Cleaning*: Documents in many applications could be "dirty" when it contains inconsistencies. Often we need to clean the inconsistencies. We need to perform data cleaning and integration by identifying the dirty words based on an existing dictionary.

One naive way to solve the AME problem is to enumerate each substring m of M and check if m matches strings in R approximately. Several algorithms have been proposed for doing the checking efficiently using two steps. In the first step, we filter the dictionary strings that are very different from m. In the second step, we compute the similarity between the remaining candidate strings and the string m to verify its approximate membership.

There are two methods to do the filtration in the first step. One is based on an *inverted index* on the dictionary R. In this method, strings are regarded as collections of *tokens*. For each token, the index stores the ids of the strings that include this token. For a given string m, we can find candidate similar strings in the dictionary by accessing the lists of the tokens in m and finding those string ids that have enough occurrences on the lists. The second method is based on *signatures generation* [1, 4]. This method focuses on exploring signature schemes that convert a string to a set of hash codes and filtering irrelevant strings using their signatures.

Recently, researchers have been trying to combine these two methods. For instance, Wang et al. propose a method called NGPP [13] to solve the AME problem by assuming the edit-distance similarity function. They shift and extend the partitions of dictionary strings to obtain an inverted index on all the *partition variations* (an implicit signature). Then by generating the neighborhood of partitions of document substrings and probing the index, the algorithm filters most irrelevant strings and performs verifications for the remaining strings. Chakrabarti et al. [2] studied how to solve the AME problem with other similarity score

functions such as Jaccard coefficient using a method called ISH (Inverted Signature-based Hashtable), which encodes a string and its signatures into a 0-1 matrix. After generating a 0-1 matrix by computing the "bitwise-or" of all small matrices encoded with dictionary strings, ISH converts the problem of searching possible evidence into finding a certain 0-1 submatrix in the big matrix.

Our work in this paper is motivated by the following observations: (1) The ISH method generates signatures for strings and prune strings that share signatures with a total weight under a certain lower bound (see Para 8, Page 4 of [2]). Unfortunately, the lower bound may be too high, causing false negatives (see Example 1 for a counter-example). (2) The filtration processing of ISH involves an NP-Complete problem requiring finding *solid submatrix* (containing only 1) from a given 0-1 matrix under certain constraints. We give the reduction proof in this paper from an NP-Complete problem: *Balanced Complete Bipartite Subgraph* (see Appendix). Due to the intrinsic complexity, [2] solves this problem by a simple heuristics, which significantly increases the number of false positives, and thus incurs numerous I/Os in the verification phase and deteriorates overall performance. But our research in this paper shows that we can avoid the NP-complete complexity to efficiently solve AME problem by adopting a novel index structure, which effectively controls the number of false positives and consequently improves the overall performance. (3) NGPP[13] and ISH[2] both require that $\delta$ is provided when preprocessing the data and generating an index for the filter. Once the threshold is fixed, these filters no longer support queries with thresholds other than $\delta$ unless the index is re-generated.

EXAMPLE 1. *(Example to illustrate the false negative in [2]) Under Jaccard similarity measurement, with weight assignment {(a, 6), (b, 3.52), (c, 3.51), (d, 3.49), (e, 3.48), (f, 1)}, string r="bcde" matches m="abef" under the similarity threshold $\delta$ =1/3. But from the following table we see that [2] incorrectly determines that m and r are impossible to match under the threshold 1/3, though their real similarity is 1/3.*

| *String* | *Signatures (k=3)* | $\tau$ *value* | *Shared Signatures* | *Lower Bound Required by ISH Filter* |
|---|---|---|---|---|
| *r="bcde"* | *{b,c,d}* | $\tau$ (r)=*1.187* | *{(b, 3.52)}* | *≥3.67* |
| *m="abef"* | *{a,b,e}* | $\tau$ (m)=*3.67* | | |

We summarize our contributions as follows:

- We adopt the prefix filtering technique [4], and propose new theorems which are all strictly proved and well applied as filtering conditions in our method. These theorems convert approximate matching to prefix signatures sharing, and give a tighter bound. More importantly, this filtering technique brings no false negatives.
- We utilize an inverted-list-based filtering index SIL and propose corresponding algorithm called EvSCAN. By performing inverted list scanning instead of introducing matrix-based combinatorial problem, EvSCAN naturally avoids solving NP-Complete problem. We also apply incremental optimizations on EvSCAN and propose EvITER, which effectively reduces the duplicate list-scanning operations when the substring window shifts over a large document M. Compared with prior solutions, our method produces far less false evidences, thus achieves better filtration-verification balance and consequently improves the overall performance significantly.
- We modify our SIL to answer queries with dynamic similarity thresholds. Specifically, once initialized with a lower bound $\delta_0$, our filter works for any query with a threshold $\delta \geq \delta_0$. However previous filters only allow static similarity threshold and need to be re-initialized (i.e., the indices in filter has to be generated again) once the query threshold is changed.
- We provide detailed and accurate experimental results to support our argument. We show that SIL is significantly efficient than ISH, both in filtering power and overall running time, and explain the intrinsic reason by analyzing their runtime statistics in details. We also compare EvSCAN and EvITER under various situations, and show that our incremental optimization is effective.

The following sections are organized as follows:

Section 2 formally defines AME and gives some preliminaries. Section 3 complements the theory of prefix filtering and applies them to build the SIL structure and EvSCAN algorithm. Section 4 introduces the EvITER incremental algorithm. Section 5 shows how to support dynamic thresholds. Section 6 reports our experimental results. Section 7 discusses related work. Section 8 is the conclusion of our study.

# 2. PRELIMINARIES AND PROBLEM STATEMENT

## 2.1 Some Notations and Problem Statement

In this paper, we use the letter "t" to denote a token, and the other lower-case letters are used for strings. We regard strings as sets of tokens. For any token t, we denote wt(t) as the weight of t (e.g. IDF weight), and for any string s, we define wt(s) as $\sum_{t \in s} wt(t)$, i.e. the sum of weights of all tokens in s. Based on the above notations we define Jaccard similarity of any strings $s_1$ and $s_2$ as:

$$J(s_1, s_2) = \frac{wt(s_1 \cap s_2)}{wt(s_1 \cup s_2)}.$$

EXAMPLE 2. *(Reusing the string and weight configuration of Example 1)Under weighted Jaccard similarity, the two strings m="abef", r="bcde" have a similarity of wt({b, e})/wt({a, b, c, d, e, f})=(3.52+3.48)/(6+3.52+3.51+3.49+3.48+1)=1/3.*

With all notations introduced, we present the formal description of AME as follow:

***Problem Statement***

*Given a dictionary R of strings and a similarity threshold $\delta \in [0,1]$, then a query M is submitted. Here M represents a relatively long string (e.g. a text file). The task of AME is to extract all M's substrings m, such that there exists some $r \in R$ satisfying $Sim(m,r) \geq \delta$.*

Sometimes we are only interested in substrings whose length is up to a length threshold L, so we may as well require that $|m| \leq L$. Here any extracted m is called approximate member of R, and corresponding evidence for r in R. Note that our algorithms we present later can handle any similarity function that satisfies the following properties:

- *Sim*(m,r) is symmetric, i.e., *Sim*(m,r) = *Sim*(r,m).

- $Sim(m,r) \le \frac{wt(m \cap r)}{wt(m)}$.

(Symmetrically we have $Sim(m,r) \le \frac{wt(m \cap r)}{wt(r)}$.)

The first property is natural, and the second is also shared by many similarity functions [2]. For example, for any m and r, we have $J(m,r) = \frac{wt(m \cap r)}{wt(m \cup r)} \le \frac{wt(m \cap r)}{wt(m)}$ because wt(m ∪ r) ≥wt(m), so the Jaccard similarity is capable of serving as a similarity function in our discussion.

## 2.2 The Filtration-Verification Framework

To efficiently solve AME and other related problems, researchers have been designing methods following two phases of filtration and verification [1, 2, 4, 7, 13]. In the AME problem, employing this framework usually requires building an indexing structure for the dictionary R. Recall that for each approximate member m extracted, we define the string r in R that is similar enough with m to be m's evidence. Thus, the task of extracting all approximate members from M can be simply reduced to determining whether there exists any evidence for each substring of M, and filtration-verification is actually referred to as evidence filtration and evidence verification.

Generally, our foundation of filtration is based on some necessary condition (denoted as NC) of our matching criterion Sim≥ $\delta$, that is, if some candidate evidence is real evidence, it must satisfy NC. With the dictionary R given offline, we build an index that quickly recommends for a query m ALL potential evidence that meets NC, so that true evidence is never missed. Then the evidence is verified against the actual matching criterion to determine whether the string m is a true approximate member.

Note that NC plays a key role in our whole framework. It ensures the correctness of the whole algorithm. Moreover, it determines how balanced our framework is. We can evaluate it through:

- How powerful is it? That is, does it eliminate as much false evidence as possible?
- Is it easy-going? That is, can we build a quick index to test it at low cost?

There is a tradeoff in the cost between two phases of filtration-verification. For example, with all dictionary strings being potentially possible evidence, the most easy-going filtration approach for them is obviously "no filtration", which leads to a brute-force method of scanning the whole dictionary. On the other hand, if we try to make our filtration the most powerful, i.e., it produces no false positive and achieves the smallest verification time; it will be expensive to perform such filtering. As a matter of fact, we need to obtain a beneficial compromise between two phases. In the following sections, we will intentionally highlight this issue through our theoretical and experimental analysis.

## 2.3 The K-Signature Scheme

The k-signature scheme is first demonstrated by Chakrabarti et al. in [2]. It is an extension of the prefix signature idea [1]. Here we briefly introduce some key definitions as follows:

DEFINITION 1. *For a given string s and similarity threshold $\delta$, we sort all its tokens by their weight in descending order (if two tokens appear with the same weight, we sort them lexicographically), and choose the first few tokens to get a subset Sig(s), such that $\tau$(s)=wt(Sig(s))-(1- $\delta$)wt(s)≥0. We call Sig(s) a prefix signature set of string s. For convenience we call it signature set from now on.*

EXAMPLE 3. *(Reusing the string and weight configuration of Example 1) Let $\delta$=0.6, then {a,b} is a signature set of m because $\tau$(m)= wt({a,b})-(1-0.6)*wt({a,b,e,f})=9.52-5.6=3.92≥0.*

Based on its definition, the following is some facts about prefix signature set:

- For any string s, Sig(s) always exists because we can let Sig(s)=s, thus making $\tau$(s)=wt(s)-(1- $\delta$)wt(s)= $\delta$*wt(s)≥0, i.e., we choose itself to be its signature set.
- A string may have more than one signature set with different sizes. For instance in Example 3, we see that m has another different signature set {a,b} besides itself.

One way to ensure the uniqueness of signature set is to use a parameter k to determine which set we choose for a string s, where k is a positive integer. We call this unique signature set k-signature set, denoted as $Sig_k(s)$. When k is fixed, we select $Sig_k(s)$ among all available signature sets as follows:

- If all signature sets' sizes are bigger than k, choose the smallest one.
- Else if there is any signature set whose size is exactly k, choose it.
- Else, choose the largest one, i.e. the string itself.

In some special case, when we set k=1, we call the derived signature scheme as *min-signature* scheme. When k is set to be ∞, we will get the string itself to be its signature set.

EXAMPLE 4. (*Following the configurations of Example 3) We list the different Sig(m) under different k settings as below:*

| *K* | *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|---|
| $Sig_k(m)$ | *{a}* | *{a,b}* | *{a,b,e}* | *{a,b,e,f}* | *{a,b,e,f}* |

**Table 1. Signature sets controlled by K**

Note that parameter k has nothing to do with the requirement of signature set, so we may randomly set k for our filter, and even choose different k for different strings. As a matter of fact, if we regard signature set as a compression of information in strings, then k is the parameter for global compression rate tuning. That is, k only influences the performance of our filter. We will further discuss its role in Section 3.3.

With the above description, we find that for any string s, the signature set is actually controlled by $\delta$ and k, thus should be written as $Sig_k(s, \delta)$. In fact, the parameter k is fixed to SIL and before Section 5 we also consider that $\delta$ is static, so when the context is clear, we still use Sig(s) to denote the signature set of s. In Section 5, we use Sig(s, $\delta$) because we start discussing how to support dynamic $\delta$ during query processing.

# 3. FILTRATION VIA SIGNATURE-BASED INVERTED LISTS

In this subsection, we show some nice properties about the signature set which will be used in our algorithms. For instance, if

String m and r meet the matching condition Sim(m,r) ≥ δ , m must contain at least one of r's signatures. This is apparently a necessary condition for matching and can be utilized to build a filter. In the following discussion, we'll further explore the property of signature sets and show that there are better filtering conditions.

## 3.1 The Property of Signature Sets

LEMMA 1. For any string s and its selected signatures, we use minsigwt(s) = $\min_{t \in sig(s)}\{wt(t)\}$ to denote the smallest weight of all s's signature tokens. Then for any string s:

*A token t ∈Sig(s) if t ∈s and wt(t) ≥minsigwt(s).*

This is apparent because the selected signature tokens must have larger weight than unselected tokens.

LEMMA 2. (PROPERTY OF PREFIX SIGNATURES). *For any string m and r, if minsigwt(m) ≥ minsigwt(r), then wt(Sig(m)∩Sig(r)) ≥ wt(Sig(m))- wt(m-r). Here m-r refers to the minus set of m and r.*

***Proof*:** *We transform it to an equivalent form as wt(m-r) ≥ wt(Sig(m)-Sig(r)), and prove this by showing that Sig(m)-Sig(r) is a subset of m-r .For any t ∈Sig(m)-Sig(r) , we have t ∈Sig(m) , so t ∈m .*

*Now we prove that t∉r.*

*Suppose that t ∈r. because t ∈Sig(m), we know that wt(t) ≥minsigwt(m) ≥ minsigwt(r) . From t ∈r , wt(t) ≥minsigwt(r) and Lemma 1, we conclude that t ∈Sig(r). This is inconsistent with the fact that t ∈Sig(m)-Sig(r) . So t∉r .*

*From t∉r and t ∈m , it's obvious t ∈m-r. So Sig(m)-Sig(r) is a subset of m-r , easily leading to the result that wt(m-r) ≥ wt(Sig(m)-Sig(r)) .* ■

In *Lemma 2*, we illustrate the signature set overlapping relationship between matching strings. Intuitively this inequality condition is tighter than other conditions proposed in [1], and we believe this property is also useful in other researches involving prefix signatures. However, the set minus operator seems costly to handle, so we need to make this condition more easy-going. We solve this by introducing *Theorem 1* as follow:

THEOREM 1 (FILTERING CONDITION). *For any m and r that satisfy Sim(m,r) ≥ δ, wt(Sig(m)∩Sig(r)) ≥ min{ τ(m), τ(r) }.*

***Proof*:** *If minsigwt(m)) ≥ minsigwt(r), according to Lemma 2, we have wt(Sig(m)∩Sig(r)) ≥ wt(Sig(m))- wt(m-m∩r)= wt(Sig(m))-wt(m)+wt(m∩r) ≥wt(Sig(m))- wt(m)+ δ*wt(m ∪r) ≥wt(Sig(m))-wt(m)+ δ*wt(m)= wt(Sig(m))-(1- δ)wt(m)= τ(m) ≥ min{ τ(m), τ(r) }.*

*If minsigwt(m) ≤ minsigwt(r), based on the symmetry of Sim() we have the same result. So in conclusion we have wt(Sig(m)∩ Sig(r)) ≥ min{ τ(m), τ(r) }.* ■

EXAMPLE 5. *(Following the configurations of Example 3) Suppose k=2, we have Sig(m)={a,b}, τ(m)= 3.92 and Sig(r) ={b,c}, τ(r)=(3.52+3.51)-0.4*14=1.43. So wt(Sig(m)∩Sig(r))= wt({b})=3.52 ≥ min{ τ(m), τ(r) } = min{3.92, 1.43}=1.43.*

We obtain the foundation of filtration phase of SIL so far. It's easy to discover that when the threshold τ(r) is computed offline, this filtering condition only involves the signature set of all strings,

ALGORITHM 1: **BuildSIL**( R, δ , k)

```
1  for each r∈R do
2      Sig←GenSig(r, δ , k);
       /*The function GenSig(r, δ , k) generates signature
       for r under k-signature scheme.                  */
3      for each t∈Sig do
4          list[t]=list[t]∪{rid(r)};//insert rid of r into list
5  return list;
```

indicating the fact that the time and space requirement of our filter is tightly related to the average signature set size of all strings in the dictionary R, which is controlled by the parameter k. Moreover, different k provides different filtering conditions. Among them we need to decide which one to choose.

## 3.2 Filtration via SIL

Since for any matched m and r, their signature sets overlaps, it's easy to come up with the idea of building an inverted index structure for the dictionary R, and filtering by merging inverted lists and accumulating weights. After this index is built up offline, by visiting list[t], we can quickly retrieve for any token t a list of *rid* of all r in R, who contains token t as a signature. Due to the fact that these lists only involves signatures of all strings in R, we call this index SIL in short for *signature-based inverted lists*, and *Algorithm 1* below shows the method to generate an SIL index.

EXAMPLE 6. *Suppose we have a dictionary R={r[1]="SIL's filtering power", r[2]="the power of filtering by SIL"}, the weight of each tokens are {<SIL's, 5>, <filtering, 4>, <power, 3.5>, <SIL, 3>, <by 2>, <the, 1>, <of, 1>}. We set k=1 and δ =0.55, then we have each strings' signature set in Table 2 and the SIL built as Figure 2.*

| *rid* | *String* | *Signature Set* |
|---|---|---|
| *1* | *"SIL's filtering power"* | *{"SIL's","filtering"}* |
| *2* | *"the power of filtering by SIL"* | *{"filtering","power"}* |

**Table 2. Signature sets of R's strings**

| *Signature* | → | *String rids* |
|---|---|---|
| "SIL's" | → | (1) |
| "filtering" | → | (1), (2) |
| "power" | → | (2) |

| *rid* | wt(r) | τ (r) |
|---|---|---|
| 1 | 12.5 | 3.375 |
| 2 | 14.5 | 0.975 |

**Figure 2. SIL and additional information for dictionary R**

When a string m's membership needs to be checked, we simply compute the signature set of m, denoted as $\{t_1,t_2\ldots t_n\}$. Then we scan all n lists that is indexed by $list[t_1]$, $list[t_2]\ldots list[t_n]$, while aggregating the weight of $t_i$ to all *rid* whose record contains $t_i$ as one of its signature. To record the aggregated weight, we may use an array Sum[] for convenience or a hash table to save memory space. With all lists scanned, the aggregated weight of any *rid* is exactly the value of wt(Sig(m)∩Sig(r)). In fact, if any *rid* appears satisfying the filtering condition of *Theorem 1*, i.e. with an aggregated weight larger than min{ τ (m), τ (r)}, we can store it for later verification to determine whether it is the one that makes m a true member.

ALGORITHM 2: **EvSCAN**( M, $\delta$, k, L)

```
1   ResultSet←Φ;// for storing approximate members
2   for each m of M's substrings (|m| ≤ L) do
3       Sig←GenSig(m, δ, k);
4       Initialize Sum[];//for weight aggregating
5       CandSet←Φ;// for storing candidate evidence
6       for each t∈Sig do
7           for each rid ∈list[t] do
8               Sum[rid]+=wt(rid);//aggregating weight
9               if Sum[rid]>=min{ τ(m), τ(rid)} then
10                  CandSet←CandSet∪{rid};
11      for each rid∈CandSet do//verification
12          if Sim(m, r(rid)) ≥ δ then//true evidence found
13              ResultSet ←ResultSet∪{m};
14              break;
15  return ResultSet;
```

Note that $\tau$ (r) and wt(r) for any dictionary string r is computed in the signature generating step of Algorithm 1 and they are stored in the main memory for the later use of our algorithms (See Figure 2).

## 3.3 Additional Discussion

In the above discussion, one may notice that we didn't involve the parameter k. This again proves the fact that with any assigned k, our algorithm will run correctly. Since the signature set is a compression of information in a string, we will certainly get more information if we choose a relatively large k, through which we can target potential matching evidences to a smaller scope, thus reducing the cost of verifying these evidences.

However, larger k causes longer inverted list length, i.e. more cost on targeting possible evidences. For instance, if we set $k = \infty$, that is, for all strings s, we set Sig(s) to be s itself, and $\tau$ (s)= $\delta$ *wt(s), we interestingly find that our method degrades into a common inverted-list based solution. Chakrabarti [2] first analyzed this problem and showed that k=3 is good on average situation, which is also proved in our experimental study.

# 4. OPTIMIZATION BY PROGRESSIVE COMPUTATION

## 4.1 Reducing Duplicate Computations

Although EvSCAN algorithm efficiently checks the approximate membership of each single substring m in a document M, it ignores the overlapping between shifting substring windows and consequently takes a lot of time on duplicate computations. Another way to solve AME is to reduce this problem to set similarity join, which is already well studied by researchers [1, 4, 6]. In set similarity join, we are given SA and SB - two columns of sets, a similarity function Sim, and a threshold $\delta$. The task of set similarity join is to join the two columns, where the joining condition is Sim(SA, SB)≥ $\delta$. In AME, if we set SA=R, SB={all substrings of M}, run set similarity join between SA and SB and project the result set along SB, we will get the result of AME. Though the problem of set similarity join is explored and optimized in many papers, this method still doesn't notice the fact that the records in SB are quite similar with each other – they are substrings of a long text.

In this section, based on the above observations, we believe that the unique property of AME should be exploited separately, and optimized method could be designed accordingly. So we study the incremental property of Theorem 1, and demonstrate Theorem 2, in purpose of decreasing the duplicate list-scanning when examining all substrings of M.

## 4.2 Optimization by Progressive Computation

Assume $m \oplus t$ denoting the string which we get by concatenating token t to the tail of string m. Consider the process of checking m and $m \oplus t$: we compute the signature set of m and verify the condition in Theorem 1, then we do the same job for $m \oplus t$. Intuitively the signature set of m and $m \oplus t$ are much alike. We observe that: if some r cannot match m and does not contain t, it is not likely to match $m \oplus t$. Before we formalize our intuition into new theorem and algorithm, we introduce *Evidence Superset* by the lemma and definition below:

LEMMA 3. *For any m and r that satisfy Sim(m,r) ≥ $\delta$, wt(m∩Sig(r)) ⩾min{$\delta$*wt(m), $\tau$(r)}}.*

This lemma is an inference of Theorem 1. Recall that in previous sections we mention that Theorem 1 remains correct even if we set different k for different strings, Lemma 3 is in fact obtained by setting $k=\infty$ for m in Theorem 1 (so Sig(m) is replaced by m and $\tau$ (m) by $\delta$ *wt(m)). With Lemma 3 we define *Evidence Superset* for any query substring m as follow:

DEFINITION 2. *Suppose $\delta$ and k are fixed, for any string m, let ES(m)= {r ∈R| wt(m ∩Sig(r)) ⩾min{$\delta$*wt(m), $\tau$(r)}}, we call ES(m) an Evidence Superset of m. Based on Lemma 3 it's obvious that any true evidence for m must be contained in ES(m).*

From its definition, we see that ES(m) is useful since any evidence matching m will be included in ES(m). If we can efficiently compute ES(m) for any substring m, we can further filter elements in ES(m) to pick out all true evidences and check m's approximate membership. Our intuition is formalized below.

LEMMA 4. *For any string m and token t, if a dictionary $r \notin ES(m)$ and $t \notin Sig(r)$, then $r \notin ES(m \oplus t)$.*

***Proof:*** *We prove $r \notin ES(m \oplus t)$ by showing that $wt((m \oplus t) \cap Sig(r)) < min\{\delta * wt(m \oplus t), \tau(r)\}$: $wt((m \oplus t) \cap Sig(r)) = wt(m \cap Sig(r))$ (Because $t \notin Sig(r)$) $< min\{\delta * wt(m), \tau(r)\}$(because $r \notin ES(m)) < min\{\delta * wt(m \oplus t), \tau(r)\}$.*

■

This lemma states a fact that if a dictionary string is far from being evidence of current substring m and it is not a signature of the coming token t, then it cannot be evidence when the substring window moves to $m \oplus t$. However this lemma still cannot serve as a method for efficiently computing ES(m), we further generalize Lemma 4 to obtain Theorem 2 to demonstrate the incremental property of ES(m).

ALGORITHM 3: **EvITER**( M={t[1],t[2],…,t[n]}, δ , k, L)

```
1   ResultSet← Φ ;// for storing approximate members
2   for i =1 to n do
3     Initialize Sum[];
            /*Sum[rid].s1 records wt(Sig(m)∩Sig(r(rid))) and
                Sum[rid].s2 for wt(m ∩ Sig(r(rid)))        */
4     LastES← Φ ;//for iterating ES
5     for j=i to min{n,i+L-1} do     //current m= t[i]…t[j]
6       ES← Φ ;
7       CandSet← Φ ;// for storing candidate evidence
8       update Sig(m) and maintain sum[];
                    /*because m and Sig(m) changes          */
9       for each rid ∈list[t] ∪ LastES do
10        if Sum[rid].s2≥min{ δ *wt(m), τ (r(rid))} then
11          ES←ES ∪ {rid};
12          if Sum[rid].s1≥min{ τ (m), τ (rid)} then
              CandSet←CandSet ∪ {rid};
13      VerifyAllCandidate();      //verification
14      LastES←ES;     //iteration for the next window
15  return ResultSet;
```

THEOREM 2. (INCREMENTAL PROPERTY). *Suppose* $\delta$*, k are fixed, it holds for any string m and token t that* $ES(m \oplus t) \subseteq ES(m) \cup list[t]$.

Notice that the approach we build SIL determines that $t \in Sig(r)$ means $r \in list[t]$, so Theorem 2 is obviously based on *Lemma 4*. This theorem indicates an efficient iterative approach of maintaining ES() for the varying substring when the right boundary of the substring windows moves by a token. That is, we check all elements in $ES(m) \cup list[t]$ and pick out proper ones into $ES(m \oplus t)$. Note that this process requires maintaining another field recording $wt(m \cap Sig(r))$ in the summing table sum[], so it can be combined with the process of filtering by SIL.

Now we get a new algorithm of incrementally checking all substrings, that is, we fix the left boundary of the substring window and shift the other boundary to the right. While the substring varies we iteratively maintain corresponding ES() to filter and verify all evidence in it. Figure 3 shows this iteration process, with an instance of $M=t_1 \oplus t_2$. For convenience, we denote this incremental filtering algorithm as EvITER (Evidence Iterating), while EcSCAN in Section 3.2 is short for Evidence Scanning.

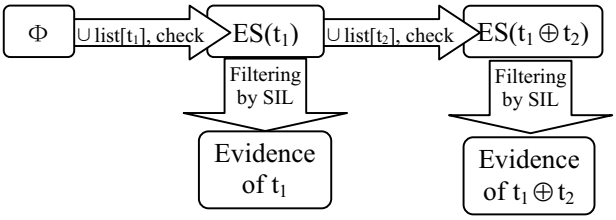



**Figure 3. Flow of dictionary strings in EvITER**

## 4.3 Combining Other Filtering Conditions

In Algorithm 1 and 2, we use "VerifyAllCandidate()" to denote the process of verification, by which one determine if m is really a true member. Since not all r's in CandSet are the ones that make m a true member, we may as well make "VerifyAllCandidate()" a small filter-verification process, via introducing some other simple yet effective filtering conditions. The following is exactly one of such conditions we want:

*If* $Sim(m,r) \geq \delta$*, then* $wt(m) * \delta \leq wt(r) \leq wt(m)/\delta$.

Intuitively, under the non-weighted situation, this condition states that if any two strings have too much difference in length, they are not likely to match each other. To apply this filtering condition, we need only store in memory the weight value of all strings in R. By checking this condition for all r in CandSet, we can quickly narrow our scope to fewer possible r, thus avoiding more disk accessing and making computation more efficient.

## 4.4 Algorithm Analysis

In this section we analyze the time and memory cost of our algorithms, and demonstrate the advantages of EvITER over EvSCAN. The notations to be used are listed in Table 3.

| | |
|---|---|
| $\|M\|$ | The length of text used as the input of AME |
| $\|R\|$ | Dictionary size |
| $L_r$ | Average length of dictionary strings |
| $L_m$ | Average substring length |
| $L_{list}$ | Average length of inverted lists |
| $\delta$ | Similarity threshold |
| E | Total number of evidence that passes the filter |
| $C_v$ | Time cost of verifying an evidence (including disk accessing and similarity score computing) |

**Table 3. Some notations in cost analysis**

In addition to the above notations, we make two assumptions in order to simplify our discussion: (1) the length of all strings is longer than k. (2) all tokens have the same weight (e.g. 1). Therefore, we obtain an upper bound for the signature set size of any string.

LEMMA 5. *Suppose k and* $\delta$ *are fixed, then for any string m with length L,* $|Sig(m)| \leq max\{k,(1-\delta)L\}$.

***Proof:*** *Because for any string m,* $|Sig(m)|=wt(Sig(m)) \geq (1-\delta) * wt(m) = (1-\delta)L$*, the smallest signature set size is* $(1-\delta)L$.

If $(1-\delta)L \leq k$*, from the definition of k-signature set we choose the first k tokens as signatures, so* $|Sig(m)|=k$*, else we choose the smallest signature set of size* $(1-\delta)L$*. Therefore we have* $|Sig(m)| \leq max\{k,(1-\delta)L\}$. ■

With above assumptions and lemmas, the memory cost of our SIL can be expressed as

$$MEMCOST=|R|max\{k, (1-\delta)L_r\}.$$

This equation is correct since every string in R produces $max\{k, (1-\delta)L_r\}$ nodes at most in the inverted lists. To estimate the time cost of our algorithms, we introduce another lemma as below:

LEMMA 6. *For each substring m, the number of inverted lists that EvSCAN and EvITER scan are respectively |Sig(m)| and 2.*

***Proof (Outline):*** *It's obvious for EvSCAN and we only give proof for EvITER. Consider the moment we finish checking m and prepare for $m \oplus t$, we have to scan list[t] once to iterate ES(m) into ES($m \oplus t$). Moreover, it's possible that t replace some signature token t' to be a new signature, so we must scan list[t'] to maintain the summing table sum[] for next iteration.* ■

Therefore, the filtration cost of EvSCAN and EvITER are respectively $|M|L_mL_{list}max\{k, (1-\delta)L_m\}$ and $2|M|L_mL_{list}$. The verification cost of EvSCAN can be estimated as $EC_v$, while that of EvITER is $E(1+C_v)$ because of the O(1) evidence iteration cost for every evidence. In summary we have

$$TIMECOST(EvSCAN)= |M|L_mL_{list}\, max\{k, (1-\delta)L_m\}+ EC_v,$$

$$TIMECOST(EvITER)= 2|M|L_mL_{list} + E(1+C_v).$$

Here we see that EvITER avoids scanning some lists by introducing the cost of evidence iteration, so it may have some advantage when k is large or E is small, which will be demonstrated by our experimental results later.

# 5. SUPPORTING DYNAMIC SIMILARITY THRESHOLDS

## 5.1 The Static Threshold Problem

In the above discussion we talk about how to perform AME with a *static* similarity threshold, where the filter can be denoted as $F(R, \delta_0)$, given a dictionary R and a fixed threshold $\delta_0$, meaning that the threshold $\delta_0$ is undesirably static. If users want to submit a query with other thresholds, the filter has to be re-initialized. This apparently leads to much inconvenience in practice. In this section we will focus on this issue and show that with a little modification, our SIL can handle this problem well. Note that in this section, the notation Sig(s) is replaced by Sig(s, $\delta$) to add a dynamic threshold $\delta$.

## 5.2 Solution and Analysis

In our SIL algorithm, we observe that the problem of static threshold is caused by the definition of prefix signatures. Recall that the prefix signatures Sig(s) for string s is a prefix subset of s that satisfies $wt(Sig(s))\geq(1-\delta)wt(s)$. Therefore, with different $\delta$ we need different number of signatures to build various filters.

Another observation is that, under min-signature schema, for any string s, if a token t is selected as a signature under some threshold, it will also be in the signature set of s when the threshold gets lower. That is, if we initialize the filter at a relatively low threshold $\delta_0$, when a query comes with a higher threshold $\delta \geq \delta_0$, those rids whose string contains t as a signature should be included in some nodes on list[t] of the current filter. For simplicity we call these nodes *active nodes*. All we need is to discriminate *active nodes*, and use them to perform filtration.

We propose Theorem 3 to provide a way to discriminate active nodes as follow:

THEOREM 3 (SUFFICIENT AND NECESSARY CONDITION OF MIN-SIGNATURE). *Under min-signature schema, for any string $s=\{t_1,t_2...t_n\}$, where $wt(t_1) \geq wt(t_2) \geq...\geq wt(t_n)$, let $U_i=1-(wt(t_1)+ wt(t_2)+...+ wt(t_i))/wt(s)$ for any $i\geq1$ and $U_0=0$. We have the following conclusion:*

*$t_i \in Sig(s, \delta)$ if and only if $\delta \in [0,U_{i-1})$.*

Because $U_{i-1}$ can be computed in the filter-constructing phase, in every node of all inverted lists, we add a field to record $U_{i-1}$ in order to test the condition $\delta \in [0,U_{i-1})$ to decide whether this is an active node. Moreover, we can sort all nodes in a list in descending order of corresponding $U_{i-1}$. In this way, for any threshold $\delta$, all active node in a list must form a prefix of the list. Therefore, we may stop our scan once an inactive node is found, by which we avoid scanning the whole list and enhance the performance.

Note that this modification should only be applied under the min-signature scheme. In fact, in Section 6 we will show that under most cases, min-signature schema is enough to serve as a good choice.

EXAMPLE 8. *(adopting the configuration in Example 6) Suppose we initialize the modified filter with min-signature schema (k=1) and $\delta_0=0.55$ as Figure 4. We have a query with $\delta=0.7$, then all nodes in Figure 4 that is circled out become active nodes and should be scanned.*

| *Signature* | → | *String rids and U* | |
|---|---|---|---|
| "SIL's" | → | (1, 1.0) | |
| "filtering" | → | (2, 1.0), | (1, 0.6) |
| "power" | → | (2, 0.725) | |

**Figure 4. Active nodes when δ=0.7**

Comparing with the origin SIL, we see that applying this modification only requires a little more space and additional sorting in the filter-building phase. For queries with various similarity thresholds, the modified SIL successfully solves the static threshold problem, without visiting any additional list nodes or trading query performance.

# 6. EXPERIMENTAL STUDY

## 6.1 Experimental Settings

The following filters and algorithms are evaluated in this section:

**ISH** (abbreviated from ***I**nverted **S**ignature-based **H**ashtable*) is a filter proposed in [2], whose idea is to optimize the query range of length filtering. In our experiment, we set the inverted hashtable length b to be 11 (8 is enough according to [2]).

**EvSCAN** on **SIL** is our proposed algorithm, which filters by scanning the **S**ignature-based **I**nverted **L**ists. **EvITER** is an optimized version of **EvSCAN**, which aims at reducing unnecessary list scanning.


**Figure 5. Comparison between ISH and SIL (k=3, |R|=1000, L=10)**

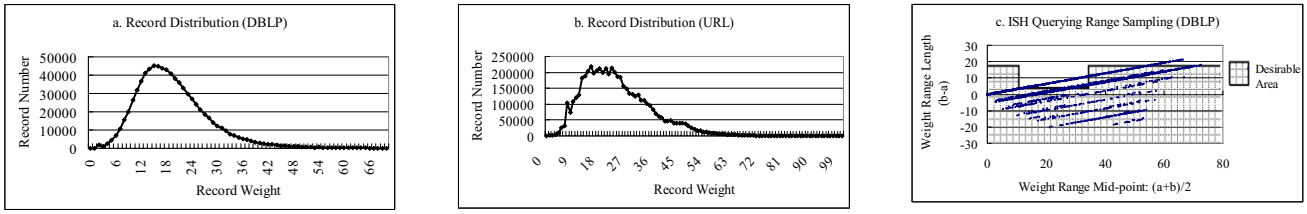



**Figure 6. Some Statistics for ISH (k=3, L=10, δ =0.85)**

We ran our experiments on the following two datasets:

**DBLP**: It includes paper titles downloaded from the DBLP site. We extracted 274,788 paper titles with a total size 17.8MB as the dictionary. The query text to this dictionary is 40 web pages from CiteSeer, each containing the title, abstract, citation, etc. of a random paper. Tokens are separated by spaces and punctuations.

**URL**: The dictionary includes the first 1,838,973 URLs from an URL dataset. The query text is 40 text files, each containing 50 random URLs from the rest of the dataset. Tokens are separated by slashes.

On both datasets, standard IDF weight [14] is applied, and all tests were conducted under the weighted Jaccard similarity measurement. We mainly judge the performance of all filters via analyzing the filtering power and overall running time of them. We evaluate the power of filters by the candidate evidence they produce since the size of candidate evidences has a great influence on overall performance.

## 6.2 Comparing with ISH

We compared our approach EvSCAN on SIL with ISH in this section. We first performed experiments on DBLP data (dictionary size: 274,788 records), and our results show that ISH produced a large amount of candidate evidences and disk-accessing, thus spending much time on verification and could not terminate in one hour. Therefore, we had to reduce the size of dictionary using the first 1000 records in DBLP.

We explain this result through reviewing the filtering approach of ISH: for every query substring m, ISH optimizes the existing length filtering condition mentioned in Section 4.3, and uses a new range (denoted as [a,b], $a = \delta * wt(m)$, $b \leq wt(m)/\delta$ ) as the SQL querying condition at evidence record retrieving phase.

In Figure 6, we show the record distribution of two datasets across the weight axis (see sub-figure (a) and (b)), where all records distribute densely, with at most 50k and on average 10k in a unit length of weight range (DBLP dataset). This implies that it's unwise to retrieve all evidence whose weight is in certain range, unless we can make our query range desirably small or far from those regions with crowded records.

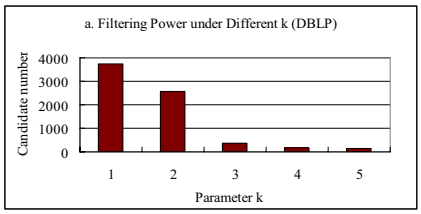
In Figure 6(c), we sampled the weight range [a,b] (each represented by a *characteristic point* (mid-point, length) or ( (a+b)/2, b-a) ) from 2,608 SQL queries ISH launches when processing a random webpage. We also flagged some areas as "desirable area", where "desirable queries" appear (queries possessing small [a,b] ranges or avoiding the most frequent weight of all records, i.e. x-coordinate of the peak in Figure 6(a) and (b) ).

We see in this figure that the mid-points of all sampled ranges vary averagely from 0 to 70, which include the most frequent weight in both datasets (15 for Figure 6(a) and 20 for (b)). Moreover, though ISH sometimes successfully confirms of no matching (denoted by ranges with negative length in Figure 6(c) ), the range length in many queries is not desirably short. Therefore, there exist too many queries, whose *characteristic point* is located far from our "desirable area". These insufficiently optimized queries lead to tons of I/Os and verification computations, thus deteriorating the overall performance of ISH.

## 6.3 Effects of parameters on SIL

Based on our analysis in the above sections, the performance of SIL is mainly influenced by the following aspects: the "compressing rate" parameter k, dictionary size, query text length, similarity threshold, and substring length threshold. We run EvITER on different parameter settings and record the results, from which we have the following observations:

- The parameter k is tightly related to every aspect of the filter. Larger k means stronger filtering power (Figure 7(a)), less verification time, and larger filter size. However, on average situation when $\delta$ =0.85 and L=10, the result shows that the most competitive k value is among 1, 2, and 3 (see Figure 7(b)), which makes the two phases of filtration and verification more balanced. This supports our discussion about compromising between the two phases.
- Though the performance of SIL depends much on the inherent property of the dataset (e.g. query texts about chemical science certainly run quickly on our URL dictionary because the number of word matching is expected to be small), it still exhibits a linear increase in running time under different dictionary size and query text length, which is expected in our cost analysis in Section 4.4.



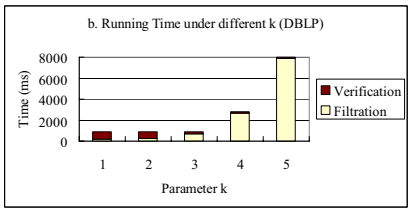




**Figure 7. Performance under different k (L=10, δ =0.85)**

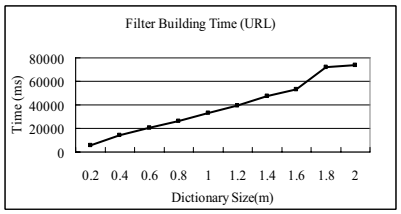
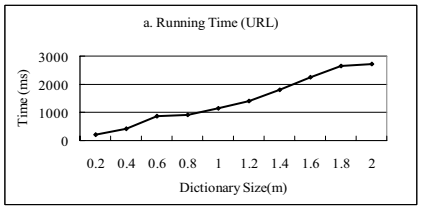
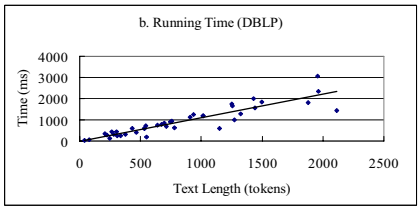



**Figure 8. Filter building (k=3, δ =0.85)**



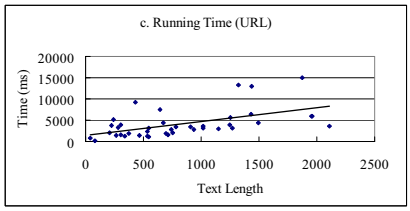
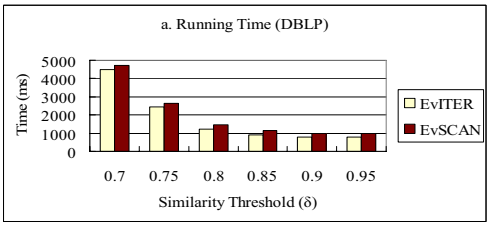




**Figure 9. Performance under different data size (k=3, L=10, δ =0.85)**



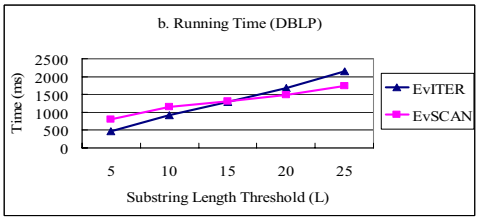




**Figure 10. Performance under different thresholds (k=3)**

## 6.4 Comparison between EvSCAN & EvITER

Besides the experimental analysis on ISH and SIL, we also performed a comparison between the two algorithms we propose: EvSCAN and EvITER. Instead of dictionary size and query length, in this subsection we mainly focus on the two varying threshold parameters: similarity threshold and substring length threshold.

Through the comparison we found that:

- The similarity threshold significantly affects the time consumed by our two algorithms. When δ decreases from 0.95 to 0.7, both algorithms requires 4 times more running time. This is easily explained by our filtering condition: when δ decreases, the signature sets of most strings get larger to make the chance of signature overlapping increase, while $\tau(.)$ for most strings leaves almost unchanged. Thus, it's easier for evidence to pass the filter.
- When L=10, k=3 and δ =0.85, EvITER shows a performance increase of about 25% over EvSCAN, this is because EvITER reduces the operation of scanning an inverted list, by iterating from one evidence set to another. When the similarity threshold is high and the candidate evidence set is small, the advantage of EvITER will be more obvious.
- To our surprise, when L is above 15, EvITER is gradually outperformed by EvSCAN. This result is not expected by us. We carefully studied this issue and find the reason: because the candidate set ES(m) we maintain (recall in Section 4.2) tends to get bigger when m is longer, therefore EvITER will spends more time iterating it as larger L allows longer m to be checked.

# 7. RELATED WORK

In the literature "approximate string matching" refers to the problem of finding a pattern string approximately in a text. There have been many studies on this problem. See [9] for an excellent survey. The problem of AME is different: searching in a long text to approximately match a string from a dictionary. In addition, AME is also different to the problem of text document indexing (finding dictionary documents approximately containing a query string) and string similarity joins (identifying approximate matching string pairs, each from one of two columns of strings).

To measure the similarity of a pair of strings, generally all similarity functions can be categorized as token-based and character-based, depending on what they regard strings as: sets of tokens, or sequences of characters.

The token-based AME problem, as discussed in this paper, can be straightforwardly reduced to set similarity join [1, 4, 6, 10]. Paper [4] discussed the framework and implements of a primitive operator SSJoin for performing similarity joins, on which a variety of similarity functions can be applied. Paper [1] solved the similarity join problem by converting set-based similarity distance into hamming distance between binary vectors, and studying the number of shared segments of two divided vectors. In [2], Chakrabarti et al. proposed a 0-1 matrix-based AME filter. In this paper we showed that their approach touches upon a NPC decision problem, whose intractability we briefly prove in the Appendix.

As a complement to the token-based approach, the character-based approximate string-matching problem has been well studied by researchers [9]. Early methods handling the edit distance constraints mostly work on the relationship of edit distance and gram sharing [12]. Due to the dilemma in choosing gram length, [8] proposes VGRAM, namely variable-length gram

to address the problem. For non-gram-based approaches, Wang et al. uses inverted lists to index the neighborhood of dictionary strings, and enhances previous neighborhood generation methods by reducing the upper bound of the neighborhood size [13].

Another line of related work is on inverted list merging, because the filtration phase needs inverted list processing. In [7], this problem is formalized into T-occurrence problem, and three efficient algorithms are proposed. T-occurrence problem requires that the threshold T should be independent from any list nodes, which is not satisfied by our method (our threshold $\min\{\tau(m),\tau(r)\}$ varies with the rid information in list nodes), the list processing technique in [7] is orthogonal to our solution here, and can be used (by some modification) on SIL index in a complementary manner.

# 8. CONCLUSION

In this paper, we studied the AME problem (Approximate Member Extraction). Under the framework of filtration-verification, we analyzed the issue of trading between the two phases, and proposed a new filtering condition and corresponding filter called SIL. Then we designed two algorithms for SIL: EvSCAN and its incrementally optimized version EvITER, which saves the cost of scanning some inverted lists by progressively maintaining a candidate evidence set of the current substrings. We also addressed the static threshold problem of previous filters, and gave a solution for it on our SIL. Finally we reported the performance of our filtering algorithms through theoretical and experimental analysis.

# 9. APPENDIX

Theorem 1 in [2] provides a method of filtering by converting it to a decision problem about 0-1 matrices. Here, we give the proof about its intractability. For convenience we call it *Constrained Solid Submatrix* problem and describe it as below:

*(Constrained Solid Submatrix problem) Given a 0-1 matrix A, whose size is p*q and two weight functions $w_1(i)$ $(1 \le i \le p)$ and $w_2(j)$ $(1 \le j \le q)$, we need to determine if there exsits a subset $I=\{ i_1, i_2, \ldots, i_r \}$ from the rows and a subset $J=\{ j_1, j_2, \ldots, j_c \}$ from the columns such that for any $i' \in I$ and $j' \in J$, $A[i'][j']=1$, and $w_1(i_1)+ w_1(i_2)+\ldots+ w_1(i_r) \ge \delta$, $w_2(j_1)+ w_2(j_2)+\ldots+ w_2(j_c) \ge \tau$ ( $\delta$ and $\tau$ are two given thresholds)*

THEOREM 4 *(intractability of Constrained Solid Submatrix problem) Constrained Solid Submatrix problem is NP-Complete.*

***Proof Outline:*** *We prove by reducing to Balanced Complete Bipartite Subgraph problem, which requires finding a K*K complete bipartite subgraph in a given bipartite $B=<V_1 \cup V_2, E>$. It is already proven to be NP-Complete (see page 196 in [5]).*

*For any instance of the Balanced Complete Bipartite Subgraph problem, let $w_1(i)=1, w_2(j)=1$, $\delta = \tau = K$, and construct a $|V_1|*|V_2|$ 0-1 matrix, whose elements are assigned as follow:*

$$A[i][j]=\begin{cases} 1 & \text{If the i-th vertex in } V_1 \text{ and j-th in } V_2 \text{ are adjacent.} \\ 0 & \text{Otherwise.} \end{cases}$$

*Then we can show that A has a constrained solid submatrix if and only if the corresponding Balanced Complete Bipartite Subgraph problem has a solution. This concludes the reduction.*

# ACKOWLEDGEMENT

The authors are grateful to Prof. Chen Li for his suggestions to motivate this work and thanks anonymous reviewers for their constructive comments. This research was partially supported by the grants from 863 National High-Tech Research and Development Plan of China (No: 2009AA01Z133, 2007AA01Z155, 2009AA011904), National Science Foundation of China (NSFC) under the number (No.60833005) and Key Project in Ministry of Education (No: 109004).

# REFERENCES

[1] A. Arasu, V. Ganti, R. Kaushik. Efficient exact set-similarity joins. In *VLDB, pages 918-929, 2006.*

[2] K. Chakrabarti, S. Chaudhuri, V. Ganti, D. Xin. An efficient filter for approximate membership checking. In *SIGMOD Conference*, 2008.

[3] A. Chandel, P. C. Nagesh, and S. Sarawagi. Efficient batch top-k search for dictionary-based entity recognition. In *ICDE, page 28, 2006.*

[4] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE, page 5, 2006.*

[5] M.R.Garey and D.S.Johnson. Computers and Intractability: Guidance to the Theory of NP-Completeness.

[6] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB, pages 491-500, 2001.*

[7] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *ICDE, pages 257–266, 2008.*

[8] C. Li, B,Wang, X. Yang, VGRAM: Improving performance of approximate queries on string collections using variable length grams. In *VLDB 2007.*

[9] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv., 33(1):31–88, 2001.*

[10] S. Sarawagi, A.Kirpal, Efficient set joins on similarity predicates. In *SIGMOD Conference, 2004.*

[11] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society* Technical Committee on Data Engineering, 24(4):35-43, 2001.

[12] E. Sutinen and J. Tarhio. On using q-grams locations in approximate string matching. In ESA, pages 327-340, 1995.

[13] W. Wang, C. Xiao, X. Lin, C. Zhang. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD Conference, 2009.*

[14] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann, 1999.*

[15] A. C. Yao and F. F. Yao. Dictionary loop-up with small errors. In *CPM*, pages 387-394, 1995.

IEEE International Conference on Data Engineering (ICDE 2009),
page 604-615, March, 29- April, 3, 2009, Shanghai, China

# Space-Constrained Gram-Based Indexing for Efficient Approximate String Search

Alexander Behm[1] Shengyue Ji[1] Chen Li[1] Jiaheng Lu[2]

[1] Department of Computer Science, University of California, Irvine, CA 92697
[2] Key Laboratory of Data Engineering and Knowledge Engineering, Renmin University of China, China
{abehm,shengyuj,chenli}@ics.uci.edu, jiahenglu@gmail.com

***Abstract*— Answering approximate queries on string collections is important in applications such as data cleaning, query relaxation, and spell checking, where inconsistencies and errors exist in user queries as well as data. Many existing algorithms use gram-based inverted-list indexing structures to answer approximate string queries. These indexing structures are "notoriously" large compared to the size of their original string collection. In this paper, we study how to reduce the size of such an indexing structure to a given amount of space, while retaining efficient query processing. We first study how to adopt existing inverted-list compression techniques to solve our problem. Then, we propose two novel approaches for achieving the goal: one is based on discarding gram lists, and one is based on combining correlated lists. They are both orthogonal to existing compression techniques, exploit a unique property of our setting, and offer new opportunities for improving query performance. For each approach we analyze its effect on query performance and develop algorithms for wisely choosing lists to discard or combine. Our extensive experiments on real data sets show that our approaches provide applications the flexibility in deciding the tradeoff between query performance and indexing size, and can outperform existing compression techniques. An interesting and surprising finding is that while we can reduce the index size significantly (up to 60% reduction) with tolerable performance penalties, for 20-40% reductions we can even improve query performance compared to original indexes.**

## I. Introduction

Many information systems need to support approximate string queries: given a collection of textual strings, such as person names, telephone numbers, and addresses, find the strings in the collection that are similar to a given query string. The following are a few applications. In record linkage, we often need to find from a table those records that are similar to a given query string that could represent the same real-world entity, even though they have slightly different representations, such as `spielberg` versus `spielburg`. In Web search, many search engines provide the "Did you mean" feature, which can benefit from the capability of finding keywords similar to a keyword in a search query. Other information systems such as Oracle and Lucene also support approximate string queries on relational tables or documents.

Various functions can be used to measure the similarity between strings, such as edit distance (a.k.a. Levenshtein distance), Jaccard similarity, and cosine similarity. Many algorithms are developed using the idea of "grams" of strings. A *q-gram* of a string is a substring of length $q$ that can be used as a signature for the string. For example, the 2-grams of the string `bingo` are `bi`, `in`, `ng`, and `go`. These algorithms rely on an index of inverted lists of grams for a collection of strings to support queries on this collection. Intuitively, we decompose each string in the collection to grams, and build an inverted list for each gram, which contains the id of the strings with this gram. For instance, Fig. 1 shows a collection of 5 strings and the corresponding inverted lists of their 2-grams.

| id | string |
|---|---|
| 1 | bingo |
| 2 | bitingin |
| 3 | biting |
| 4 | boing |
| 5 | going |

(a) Strings.

| gram | | string ids |
|---|---|---|
| bi | → | 1, 2, 3 |
| bo | → | 4 |
| gi | → | 2 |
| go | → | 1, 5 |
| in | → | 1, 2, 3, 4, 5 |
| it | → | 2, 3 |
| ng | → | 1, 2, 3, 4, 5 |
| oi | → | 4, 5 |
| ti | → | 2, 3 |

(b) Inverted lists.

Fig. 1. Strings and their inverted lists of 2-grams.

The algorithms answer a query using the following observation: if a string $r$ in the collection is similar enough to the query string, then $r$ should share a certain number of common grams with the query string. Therefore, we decompose the query string to grams, and locate the corresponding inverted lists in the index. We find those string ids that appear at least a certain number of times on these lists, and these candidates are post-processed to remove the false positives.

**Motivation**: These gram-based inverted-list indexing structures are "notorious" for their large size relative to the size of their original string data. This large index size causes problems for applications. For example, many systems require a very high real-time performance to answer a query. This requirement is especially important for those applications adopting a Web-based service model. Consider online spell checkers used by email services such as Gmail, Hotmail, and Yahoo! Mail, which have millions of online users. They need to process many user queries each second. There is a big difference between a 10ms response time versus a 20ms response time, since the former means a throughput of 50 queries per second (QPS), while the latter means 20 QPS. Such a high-performance requirement can be met only if the index is in memory. In another scenario, consider the case where

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.32

these algorithms are implemented inside a database system, which can only allocate a limited amount of memory for the inverted-list index, since there can be many other tasks in the database system that also need memory. In both scenarios, it is very critical to reduce the index size as much as we can to meet a given space constraint.

**Contributions**: In this paper we study how to reduce the size of such index structures, while still maintaining a high query performance. In Section III we study how to adopt existing inverted-list compression techniques to our setting [31]. That is, we partition an inverted list into fixed-size segments and compress each segment with a word-aligned integer coding scheme. To support fast random access to the compressed lists, we can use synchronization points [24] at each segment, and cache decompressed segments to improve query performance. Most of these compression techniques were proposed in the context of information retrieval, in which conjunctive keyword queries are prevalent. In order to ensure correctness, lossless compression techniques are usually required in this setting.

The setting of approximate string search is unique in that a candidate result needs to occur at least a *certain number* of times among all the inverted lists, and not necessarily on all the inverted lists. We exploit this unique property to develop two novel approaches for achieving the goal. The first approach is based on the idea of discarding some of the lists. We study several technical challenges that arise naturally in this approach (Section IV). One issue is how to compute a new lower bound on the number of common grams (whose lists are not discarded) shared by two similar strings, the formula of which becomes technically interesting. Another question is how to decide lists to discard by considering their effects on query performance. In developing a cost-based algorithm for selecting lists to discard, we need to solve several interesting problems related to estimating the different pieces of time in answering a query. For instance, one of the problems is to estimate the number of candidates that share certain number of common grams with the query. We develop a novel algorithm for efficiently and accurately estimating this number. We also develop several optimization techniques to improve the performance of this algorithm for selecting lists to discard.

The second approach is combining some of the correlated lists (Section V). This approach is based on two observations. First, the string ids on some lists can be correlated. For example, many English words that include the gram "`tio`" also include the gram "`ion`". Therefore, we could combine these two lists to save index space. Each of the two grams shares the union list. Notice that we could even combine this union list with another list if there is a strong correlation between them. Second, recent algorithms such as [20], [11] can efficiently handle long lists to answer approximate string queries. As a consequence, even if we combine some lists into longer lists, such an algorithm can still achieve a high performance. We study several technical problems in this approach, and analyze the effect of combining lists on a query. Also, we exploit a new opportunity to improve the performance of existing list-merging algorithms. Based on our analysis we develop a cost-based algorithm for finding lists to combine.

We have conducted extensive experiments on real datasets for the list-compression techniques mentioned above (Section VI). While existing inverted-list compression techniques can achieve compression ratios up to 60%, they considerably increase the average query running time due to the online decompression cost. The two novel approaches are orthogonal to existing inverted-list-compression techniques, and offer unique optimization opportunities for improving query performance. Note that using our novel approaches we can still compute the *exact* results for an approximate query without missing any true answers. The experimental results show that (1) the novel techniques can outperform existing compression techniques, and (2) the new techniques provide applications the flexibility in deciding the tradeoff between query performance and indexing size. An interesting and surprising finding is that while we can reduce the index size significantly (up to a 60% reduction) with tolerable performance penalties, for 20-40% reductions we can even improve the query performance compared to the original index. Our techniques work for commonly used functions such as edit distance, Jaccard, and cosine. We mainly focus on edit distance as an example for simplicity.

Due to space limitations, we leave more results in the 18-page full version of this paper [4].

### A. Related Work

In the literature the term *approximate string query* also means the problem of finding within a long text string those substrings that are similar to a given query pattern. See [25] for an excellent survey. In this paper, we use this term to refer to the problem of finding from a collection of strings those similar to a given query string.

In the field of list compression, many algorithms [23], [30], [7], [9] are developed to compress a list of integers using encoding schemes such as LZW, Huffman codes, and bloom filters. In Section III we discuss in more detail how to adopt these existing compression techniques to our setting. One observation is that these techniques often need to pay a high cost of increasing query time, due to the online decompression operation, while our two new methods could even reduce the query time. In addition, the new approaches and existing techniques can be integrated to further reduce the index size, as verified by our initial experiments.

Many algorithms have been developed for the problem of approximate string joins based on various similarity functions [2], [3], [5], [6], [10], [27], [28], especially in the context of record linkage. Some of them are proposed in the context of relational DBMS systems. Several recent papers focused on approximate *selection* (or search) queries [11], [20]. The techniques presented in this paper can reduce index sizes, which should also benefit join queries, and the corresponding cost-based analysis for join queries needs future research. Hore et al. [13] proposed a gram-selection technique for indexing text data under space constraints, mainly considering SQL LIKE queries. Other related studies include [17], [26]. There

are recent studies on the problem of estimating the selectivity of SQL LIKE substring queries [15], [18], and approximate string queries [22], [16], [19], [12].

Recently a new technique called VGRAM [21], [29] was proposed to use variable-length grams to improve approximate-string query performance and reduce the index size. This technique, as it is, can only support edit distance, while the techniques presented in this paper support a variety of similarity functions. Our techniques can also provide the user the flexibility to choose the tradeoff between index size and query performance, which is not provided by VGRAM. Our experiments show that our new techniques can outperform VGRAM, and potentially they can be integrated with VGRAM to further reduce the index size (Section VI-E).

## II. Preliminaries

Let $S$ be a collection of strings. An approximate string search query includes a string $s$ and a threshold $k$. It asks for all $r \in S$ such that the distance between $r$ and $s$ is within the threshold $k$. Various distance functions can be used, such as edit distance, Jaccard similarity and cosine similarity. Take edit distance as an example. Formally, the *edit distance* (a.k.a. Levenshtein distance) between two strings $s_1$ and $s_2$ is the minimum number of edit operations of single characters that are needed to transform $s_1$ to $s_2$. Edit operations include insertion, deletion, and substitution. We denote the edit distance between two strings $s_1$ and $s_2$ as $ed(s_1, s_2)$. For example, $ed$("`Levenshtein`", "`Levnshtain`") $= 2$. Using this function, an approximate string search with a query string $q$ and threshold $k$ is finding all $s \in S$ such that $ed(s, q) \leq k$.

Let $\Sigma$ be an alphabet. For a string $s$ of the characters in $\Sigma$, we use "$|s|$" to denote the length of $s$. We introduce two characters $\alpha$ and $\beta$ not in $\Sigma$. Given a string $s$ and a positive integer $q$, we extend $s$ to a new string $s'$ by prefixing $q-1$ copies of $\alpha$ and suffixing $q-1$ copies of $\beta$. (The results in the paper extend naturally to the case where we do not extend a string to produce grams.) A *positional q-gram* of $s$ is a pair $(i, g)$, where $g$ is the substring of length $q$ starting at the $i$-th character of $s'$. The set of *positional q-grams* of $s$, denoted by $G(s, q)$, or simply $G(s)$ when the $q$ value is clear in the context, is obtained by sliding a window of length $q$ over the characters of $s'$. For instance, suppose $\alpha$ =#, $\beta$ = \$, $q = 3$, and $s$ = `irvine`. We have: $G(s, q)$ = {(1, `##i`), (2, `#ir`), (3, `irv`), (4, `rvi`), (5, `vin`), (6, `ine`), (7, `ne$`), (8, `e$$`)}. The number of positional $q$-grams of the string $s$ is $|s|+q-1$. For simplicity, in our notations we omit positional information, which is assumed implicitly to be attached to each gram.

We construct an index as follows. For each gram $g$ of the strings in $S$, we have a list $l_g$ of the ids of the strings that include this gram (possibly with the corresponding positional information). It is observed in [27] that an approximate query with a string $s$ can be answered by solving the following generalized problem:

> *$T$-occurrence Problem*: Find the string ids that appear at least $T$ times on the inverted lists of the grams in $G(s, q)$, where $T$ is a constant related to the similarity function, the threshold in the query, and the gram length $q$.

Take edit distance as an example. For a string $r \in S$ that satisfies the condition $ed(r, s) \leq k$, it should share at least the following number of $q$-grams with $s$:

$$T_{ed} = (|s| + q - 1) - k \times q. \quad (1)$$

Several existing algorithms [20], [27] are proposed for answering approximate string queries efficiently. They first solve the $T$-occurrence problem to get a set of string candidates, and then check their real distance to the query string to remove false positives. Note that if the threshold $T \leq 0$, then the entire data collection needs to be scanned to compute the results. We call it a *panic case*. One way to reduce this scan time is to apply filtering techniques [10], [20]. To summarize, the following are the pieces of time needed to answer a query:

- If the lower bound $T$ (called "merging threshold") is positive, the time includes the time to traverse the lists of the query grams to find candidates (called "merging time") and the time to remove the false positives (called "post-processing time").
- If the lower bound $T$ is zero or negative, we need to spend the time (called "scan time") to scan the entire data set, possibly using filtering techniques.

In the following sections we adopt existing techniques and develop new techniques to reduce this index size. For simplicity, we mainly focus on the edit distance function, and the results are extended for other functions as well.

## III. Adopting Existing Compression Techniques

There are many techniques [31], [7], [9] on list compression, which mainly study the problem of representing integers on inverted lists efficiently to save storage space. In this section we study how to adopt these techniques to solve our problem and discuss their limitations.

Most of these techniques exploit the fact that ids on an inverted list are monotonically increasing integers. For example, suppose we have a list $l = (id_1, id_2, \ldots, id_n)$, $id_i < id_{i+1}$ for $1 \leq i < n$. If we take the differences of adjacent integers to construct a new list $l' = (id_1, id_2 - id_1, id_3 - id_2, \ldots, id_n - id_{n-1})$ (called the gapped list of $l$), the new integers tend to be smaller than the original ids. Many integer-compression techniques such as gamma codes, delta codes [7], and Golomb codes [9] can efficiently encode the gapped lists by using shorter representations for small integers. As an example, we study how to adopt one of the recent techniques called Carryover-12 [1].

An issue arises when using the encoded, gapped representation of a list. Many efficient list-merging algorithms in our setting [20] rely heavily on binary search on the inverted lists. Since decoding is usually achieved in a sequential way, a sequential scan on the list might not be affected too much. However, random accesses could become expensive. Even if the compression technique allows us to decode the desired integer directly, the gapped representation still requires restoring of all preceding integers. This problem can be solved

by segmenting the list and introducing *synchronization points* [24]. Each segment is associated with a synchronization point. Decoding can start from any synchronization point, so that only one segment needs to be decompressed in order to read a specific integer. We can make each segment contain the same number of integers. Since different encoded segments could have different sizes, we can index the starting offset of each encoded segment, so that they can be quickly located and decompressed. Figure 2 illustrates the idea of segmenting inverted lists and indexing compressed segments.

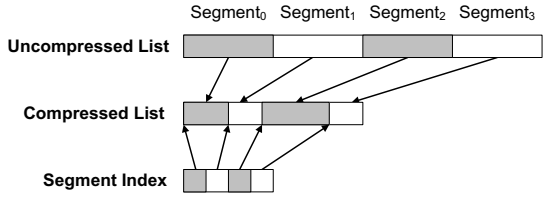



Fig. 2. Inverted-list compression with segmenting and indexing

One way to access elements is by decoding the corresponding segment for each random access. If multiple integers within the same segment are requested, the segment might be decompressed multiple times. The repeated efforts can be alleviated using caching. We allocate a global cache pool for all inverted lists. Once a segment is decoded, it will remain in the cache for a while. All integer accesses to that segment will be answered using the cache without decoding the segment.

*Limitations*: Most of these existing techniques were initially designed for compressing disk-based inverted indexes. Using a compressed representation, we can not only save disk space, but also decrease the number of disk I/Os. Even with the decompression overhead, these techniques can still improve query performance since disk I/Os are usually the major cost. When the inverted lists are in memory, these techniques require additional decompression operations, compared to non-compressed indexes. Thus, the query performance can only decrease. These approaches have limited flexibility in trading query performance with space savings. Next we propose two novel methods that do not have these limitations.

## IV. Discarding Inverted Lists

In this section we study how to reduce the size of an inverted-list index by discarding some of its lists. That is, for all the grams from the strings in $S$, we only keep inverted lists for *some* of the grams, while we do not store those of the other grams. A gram whose inverted list has been discarded is called a *hole gram*, and the corresponding discarded list is called its *hole list*. Notice that a hole gram is different from a gram that has an empty inverted list. The former means the ids of the strings with this gram are not stored in the index, while the latter means no string in the data set has this gram.

We study the effect of hole grams on query answering. In Section IV-A we analyze how they affect the merging threshold, the list merging and post-processing, and discuss how the new running time of a single query can be estimated. Based on our analysis, we propose an algorithm to wisely choose grams to discard in the presence of space constraints, while retaining efficient processing. We develop various optimization techniques to improve the performance (Section IV-B).

### A. Effects of Hole Grams on a Query

*1) Merging Threshold:* Consider a string $r$ in collection $S$ such that $ed(r, s) \leq k$. For the case without hole grams, $r$ needs to share at least $T = (|s|+q-1)-k\times q$ common grams in $G(s)$ (Equation 1). To find such an $r$, in the corresponding $T$-occurrence problem, we need to find string ids that appear on at least $T$ lists of the grams in $G(s)$. If $G(s)$ does have hole grams, the id of $r$ could have appeared on some of the hole lists. But we do not know on how many hole lists $r$ could appear, since these lists have been discarded. We can only rely on the lists of those nonhole grams to find candidates. Thus the problem becomes deciding a lower bound on the number of occurrences of string $r$ on the *nonhole* gram lists.

One simple way to compute a new lower bound is the following. Let $H$ be the number of hole grams in $G(s)$, where $|G(s)| = |s| + q - 1$. Thus, the number of nonhole grams for $s$ is $|G(s)| - H$. In the worst case, every edit operation can destroy at most $q$ nonhole grams, and $k$ edit operations could destroy at most $k \times q$ nonhole grams of $s$. Therefore, $r$ should share at least the following number of nonhole grams with $s$:

$$T' = |G(s)| - H - k \times q. \tag{2}$$

We can use this new lower bound $T'$ in the $T$-occurrence problem to find all strings that appear at least $T'$ times on the nonhole gram lists as candidates.

The following example shows that this simple way to compute a new lower bound is pessimistic, and the real lower bound could be tighter. Consider a query string $s$ = `irvine` with an edit-distance threshold $k = 2$. Suppose $q = 3$. Thus the total number of grams in $G(s)$ is 8. There are two hole grams `irv` and `ine` as shown in Figure 3. Using the formula above, an answer string should share at least 0 nonhole grams with string $s$, meaning the query can only be answered by a scan. This formula assumes that a single edit operation could potentially destroy 3 grams, and two operations could potentially destroy 6 grams. However, a closer look at the positions of the hole grams tells us that a single edit operation can destroy at most 2 nonhole grams, and two operations can destroy at most 4 nonhole grams. Figure 3 shows two deletion operations that can destroy the largest number of nonhole grams, namely 4. Thus, a tighter lower bound is 2 and we can avoid the panic case. This example shows that we can exploit the positions of hole grams in the query string to compute a tighter threshold. We develop a dynamic programming algorithm to compute a tight lower bound on the number of common nonhole grams in $G(s)$ an answer string needs to share with the query string $s$ with an edit-distance threshold $k$ (a similar idea is also adopted in an algorithm in [29] in the context of the VGRAM technique [21]). Our experiments have shown that this algorithm can increase query

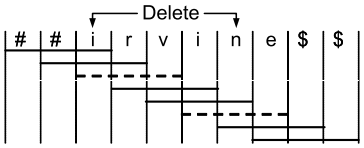



Fig. 3. A query string `irvine` with two hole grams. A solid horizontal line denotes a nonhole gram, and a dashed line denotes a hole gram. The arrows denote character deletions.

performance by tightening the bound. More details about the algorithm and experiments are in [4].

*2) List-Merging Time:* The running time of some merging algorithms (e.g. HeapMerge, ScanCount [20]) is insensitive to the merging threshold $T$ and mainly depends on the total number of elements in all inverted lists. Therefore, their running time can only decrease by discarding some lists. Other merging algorithms (e.g., MergeOpt, DivideSkip [20]) separate the inverted lists into a group of long lists and a group of short lists, and process them separately. The performance of these algorithms depends on how the two groups are formed, which is related to $T$. Thus their performance is sensitive to changes in $T$. Another class of algorithms such MergeSkip and DivideSkip [20] utilize $T$ to skip irrelevant elements on the lists. Decreasing $T$ by discarding some lists might negatively affect their performance. Meanwhile, we might have fewer lists to process, possibly resulting in an improvement of the query performance.

*3) Post-Processing Time:* For a given query, introducing hole grams may only increase the number of candidates to post-process if we use Equation 2. Surprisingly, if we use the dynamic programming algorithm to derive a tighter $T'$, then the number of candidates for post-processing might even decrease [4]. Take the example given in Fig. 3. Suppose the edit-distance threshold $k = 2$. Say that some string id $i$ only appears on the inverted lists of `irv` and `ine`. Since $T = 2$, it is a candidate result. If we choose to discard the grams `irv` and `ine` as shown in Fig. 3, as discussed earlier, the new threshold $T' = 2$. After discarding the lists, the string $i$ is not a candidate anymore, since all the lists containing it have been discarded. Thus we can reduce the post-processing cost. Note that any string id which appears only on `irv` and `ine` cannot be an answer to the query and would have been removed from the results during post-processing.

*4) Estimating Time Effects on a Query:* Since we are evaluating whether it is a wise choice to discard a specific list $l_i$, we want to know, by discarding list $l_i$, how the performance of a single query $Q$ will be affected using the indexing structure. We now quantify these effects discussed above by estimating the running time of a query with hole grams. In [4] we discuss how to estimate the merging time and scan time. We focus on estimating the post-processing time.

For each candidate from the $T$-occurrence problem, we need to compute the corresponding distance to the query to remove the false positives. This time can be estimated as the number of candidates multiplied by the average edit-distance time. Therefore, the main problem becomes how to estimate the number of candidates after solving the $T$-occurrence problem. This problem has been studied in the literature recently [22], [16], [19]. While these techniques could be used in our context, they have two limitations. First, their estimation is not 100% accurate, and an inaccurate result could greatly affect the accuracy of the estimated post-processing time, thus affecting the quality of the selected nonhole lists. Second, this estimation may need to be done *repeatedly* when choosing lists to discard, and therefore needs to be very efficient.

We develop an efficient, *incremental* algorithm that can compute a *very accurate* number of candidates for query $Q$ if list $l_i$ is discarded. The algorithm is called ISC, which stands for "Incremental-Scan-Count." Its idea comes from an algorithm called ScanCount developed in [20]. Although the original ScanCount is not the most efficient one for the $T$-occurrence problem, it has the nice property that it can be run incrementally. Figure 4 shows the intuition behind this ISC algorithm. First, we analyze the query $Q$ on the original indexing structure without any lists discarded. For each string id in the collection, we remember how many times it occurs on all the inverted lists of the grams in the query and store them in an array $C$. Now we want to know if a list is discarded, how it affects the number of occurrences of each string id. For each string id $r$ on list $l$ belonging to gram $g$ to be discarded, we decrease the corresponding value $C[r]$ in the array by the number of occurrences of $g$ in the query string, since this string $r$ will no longer have $g$ as a nonhole gram. After discarding this list for gram $g$, we first compute the new merging threshold $T'$. We find the new candidates by scanning the array $C$ and recording those positions (corresponding to string ids) whose value is at least $T'$.

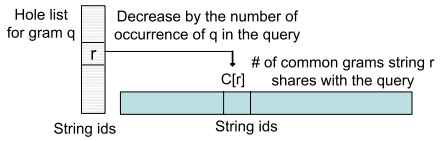



Fig. 4. Intuition behind the Incremental-Scan-Count (ISC) algorithm.

For instance, in Fig. 5, the hole list includes string ids 0, 2, 5, and 9. For each of them, we decrease the corresponding value in the array by 1 (assuming the hole gram occurs once in the query). Suppose the new threshold $T'$ is 3. We scan the new array to find those string ids whose occurrence among all non-hole lists is at least 3. These strings, which are 0, 1, and 9 (in bold face in the figure), are candidates for the query using the new threshold after this list is discarded.

### *B. Choosing Inverted-Lists to Discard*

We now study how to wisely choose lists to discard in order to satisfy a given space constraint. The following are several simple approaches: choosing the longest lists to discard (LongList), choosing the shortest lists to discard (ShortList), or choosing random lists to discard (RandomList). These naive approaches blindly discard lists without considering the

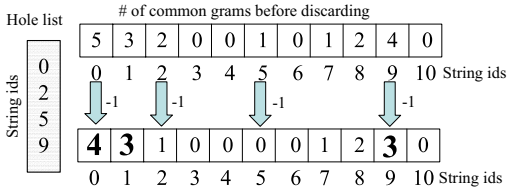



Fig. 5. Running the ISC algorithm ($T' = 3$).

effects on query performance. Clearly, a good choice of lists to discard depends on the query workload. Based on our previous analysis, we present a cost-based algorithm called DiscardLists, as shown in Figure 6. Given the initial set of inverted lists, the algorithm iteratively selects lists to discard, based on the size of a list and its effect on the average query performance for a query workload $\mathcal{Q}$ if it is discarded. The algorithm keeps selecting lists to discard until the total size of the remaining lists meets the given space constraint (line 2).

```
Algorithm: DiscardLists
Input: Inverted lists L = {l_1, ..., l_n}
       Constraint B on the total list size
       Query workload Q = {Q_1, ..., Q_m}
Output: A set D of lists in L that are discarded
Method:
1.  D = ∅;
2.  WHILE (B < (total list size of L)) {
3.      FOR (each list l_i ∈ L) {
4.          Compute size reduction Δ^i_size if discarding l_i
5.          Compute difference of average query time Δ^i_time
                for queries in Q if discarding l_i
        }
6.      Use Δ^i_size's and Δ^i_time's of the lists to decide what
            lists to discard
7.      Add discarded lists to D
8.      Remove the discarded lists from L
    }
9.  RETURN D
```

Fig. 6. Cost-based algorithm for choosing inverted lists to discard.

In each iteration (lines 3-8), the algorithm needs to evaluate the quality of each remaining list $l_i$, based on the expected effect of discarding this list. The effect includes the reduction $\Delta^i_{size}$ on the total index size, which is the length of this list. It also includes the change $\Delta^i_{time}$ on the average query time for the workload $\mathcal{Q}$ after discarding this list. (Surprisingly, $\Delta^i_{time}$ can be both positive and negative, since in some cases discarding lists can even reduce the average running time for the queries.) In each iteration (line 6), we need to use the $\Delta^i_{size}$'s and $\Delta^i_{time}$'s of the lists to decide what lists should be really discarded. There are many different ways to make this decision. One way is to choose a list with the smallest $\Delta^i_{time}$ value (notice that it could be negative). Another way is to choose a list with the smallest $\Delta^i_{time}/\Delta^i_{space}$ ratio.

There are several ways to reduce the computation time of the estimation: (1) When discarding the list $l_i$, those queries whose strings do not have the gram of $l_i$ will not be affected, since they will still have the same set of nonhole grams as before. Therefore, we only need to re-evaluate the performance of the queries whose strings have this gram of $l_i$. In order to find these strings efficiently, we build an inverted-list index structure for the *queries*, similar to the way we construct inverted lists for the strings in the collection. When discarding the list $l_i$, we can just consider those queries on the *query* inverted list of the gram for $l_i$. (2) We run the algorithm on a random subset of the strings. As a consequence, (i) we can make sure the entire inverted lists of these sample strings can fit into a given amount of memory. (ii) We can reduce the array size in the ISC algorithm, as well as its scan time to find candidates. (iii) We can reduce the number of lists to consider initially since some infrequent grams may not appear in the sample strings. (3) We run the algorithm on a random subset of the queries in the workload $\mathcal{Q}$, assuming this subset has the same distribution as the workload. As a consequence, we can reduce the computation to estimate the scan time, merging time, and post-processing time (using the ISC algorithm). (4) We do not discard those very short lists, thus we can reduce the number of lists to consider initially. (5) In each iteration of the algorithm, we choose multiple lists to discard based on the effect on the index size and overall query performance. In addition, for those lists that have very poor time effects (i.e., they affect the overall performance too negatively), we do not consider them in future iterations, i.e., we have decided to keep them in the index structure. In this way we can reduce the number of iterations significantly.

## V. Combining Inverted Lists

In this section, we study how to reduce the size of an inverted-list index by *combining* some of the lists. Intuitively, when the lists of two grams are similar to each other, using a single inverted list to store the union of the original two lists for both grams could save some space. One subtlety in this approach is that the string ids on a list are treated as a *set* of ordered elements (without duplicates), instead of a *bag* of elements. By combining two lists we mean taking the *union* of the two lists so that space can be saved. Notice that the $T$ lower bound in the $T$-occurrence problem is derived from the perspective of the grams in the query. (See Equation 1 in Section II as an example.) Therefore, if a gram appears multiple times in a data string in the collection (with different positions), on the corresponding list of this gram the string id appears only once. If we want to use the positional filtering technique (mainly for the edit distance function) described in [10], [20], for each string id on the list of a gram, we can keep a range of the positions of this gram in the string, so that we can utilize this range to do filtering. When taking the union of two lists, we need to accordingly update the position range for each string id.

We will first discuss the data structure and the algorithm for efficiently combining lists in Section V-A, and then analyze the effects of combining lists on query performance in Section V-B. We also show that an index with combined inverted lists gives us a new opportunity to improve the performance

of list-merging algorithms (Section V-B.1). We propose an algorithm for choosing lists to combine in the presence of space constraints (Section V-C).

### A. Data Structures for Combining Lists

In the original inverted-list structure, different grams have different lists. Combining two lists $l_1$ and $l_2$ will produce a new list $l_{new} = l_1 \cup l_2$. The size reduction of combining two lists $l_1$ and $l_2$ can be computed as

$$\Delta_{size}^{(1,2)} = |l_1| + |l_2| - |l_1 \cup l_2| = |l_1 \cap l_2|.$$

All grams that shared $l_1$ and $l_2$ (there could be several grams due to earlier combining operations) will now share list $l_{new}$. In this fashion we can support combining more than two lists iteratively. We use a data structure called Disjoint-Set with the algorithm Union-Find [8] to efficiently combine more than two lists, as illustrated in Figure 7. More details are in [4].

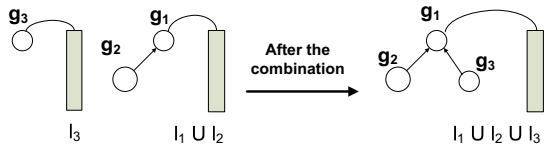



Fig. 7. Combining list of $g_2$ with list of $g_3$ using Union-Find

### B. Effects of Combining Lists on Query Performance

We study how combining lists affects query performance. For a similarity query with a string $s$, if the lists of the grams in $G(s)$ are combined (possibly with lists of grams not in $G(s)$), then the performance of this query can be affected in the following ways. (1) Different from the approach of discarding lists, the lower bound $T$ in the $T$-occurrence problem remains the same, since an answer still needs to appear at least this number of times on the lists. Therefore, if a query was not in a panic case before, then it will not be in a panic case after combining inverted lists. (2) The lists will become longer. As a consequence, it will take more time to traverse these lists to find candidates during list merging, and more false positives may be produced to be post-processed.

*1) List-Merging Time:* As inverted lists get combined, some of them will become longer. In this sense it appears that combining lists can only increase the list-merging time in query answering. However, the following observation opens up opportunities for us to further decrease the list-merging time, given an index structure with combined lists. We notice that a gram could appear in the query string $s$ multiple times (with different positions), thus these grams share common lists. In the presence of combined lists, it becomes possible for even different grams in $G(s)$ to share lists. This sharing suggests a way to improve the performance of existing list-merging algorithms for solving the $T$-occurrence problem [27], [20]. A simple way to use one of these algorithms is to pass it a list for each gram in $G(s)$. Thus we pass $|G(s)|$ lists to the algorithm to find string ids that appear at least $T$ times on these (possibly shared) lists. We can improve the performance of the algorithm as follows. We first identify the shared lists for the grams in $G(s)$. For each *distinct* list $l_i$, we also pass to the algorithm the number of grams sharing this list, denoted by $w_i$. Correspondingly, the algorithm needs to consider these $w_i$ values when counting string occurrences. In particular, if a string id appears on the list $l_i$, the number of occurrences should increase by $w_i$, instead of "1" in the traditional setting. Thus we can reduce the number of lists passed to the algorithm, thus possibly even reducing its running time. The algorithms in [27] already consider different list weights, and the algorithms in [20] can be modified slightly to consider these weights.[1]

*2) Post-processing Time:* We want to compute the number of candidates generated from the list-merging algorithm. Before combining any lists, the candidate set generated from a list-merging algorithm contains all correct answers and some false positives. We are particularly interested to know how many *new* false positives will be generated by combining two lists $l_1$ and $l_2$. The ISC algorithm described in Section IV-A.4 can be modified to adapt to this setting.

In the algorithm, a ScanCount vector is maintained for a query $Q$ to store the number of grams $Q$ shares with each string id in the collection. The strings whose corresponding values in the ScanCount vector are at least $T$ will be candidate answers. By combining two lists $l_1$ and $l_2$, the lists of those grams that are mapped to $l_1$ or $l_2$ will be conceptually extended. Every gram previously mapped to $l_1$ or $l_2$ will now be mapped to $l_1 \cup l_2$. The extended part of $l_1$ is $ext(l_1) = l_2 \backslash l_1$. Let $w(Q, l_1)$ denote the number of times grams of $Q$ reference $l_1$. The ScanCount value of each string id in $ext(l_1)$ will be increased by $w(Q, l_1)$. Since for each reference, all string ids in $ext(l_1)$ should have their ScanCount value increased by one, the total incrementation will be $w(Q, l_1)$ (*not* $w(Q, l_2)$). The same operation needs to be done for $ext(l_2)$ symmetrically. It is easy to see the ScanCount values are monotonically increasing as lists are combined. The strings whose ScanCount values increase from below $T$ to at least $T$ become new false positives after $l_1$ and $l_2$ are combined.

Figure 8 shows an example, in which $l_1 = \{0, 2, 8, 9\}$, $l_2 = \{0, 2, 3, 5, 8\}$. Before combining $l_1$ and $l_2$, two grams of $Q$ are mapped to $l_1$ and three grams are mapped to $l_2$. Therefore, $w(Q, l_1) = 2$ and $w(Q, l_2) = 3$. For every string id in $ext(l_1) = \{3, 5\}$, their corresponding values in the ScanCount vector will be increased by $w(Q, l_1)$. Let $C$ denote the ScanCount vector. $C[3]$ will be increased from 6 to 8, while $C[5]$ will be increased from 4 to 6. Given the threshold $T = 6$, the change on C[5] indicates that string 5 will become a new false positive. The same operation is carried out on $ext(l_2)$.

### C. Choosing Lists to Combine

We use two steps to combine lists: discovering candidate gram pairs, and selecting some of them to combine.

[1] Interestingly, our experiments showed that, even for the case we do not combine lists, this optimization can already reduce the running time of existing list-merging algorithms by up to 20%.

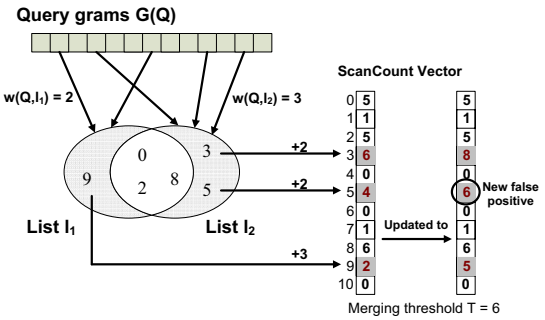



Fig. 8. Example of ISC for computing new false positives after combining lists $l_1$ and $l_2$.

**Step 1: Discovering Candidate Gram Pairs**: We are only interested in combining *correlated* lists. We can use Jaccard similarity to measure the correlation of two lists, defined as $jaccard(l_1, l_2) = \frac{|l_1 \cap l_2|}{|l_1 \cup l_2|}$. Two lists are considered to be combined only if their correlation is greater than a threshold. Clearly it is computationally prohibitive to consider all pairs of grams. There are different ways for generating such pairs. One way is using adjacent grams. We only consider pairs of *adjacent* grams in the strings. If we use $q$-grams to construct the inverted lists, we can just consider those $(q+1)$-grams. Each such gram corresponds to a pair of $q$-grams. For instance, if $q = 3$, then the 4-gram `tion` corresponds to the pair $(\texttt{tio}, \texttt{ion})$. For each such adjacent pair, we treat it as a candidate pair if the Jaccard similarity of their corresponding lists is greater than a predefined threshold. One limitation of this approach is that it cannot find strongly correlated grams that are not adjacent in strings. In the literature there are efficient techniques for finding strongly correlated pairs of lists. One of them is called Locality-Sensitive Hashing (LSH) [14]. Using a small number of so-called MinHash signatures for each list, we can use LSH to find those gram pairs whose lists satisfy the above correlation condition with a high probability.

**Step 2: Selecting Candidate Pairs to Combine**: The second step is selecting candidate pairs to combine. One basic algorithm is the following. We iteratively pick gram pairs and combine their lists if their correlation satisfies the threshold. Notice that each time we process a new candidate gram pair, since the list of each of them could have been combined with other lists, we still need to verify their (possibly new) correlation before deciding whether we should combine them. After processing all these pairs, we check if the index size meets a given space constraint. If so, the process stops. Otherwise, we decrease the correlation threshold and repeat the process above, until the new index size meets the given space constraint.

This basic algorithm does not consider the effect of combining two lists on the overall query performance. We propose a cost-based algorithm to wisely choose lists to combine in the second step. Figure 9 shows the cost-based algorithm which takes the estimated cost of a query workload into consideration when choosing lists to combine. It iteratively selects pairs to combine, based on the space saving and the impact on the average query performance of a query workload $\mathcal{Q}$. The algorithm keeps selecting pairs to combine until the total size of the inverted lists meets a given space constraint $B$. For each gram pair $(g_i, g_j)$, we need to get their current corresponding lists, since their lists could have been combined with other lists (lines 3 and 4). We check whether these two lists are the same list as reference (line 5), and also whether their correlation is above the threshold (line 6). Then we compute the size reduction (line 8) and estimate the average query time difference and the ISC algorithm (line 9), based on which we decide the next list pair to combine (lines 10 and 11).

Algorithm: **CombineLists**
Input: Candidate gram pairs $P = \{(g_i, g_j)\}$
Constraint $B$ on the total list size
Query workload $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$
Output: Combined lists.
Method:

```
1.  WHILE ((expected total index size) > B ) {
2.    FOR (each gram pair (g_i, g_j) ∈ P) {
3.      l_i = current list of g_i
4.      l_j = current list of g_j
5.      if (l_i and l_j are the same list as reference
6.          or corr(l_i, l_j) < δ)
7.          remove (g_i, g_j) from P and continue
8.      Compute size reduction Δ_size^(l_i,l_j) if combining l_i, l_j
9.      Compute difference of average query time Δ_time^(l_i,l_j)
            for queries in Q if combining l_i, l_j
      }
10.   Use Δ_size^(l_i,l_j)'s and Δ_time^(l_i,l_j)'s of the gram pairs to decide
        which pair to combine
11.   Combine the two lists l_i and l_j based on the decision
12.   Remove the combined gram pair from P
    }
```

Fig. 9. Cost-based algorithm to select gram pairs to combine.

We can use similar optimization techniques as described in Section IV to improve the performance of CombineLists.

## VI. Experiments

We used three real data sets. (1) *IMDB Actor Names*: It consists of the actor names downloaded from the IMDB website (http://www.imdb.com). There were 1,199,299 names. The average string-length was 17 characters. (2) *WEB Corpus Word Grams*: This data set (http://www.ldc.upenn.edu/Catalog, number LDC2006T13) contained word grams and their observed frequency counts on the Web. We randomly chose 2 million records with a size of 48.3MB. The number of words of a string varied from 3 to 5. The average string-length was 24. (3) *DBLP Paper Titles*: It includes paper titles downloaded from the DBLP Bibliography site (http://www.informatik.uni-trier.de/∼ley/db). It had 274,788 paper titles. The average string-length was 65.

For all experiments the gram length $q$ was 3, and we applied length filtering [10]. The inverted-list index was held in main memory. Also, for the cost-based DiscardLists and CombineLists approaches, by doing sampling we guaranteed that

the index structures of sample strings fit into memory. We used the DivideSkip algorithm described in [20] to solve the $T$-occurrence problem due to its high efficiency. From each data set we used 1 million strings to construct the inverted-list index (unless specified otherwise). We tested query workloads using different distributions, e.g., a Zipfian distribution or a uniform distribution. To do so, we randomly selected 1,000 strings from each data set and generated a workload of 10,000 queries according to some distribution. We conducted experiments using edit distance, Jaccard similarity, and cosine similarity. We mainly focused on the results of edit distance (with a threshold 2). We report additional results of other functions in Section VI-D. All the algorithms were implemented using GNU C++ and run on a Dell PC with 2GB main memory, and a 3.4GHz Dual Core CPU running the Ubuntu OS.

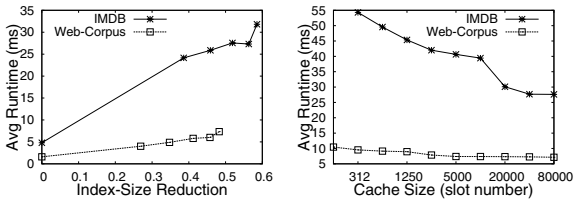
### A. Evaluating the Carryover-12 Compression Technique

We adopted the Carryover-12 compression technique as discussed in Section III into our problem setting. We varied the segment size to achieve different index-size reduction ratios. We measured the corresponding query performance. Figure 10(a) shows the results for the IMDB and Web Corpus datasets as the reduction ratio increased. (Notice that the two data sets used different reduction ratios because of the limitation of the technique.) Consider the Web Corpus dataset. The original average query running time (without compression) was about 1.6ms. After compressing the index, the query time increased significantly. For example, when the reduction ratio was about 41%, the query time increased to 5.7ms. The time kept increasing as we compressed the index further.

Figure 10(b) shows how the query time was affected as we increased the cache size. (The cache size was significantly smaller than the compressed index size.) On the WebCorpus data set, when we used no cache, the average query time was 64.4ms, which is more than 8 times the average query time with a cache of 5000 slots. Since the whole purpose of compressing inverted list is to save space, it is contradictory to improve query performance by increasing the cache size too much. As we allocated more cache to the compressed index, the query time did decrease. Notice that if we allocate enough cache for the entire compressed index, the performance can become almost the same as that of the original index (without considering the cache lookup overhead). As the cache size is typically much smaller than the original index size, the performance should always be worse than the original case due to the online decompression cost.

### B. Evaluating the DiscardLists Algorithm

In this section we evaluate the performance of the DiscardLists algorithm for choosing inverted-lists to discard. In addition to the three basic methods to choose lists to discard (LongList, ShortList, RandomList), we also implemented the following cost-based methods. (1) PanicCost: In each iteration we discard the list with the smallest ratio between the list size and the number of additional panic cases. Another similar approach, called PanicCost$^+$, discards the list with the smallest number of additional panic cases, disregarding the list length. (2) TimeCost: It is similar to PanicCost, except that we use the ratio between the list size and the total time effect of discarding a list (instead of the number of additional panics). Similarly, an approach called TimeCost$^+$ discards the list with the smallest time effect.




(a) Effect of index compression (b) Effect of caching

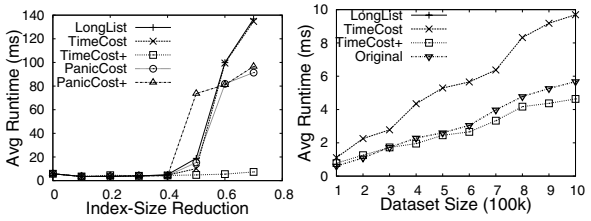
Fig. 10. Carryover-12 compression.

The index-construction time consisted of two major parts: selecting lists to discard and generating the final inverted-list structure. The time for generating samples was negligible. For the LongList, ShortList, and RandomList approaches, the time for selecting lists to discard was small, whereas in the cost-based approaches the list-selection time was prevalent. In general, increasing the size-reduction ratio also increased the list-selection time. For instance, for the IMDB dataset, at a 70% reduction ratio, the total index-construction time for the simple methods was about half a minute. The construction time for PanicCost and PanicCost$^+$ was similar. The more complex TimeCost and TimeCost$^+$ methods needed 108s and 353s, respectively.

**Different Methods to Choose Lists to Discard:** We first considered the three simple methods, namely LongList, ShortList, RandomList. Experiments [4] showed that in most cases, the LongList method gave us the best query performance, while the RandomList method was the best for high reduction ratios. The ShortList was always the worst.

For those cost-based approaches, we used a sampling ratio of 0.1% for the data strings and a ratio of 25% for the queries. Figure 11(b) shows the benefits of employing the cost-based methods to select lists to discard. Most noteworthy of which is the TimeCost$^+$ method, which consistently delivered good query performance. As shown in Fig. 11(b), the method achieved a 70% reduction ratio while increasing the query processing time from the original 5.8ms to 7.4ms only. All the other methods increased the time up to at least 96ms for that reduction ratio. Notice that TimeCost$^+$ ignored the list size when selecting a list to discard. TimeCost$^+$ over-topped all the other methods because it can balance the merging time, post-processing time, and scan time.

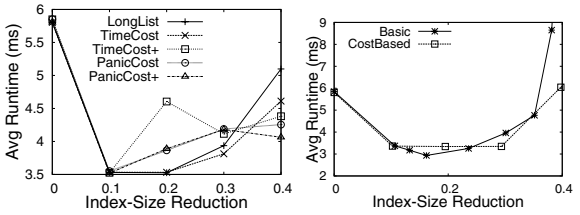
**Surprising Improvement on Performance:** Figs. 12(a) shows more details when the reduction ratio was smaller (less than 40%). A surprising finding is that, for low to moderate reduction ratios, discarding lists could even improve the query performance! All the methods reduced the average query time from the original 5.8ms to 3.5ms for a 10% reduction ratio. The main reason of the performance improvement is that




(a) Choosing lists to discard (b) Scalability

Fig. 11. Reducing index size by discarding lists (IMDB).

by discarding long lists we can help list-merging algorithms solve the $T$-occurrence problem more efficiently. We see that significantly reducing the number of total list-elements to process can overcompensate for the decrease in the threshold.




(a) DiscardLists (b) CombineLists

Fig. 12. Improving query performance using two new approaches (IMDB).

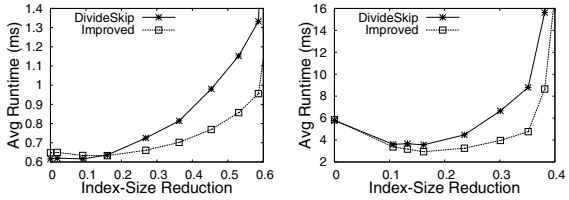
**Scalability**: For each data set, we increased its number of strings, and used 50% as the index-size-reduction ratio. Fig. 11(d) shows that the TimeCost$^+$ method performed consistently well, even outperforming the corresponding uncompressed indexes (indicated by "original"). At 100,000 data strings, the average query time increased from 0.61ms to 0.78ms for TimeCost$^+$. As the data size increased, TimeCost$^+$ began outperforming the uncompressed index.

### C. Evaluating the CombineLists Algorithm

We evaluated the performance of the CombineLists algorithm on the same three data sets. In step 1, we generated candidate list pairs by using both $(q+1)$-grams and LSH. In step 2, we implemented both the basic and the cost-based algorithms for iteratively selecting list pairs to combine.

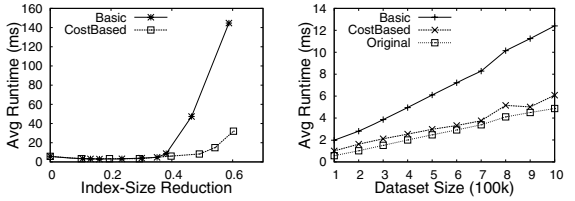
**Benefits of Improved List-Merging Algorithms**: We first evaluated the benefits of using the improved list-merging algorithms to solve the $T$-occurrence problem for queries on combined inverted lists, as described in Section V-B. As an example, we compared the DivideSkip algorithm in [20] and its improved version that considers duplicated inverted lists in a query. We used the basic algorithm to select lists to combine. Figure 13 shows the average running time for the algorithm and its improved version (marked as "Improved"). When the reduction ratio increased, more lists were combined, and the improved algorithm did reduce the average query time.

**Choosing Lists to Combine**: We compared the basic algorithm with the cost-based algorithm for choosing lists to combine, and the results are shown in Figure 14(a). The average query time was plotted over different reduction ratios




(a) DBLP titles (b) IMDB actors

Fig. 13. Reducing query time using improved list-merging algorithm.




(a) Choosing lists to combine (b) Scalability

Fig. 14. Reducing index size by combining lists (IMDB).

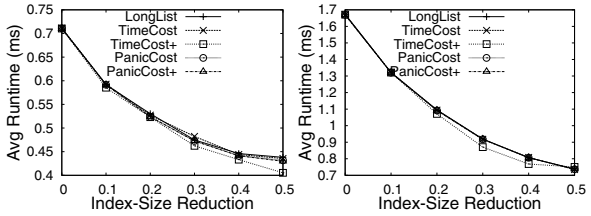
for both algorithms. We observe that on all three data sets, the query running time for both algorithms increased very slowly as we increased the index size reduction ratio, until about 40% to 50%. That means, this technique can reduce the index size without increasing the query time! As we further increased the index size reduction, the query time started to increase. For the cost-based algorithm, the time increased slowly, especially on the IMDB data set. The reason is that this cost-based algorithm avoided choosing bad lists to combine, while the basic algorithm blindly chose lists to combine.

Figure 12(b) shows that when the reduction ratio is less than 40%, the query time even decreased. This improvement is mainly due to the improved list-merging algorithms. Figure 14(b) shows how the algorithms of combining lists affected query performance as we increased the data size, for a reduction ratio of 40%.

### D. Extension to Other Similarity Functions

For simplicity, our discussion so far mainly focused on the edit distance metric. We can generalize the results to commonly used similarity measures such as Jaccard and cosine. To reduce the size of inverted lists based on those similarity functions, the main procedure of algorithms DiscardLists and CombineLists remains the same. The only difference is that in DiscardLists, to compute the merging threshold $T$ for a query after discarding some lists, we need to subtract the number of hole lists for the query from the formulas proposed in [20]. In addition, for the estimation of the post-processing time, we also need to replace the estimation of the edit distance time with that of Jaccard and cosine time respectively. Figure 15 shows the average running time for the DBLP data using variants of the TimeCost$^+$ algorithm for these two functions. The results on the other two data sets were similar. We see that the average running time continuously decreased when the reduction ratio increased to up to 40%. For example, at a

40% reduction ratio for the cosine function, the running time decreased from 1.7ms to 0.8ms.




(a) Jaccard function (b) Cosine function

Fig. 15. Jaccard and Cosine functions (DiscardLists, DBLP titles)

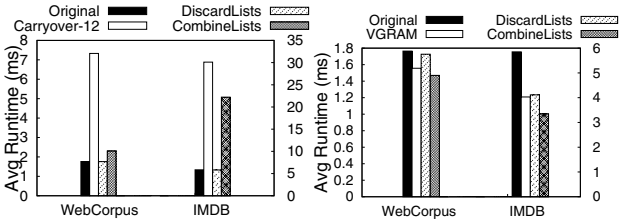
The performance started degrading at a 50% reduction ratio and increased rapidly at a ratio higher than 60%. For a 70% ratio, the time for the cosine and jaccard functions increased to 150ms and 115ms for LongList. For high reduction ratios the TimeCost and TimeCost$^+$ methods became worse than the panic-based methods, due to the inaccuracy in estimating the merging time. Note that the Cosine and Jaccard functions are expensive to compute, therefore the punishment (in terms of post-processing time) for inaccurately estimating the merging time can be much more severe than that for the edit distance.

### *E. Comparing Different Compression Techniques*

We implemented the compression techniques discussed so far as well as the VGRAM technique. Since Carryover-12 and VGRAM do not allow explicit control of the compression ratio, for each of them we reduced the size of the inverted-list index and computed their compression ratio. Then we compressed the index using DiscardLists and CombineLists separately to achieve the same compression ratio.

**Comparison with Carryover-12**: Figure 16(a) compares the performance of the two new techniques with Carryover-12. For Carryover-12, to achieve a good balance between the query performance and the index size, we used fixed-size segments of 128 4-byte integers and a synchronization point for each segment. The cache contained 20,000 segment slots (approximately 10MB). It achieved a compression ratio of 58% for the IMDB dataset and 48% for the WebCorpus dataset. We see that its online decompression has a profound impact on the performance. It increased the average running time from an original 5.85ms to 30.1ms for the IMDB dataset, and from an original 1.76ms to 7.32ms for the WebCorpus dataset. The CombineLists method performed significantly better at 22.15ms for the IMDB dataset and 2.3ms for the WebCorpus dataset. The DiscardLists method could even slightly decrease the running time compared to the original index to 5.81ms and 1.75ms for the IMDB and WebCorpus datasets, respectively.

**Comparison with VGRAM**: Figure 16(b) compares the performance of two new techniques with VGRAM. We set its $q_{min}$ parameter to 4. We did not take into account the memory requirement for the dictionary trie structure because it was negligible. The compression ratio was 30% for the IMDB dataset and 27% for the WebCorpus dataset. Interestingly,




(a) Carryover-12 (b) VGRAM

Fig. 16. Comparing DiscardLists and CombineLists with existing techniques at the same reduction ratio. In each figure, the left scale corresponds to the WebCorpus data set, and the right scale corresponds to the IMDB data set.

all methods could outperform the original, uncompressed index. As suspected, VGRAM can considerably reduce the running time for both datasets. For the IMDB dataset, it reduced the time from an original 5.85ms to 4.02ms, and for the WebCorpus dataset from 1.76ms 1.55ms. Surprisingly, the CombineLists method reduced the running time even more than VGRAM to 3.34ms for the IMDB dataset and to 1.47ms for the WebCorpus dataset. The DiscardLists method performed competitively for the IMDB dataset at 3.93ms and slightly faster than the original index (1.67ms) on the WebCorpus dataset.

**Summary**: (1) CombineLists and DiscardLists can significantly outperform Carryover-12 at the same memory reduction ratio because of the online decompression required by Carryover-12. (2) For small compression ratios CombineLists performs best, even outperforming VGRAM. (3) For large compression ratios DiscardLists delivers the best query performance. (4) While Carryover-12 can achieve reductions up to 60% and VGRAM up to 30%, neither allows explicit control over the reduction ratio. DiscardLists and CombineLists offer this flexibility with good query performance.

### *F. Integrating Several Approaches*

The methods studied in this paper are indeed orthogonal, thus we could even use their combinations to further reduce the index size and/or improve query performance. As an example, we integrated CombineLists with Carryover-12. We first compressed the index using CombineLists approach with a reduction $\alpha$, and then applied Carryover-12 on the resulting index. We varied $\alpha$ from 0 (no reduction for CombineLists) to 60% in 10% increments. The results of the overall reduction ratio and the average query time are shown in the "CL+Carryover-12" curve in Figure 17. The leftmost point on the curve corresponds to the case where $\alpha = 0$. For comparison purposes, we also plotted the results of using the CombineLists alone shown on the other curve. The results clearly show that using both methods we can achieve high reduction ratios with a better query performance than using CombineLists alone. Consider the first point that only uses Carryover-12. It could achieve a 48% reduction with an average query time of 7.3ms. By first using CombineLists at a 30% ratio (4th point on the curve) we could achieve a higher reduction ratio (61%) at a lower query time (6.34ms).

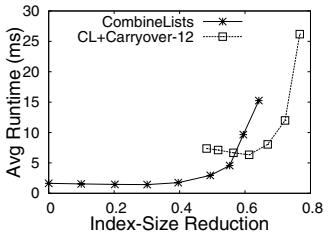



Fig. 17. Reducing index size using CombineLists with Carryover-12.

One way to integrate multiple methods is to distribute the global memory constraint among several methods. Notice since Carryover-12 and VGRAM do not allow explicit control of the index size, it is not easy to use them to satisfy an arbitrary space constraint. Several open challenging problems need more future research. First, we need to decide how to distribute the global memory constraint among different methods. Second, we need to decide in which order to use them. For example, if we use CombineLists first, then we never consider discarding merged lists in DiscardLists. Similarly, if we run DiscardLists first, then we never consider combining any discarded list in CombineLists.

**Additional Experiments**: In [4] we included many additional experimental results, including experiments on more data sets, performance of different methods to choose candidate pairs to combine, and how the techniques perform in the presence of query-workload changes. We also discuss how to utilize filtering techniques for compression.

## VII. Conclusions

In this paper, we studied how to reduce the size of inverted-list index structures of string collections to support approximate string queries. We studied how to adopt existing inverted-list compression techniques to achieve the goal, and proposed two novel methods for achieving the goal: one is based on discarding lists, and one based on combining correlated lists. They are both orthogonal to existing compression techniques, exploit a unique property of our setting, and offer new opportunities for improving query performance. We studied technical challenges in each method, and proposed efficient, cost-based algorithms for solving related problems. Our extensive experiments on real data sets show that our approaches provide applications the flexibility in deciding the tradeoff between query performance and indexing size and can outperform existing compression techniques.

**Acknowledgements:** The work was partially supported by the National Science Foundation of China under Grant No. 60828004.

## References

[1] V. N. Anh and A. Moffat. Inverted index compression using word-aligned binary codes. *Inf. Retr.*, 8(1):151–166, 2005.

[2] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, pages 918–929, 2006.

[3] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[4] A. Behm, S. Ji, C. Li, and J. Lu. Space-constrained gram-based indexing for efficient approximate string search (full version). Technical report, Department of Computer Science, UC Irvine, June 2008.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD Conference*, pages 313–324, 2003.

[6] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[7] P. Elias. Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on*, 21(2):194–203, Mar 1975.

[8] Z. Galil and G. F. Italiano. Data structures and algorithms for disjoint set union problems. *ACM Comput. Surv.*, 23(3):319–344, 1991.

[9] S. Golomb. Run-length encodings (corresp.). *Information Theory, IEEE Transactions on*, 12(3):399–401, Jul 1966.

[10] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[11] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In *ICDE*, pages 267–276, 2008.

[12] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava. Hashed samples: Selectivity estimators for set similarity selection queries. In *VLDB*, 2008.

[13] B. Hore, H. Hacigümüs, B. R. Iyer, and S. Mehrotra. Indexing text data under space constraints. In *CIKM*, pages 198–207, 2004.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC Conference*, 1998.

[15] H. V. Jagadish, R. T. Ng, and D. Srivastava. Substring selectivity estimation. In *PODS*, pages 249–260, 1999.

[16] L. Jin and C. Li. Selectivity estimation for fuzzy string predicates in large data sets. In *VLDB*, pages 397–408, 2005.

[17] M.-S. Kim, K.-Y. Whang, J.-G. Lee, and M.-J. Lee. n-Gram/2L: A space and time efficient two-level n-gram inverted index structure. In *VLDB*, pages 325–336, 2005.

[18] P. Krishnan, J. S. Vitter, and B. R. Iyer. Estimating alphanumeric selectivity in the presence of wildcards. In *SIGMOD Conference*, pages 282–293, 1996.

[19] H. Lee, R. T. Ng, and K. Shim. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB*, pages 195–206, 2007.

[20] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *ICDE*, pages 257–266, 2008.

[21] C. Li, B. Wang, and X. Yang. VGRAM: Improving performance of approximate queries on string collections using variable-length grams. In *VLDB*, pages 303–314, 2007.

[22] A. Mazeika, M. H. Böhlen, N. Koudas, and D. Srivastava. Estimating the selectivity of approximate string queries. *ACM Trans. Database Syst.*, 32(2):12, 2007.

[23] M. D. McIllroy. Development of a spelling list. *IEEE Transactions on Communications*, 30(1):91–99, 1998.

[24] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.*, 14(4):349–379, 1996.

[25] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[26] S. C. Sahinalp, M. Tasan, J. Macker, and Z. M. Özsoyoglu. Distance based indexing for string proximity search. In *ICDE*, pages 125–, 2003.

[27] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *SIGMOD Conference*, pages 743–754, 2004.

[28] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[29] X. Yang, B. Wang, and C. Li. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In *SIGMOD Conference*, 2008.

[30] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[31] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6, 2006.

*Science in China Series F: Information Sciences*


www.scichina.com
info.scichina.com
www.springer.com/scp
www.springerlink.com

# Efficient processing of partially specified twig pattern queries

ZHOU JunFeng[1,2], MENG XiaoFeng[1†] & LING TokWang[3]

[1] School of Information, Renmin University of China, Beijing 100872, China;
[2] Department of Computer Science and Technology, Yanshan University, Qinhuangdao 066004, China;
[3] School of Computing, National University of Singapore, 117590, Singapore

**As huge volumes of data are organized or exported in tree-structured form, it is quite necessary to extract useful information from these data collections using effective and efficient query processing methods. A natural way of retrieving desired information from XML documents is using twig pattern (TP), which is, actually, the core component of existing XML query languages. Twig pattern possesses the inherent feature that query nodes on the same path have concrete precedence relationships. It is this feature that makes it infeasible in many actual scenarios. This has driven the requirement of relaxing the complete specification of a twig pattern to express more flexible semantic constraints in a single query expression. In this paper, we focus on query evaluation of partially specified twig pattern (PSTP) queries, through which we can reap the most flexibility of specifying partial semantic constraints in a query expression. We propose an extension to XPath through introducing two Samepath axes to support partial semantic constraints in a concise but effective way. Then we propose a stack based algorithm, pTwigStack, to process a PSTP holistically without deriving the concrete twig patterns and then processing them one by one. Further, we propose two DTD schema based optimization methods to improve the performance of pTwigStack algorithm. Our experimental results on various datasets indicate that our method performs significantly better than existing ones when processing PSTPs.**

XML database, query processing, partially specified twig pattern, holistic twig join, XPath

## 1 Introduction

As a de facto standard for information representation and exchange over the Internet, XML has been used extensively in many applications. Query capabilities are provided through twig patterns (TPs), which are the core components for standard XML query languages, e.g. XPath (http://www.w3.org/TR/xpath20/) and XQuery (http://www.w3.org/TR/xquery/). A TP can be naturally represented as a node-labeled tree, where each edge denotes either Parent-Child (P-C) or Ancestor-Descendant (A-D) relationship. For example, the TP written in XPath format, $Q_1$: "//book[.//author/name= 'Mike']/title", selects

Received January 18, 2008; accepted November 5, 2008
doi: 10.1007/s11432-009-0152-3
†Corresponding author (email: xfmeng@ruc.edu.cn)
Supported partially by the National Natural Science Foundation of China (Grant No. 60833005), the National High-Tech Research & Development Program of China (Grant Nos. 2007AA01Z155, 2009AA011904), and the National Basic Research Program of China (Grant No. 2003CB317000)

**Citation:** Zhou J F, Meng X F, Ling T W. Efficient processing of partially specified twig pattern queries. Sci China Ser F-Inf Sci, 2009, 52(10): 1830–1847, doi: 10.1007/s11432-009-0152-3

title elements which are children of some book elements written by an author named "Mike". While many existing algorithms[1−4] can efficiently process a given TP, an inherent restrictive feature of TP is that a concrete precedence order between the nodes in every path of the query expression should be clearly specified. In $Q_1$, for example, book should be an ancestor of author; thus $Q_1$ can only be used to retrieve information from $D_2$ in Figure 1, not $D_1$.

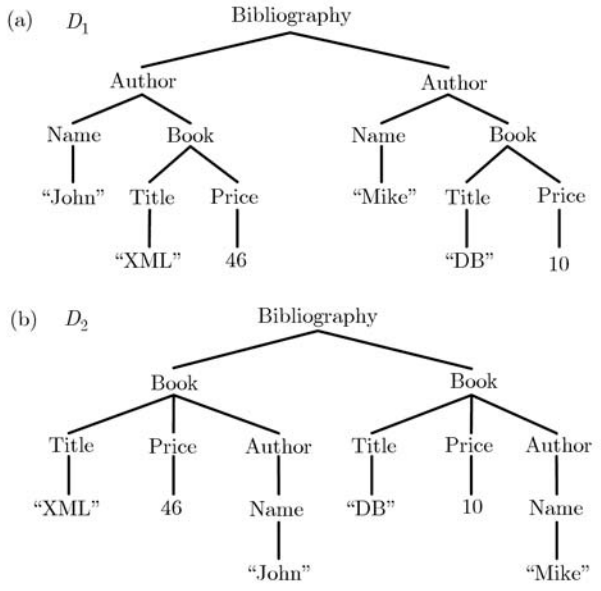



**Figure 1** An example data organization of different hierarchical structures.

In fact, XQuery and XPath allow no concrete precedence order between query nodes. For example,

$Q_2$: //author[child::name="Mike"]/descendant-or-self::*/ancestor-or-self::book/child::title,

can also be used to find title elements which are children of book elements which are written by an author named "Mike" from $D_2$. Although book and author should be on the same path, we know nothing about which one is the ancestor of the other or vice versa. The benefits of using $Q_2$ is obvious; it can be used to retrieve useful information from both $D_1$ and $D_2$ without considering their structural heterogeneity. However, such a query cannot be easily evaluated. Although Olteanu et al.[5,6] show that using special rules, XPath queries with reverse axes, e.g. $Q_2$, can be equivalently rewritten as a set of TPs, they also show that this transformation may lead to an exponential blowup of the number of TPs. Further, Gottlob et al.[7] show that the combined complexity of XPath is P-hard (i.e., hard for polynomial time).

Usually in many scenarios, we cannot specify the precedence relationships of query nodes when we formulate query expression. 1) The document structure is not available. 2) Extracting desired information from XML documents of structural heterogeneity. It is complex to use TPs in conjunction with data integration mapping rules between a global schema and local schema, and may cause errors since maintaining the mapping relationship may involve extensive manual intervention. 3) The change of business strategy and corporate environments may cause the data to be organized with a different structure, which makes existing path expression that depends on particular hierarchical structure no longer feasible.

Although keyword based methods[8,9] can be used freely without schema knowledge, only limited semantic constraints can be contained in such a query expression. Query relaxation based methods[10,11] will also produce a large number of relaxed query expressions by relaxation operations, which will further result in too many approximate answers.

In ref. [12], the notion of partially specified twig pattern (PSTP) was proposed to tackle this problem. Compared with TP, PSTP provides us with the most flexibility: 1) we can specify the full structural constraints if the schema or document structure is available; 2) we can specify just keywords to retrieve desired information if the schema or document structure is not available; and 3) we can make full use of whatever partial knowledge we have to specify more flexible semantic constraints.

As a PSTP may correspond to multiple TPs, a naive evaluation method[12] for PSTP is as follows:

Let $Q$ be a PSTP, $Q_1, Q_2, \ldots, Q_n$ be TPs derived from $Q$, $R, R_1, R_2, \ldots, R_n$ be the answer sets of $Q, Q_1, Q_2, \ldots, Q_n$ on an XML document $D$, respectively. Then $R = \bigcup_{i\in[1,n]} R_i$.

While PSTP can express more flexible semantic information, it is not feasible to directly apply it

in practice, because $n$ may be too large and thus has great impact on query performance. Our contributions are as follows:

1. We propose an extension to XPath by introducing two Samepath axes to enhance the expressiveness of XPath.

2. We give a detailed analysis of the challenges of evaluating PSTP, and then propose an efficient algorithm, pTwigStack, to process a PSTP holistically. Our method possesses the following three features: scanning only once; no redundant output; and bounded space complexity.

3. We implemente related algorithms and make comparison between our methods and existing ones. Experimental results demonstrate that our method is efficient in terms of various evaluation metrics.

## 2 Preliminaries

### 2.1 Data model and numbering schemes

An XML document can be modeled as a node-labeled tree, where nodes represent elements, attributes and text data, while edges represent direct nesting relationship between nodes in the tree. Formally, tree $T = (V, E, \Sigma, M)$, where $V$ is the node set and there is a unique root node $R$ in $V$, $E$ is the edge set, and no cycle among the edges is permitted, $\Sigma$ is an alphabet of labels and text values, $M$ is a function that maps each node to its label.

Most XML query processing algorithms use a special positional representation to represent an element; we use pre($v$), which is compatible with preorder numbering, to denote the numerical id assigned to node $v$, in the sense that if a node $v_1$ precedes a node $v_2$ in the preorder left-to-right depth-first traversal of the tree, then pre($v1$) <pre($v2$). This positional representation can be easily implemented using either region encoding[13] or Dewey ID[14]. In the first case, pre($v$) equals a tuple of three fields: ($start$; $end$; $level$). We say that element $u$ is an ancestor of element $v$ if and only if $u.start < v.start < u.end$. $u$ is the parent of $v$ if and only if $u.start < v.start < u.end$ and $u.level = v.level - 1$. In the second case, if $u$ is the root node, $label(u) = 1$, otherwise, $label(u) = label(v).x$, where $u$ is the $x$th child of $v$, and "." in "$label(v).x$" is the concatenation operator which is different from the "." in $u.start$, $u.end$ and $u.level$ in the previous sentences.

### 2.2 Twig pattern (TP) matching

TPs are used to match data fragments from XML data. The edges in a TP indicate either Parent-Child (P-C) or Ancestor-Descendant (A-D) relationship of query nodes. For convenience, we use "node" to denote query node and "element" to denote data element in an XML document.

Matching a TP against an XML document is to find all occurrences of the TP in the database. Formally, given a TP $Q$ and an XML document $D$, a match of $Q$ in $D$ is identified by a mapping from nodes in $Q$ to elements in $D$, such that: i) the query node predicates are satisfied by the corresponding database elements; ii) the structural relationships (P-C or A-D) among query nodes are satisfied by the corresponding database elements. The answer to query $Q$ with $n$ nodes can be represented as a $n$-array tuple $(e_1, e_2, \ldots, e_n)$ which consists of the database elements that identify a distinct match of $Q$ in $D$.

## 3 The Samepath axis

**Definition 1** (SamepathStep)**.** A SamepathStep returns a sequence of nodes that are reachable from the context node via a specified axis (PC-samepath or AD-samepath axis). The SamepathStep has two parts: an axis, which defines the "direction of movement" for the step, and a node test, which selects nodes based on their kind and name. The resulting node sequence is returned in document order.

**Definition 2** (PC-samepath axis ("→"))**.** The PC-samepath axis contains the set of data elements that are children and parent of the context node.

**Definition 3.** (AD-samepath axis ("⇒"))**.** The AD-samepath axis is the transitive closure of the PC-samepath axis; it contains the set of data elements that are descendant and ancestor of the context node.

There are totally 82 rules in the current XPath

grammar, among which only rules 26th and 28th have to be modified. As shown in Figure 2, SamepathStep is further defined by the new rules [n1] and [n2]. It consists of two axes, i.e. PC-samepath and AD-samepath. Except the two axes, we also introduce two separators, "→" and "⇒", to indicate the semantic constraints of being on the same path. Thus a path expression consisting of a series of step expressions may be separated by "→" or "⇒". For example, "book⇒author" is short for "child::book/AD-samepath::author" and will return all authors that are descendant or ancestor of book. As a result, there are totally 84 rules in the extended XPath grammar with two of them modified and two newly added. This extension to XPath provides users with the ability to specify the semantic constraints of two nodes on the same path in a very simple way. The Samepath axis is similar to the current XPath axes insofar as it returns a set of nodes corresponding to the context node. We can use a node test and predicates to filter those undesired nodes.

[26] RelativePathExpr ::= StepExpr (("/"|"//"|"->"|"=>")StepExpr)*
[27] StepExpr ::= FilterExpr|AxisStep
[28] AxisStep ::= (ReverseSetp|ForwardStep|***SamepathStep***)PredicateList
[n1] ***SamepathStep*** ::= ***SamepathAxis*** NodeTest
[n2] ***SamepathAxis*** ::= ("PC-samepath" "::")|("AD-samepath" "::")

**Figure 2** EBNF grammar for the extended XPath.

If both $A$ and $B$ are query nodes, we use "$A \rightarrow B$" to denote that $A$ is the parent node of $B$ or vice versa, "$A \Rightarrow B$" denotes that $A$ and $B$ are on the same path. If $e_A$ and $e_B$ are data elements of tag $A$ and $B$, "$e_A \rightarrow e_B$" or "$e_A \Rightarrow e_B$" denotes that $e_A$ and $e_B$ satisfy the structural constraints of "$A \rightarrow B$" or "$A \Rightarrow B$", respectively.

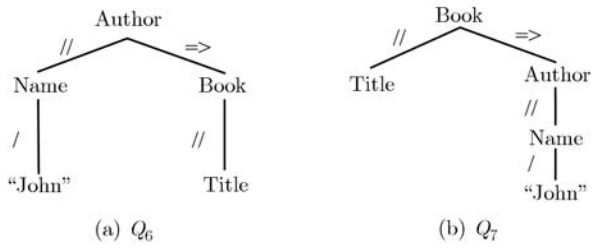



**Figure 3** Two partially specified twig patterns.

**Example 1.** Consider query $Q_3$: "find the title of the books that have an author named 'John' from the two documents in Figure 1". For $D_1$, the query should be $Q_4$: //author[.//name="John"]//book//title; for $D_2$, the query should be $Q_5$: //book[.//author//name="John"]//title. With Samepath axis, $Q_4$ and $Q_5$ can be replaced by a PSTP with Samepath axis in either Figure 3(a) or Figure 3(b), which can be written in extended XPath format as $Q_6$: author[.//name="John"]⇒book//title or $Q_7$: book[.⇒author//name="John"]//title. Further, PSTP expression can be seamlessly incorporated into XQuery. For $Q_3$, our solution using the Samepath axis is as follows.

```
for $a in doc()//author,
    $t in $a⇒book//title
where $a//name="John"
return $t
```

Although the Samepath axis can be used to specify the semantic constraints of two nodes on the same path, it only involves the relationship of two nodes. Considering the PSTP in Figure 4(a), we can easily understand that the semantics of $A \Rightarrow B$ equals $A//B$ or $B[.//A]$. Further, considering the PSTP expression $A \Rightarrow B \Rightarrow C$ and its derived TPs in Figure 4(b), where $A$ and $B$ should be on the same path and $B$ and $C$ should be on the same path, but not necessary for $A$ and $C$. $A \Rightarrow B \Rightarrow C$ may correspond to $B[.//A]//C$ since elements with tag $A$ and $C$ are not required to be on the same path. The PSTP in Figure 4(c) has not been specified with a concrete root node, and the related TPs are not shown due to limited space. A problem we should notice is that node $B$ in the PSTP in Figure 4(c) should be on the same path with $A$, and $D$ should be on the same path with $C$, which means that either $B$ or $D$ or both of them may be ancestors of $A$ and $C$.

## 4 Problems and our solutions

As we know from the above description that some PSTPs may not have been specified with a concrete root node, like the one in Figure 4(c), which corresponds to a keyword-like query, the difference is that each part of the PSTP is also a path expression that may contain partial semantic constraints.

Such a query can be evaluated easily by extending existing keyword based methods. We present in this paper only query processing method for PSTPs that are specified with concrete root nodes, like the ones in Figure 4(a) and (b). Moreover, all results presented in this section are based on A-D and AD-samepath relationships. In this section, we first analyze existing stack based TP matching method, and then show challenges of evaluating PSTP.

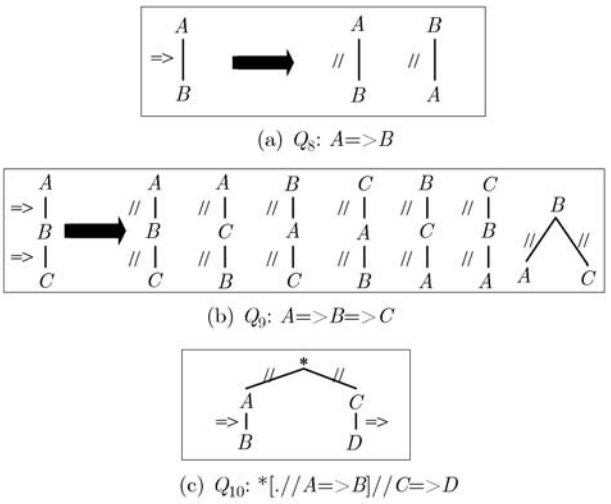



**Figure 4** Three PSTPs and their corresponding TPs.

### 4.1 Insight into the TwigStack algorithm

In the TwigStack algorithm, each query node $q$ in a TP is associated with a stack $S_q$, a cursor $C_q$ and a data stream $T_q$. $C_q$ can point to some elements in $T_q$, especially, we say that $C_q$ is NULL if all elements in $T_q$ are processed, and $C_q$ is also used to denote the element it points to. Before executing, all cursors point to the first elements in each data stream. We use Advance($C_q$) to make $C_q$ pointing to the next element. The self-explaining functions isRoot($q$) and isLeaf($q$) are used to determine whether $q$ is a root node or a leaf node. The function children($q$) is used to return all the child nodes of $q$ and parent($q$) is used to return the parent node of $q$.

TwigStack works in two steps. In the first step, it repeatedly calls getNext(root) to get a query node $q$ with Solution Extension[1)], and then $C_q$ is processed by either being pushed into stack as a useful element, or being skipped as a useless element. Such operations will repeat until all elements of leaf nodes are processed. At the end of this step, TwigStack will produce all path solutions. In the second step, all produced path solutions are merge-joined to get the final answers. When all edges in the TP are A-D edges, TwigStack guarantees that both its time and I/O complexity are independent of the size of partial matches to any root-to-leaf path.

In the TwigStack algorithm, the objective of getNext(root) is finding the first element that may participate in final answers from the elements that are still not being processed, and Solution Extension is used here to guide the execution of getNext(root). As shown in Figure 5, $Q_{\mathrm{AD}}$ is a TP with just A-D edges. Suppose $B$ is returned by getNext($A$). From the definition of Solution Extension we know that it guarantees that the structural constraints below $B$ are satisfied, which is denoted by (1) with arrows in Figure 5(a). Thus whether $C_B$ is a useful element is determined by just checking whether the top element in $S_A$ is an ancestor of $C_B$, which is denoted by (2) with an arrow. Obviously, Solution Extension, (1) with arrows, works in down direction in TwigStack; while useful element, (2) with an arrow, works in up direction.

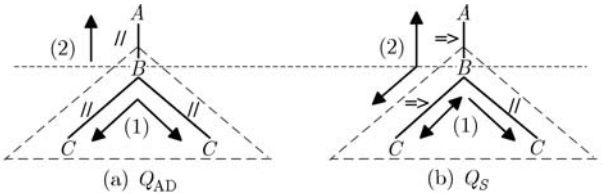



**Figure 5** Processing strategy for different methods.

**Example 2.** For $D_3$ and $Q_{11}$ in Figure 6, the first call of getNext($A$) in TwigStack will return $B$ with cursor $C_B$ pointing to $b_1$, which means that all descendant nodes of $B$ have Solution Extension. Among all cursors of descendant of $B$, $C_B$ has the smallest preorder value. However, as no element in $S_A$ is ancestor of $b_1$, $b_1$ is skipped as a useless

1) A node $q$ has a Solution Extension if there is a solution for the sub query rooted at $q$ composed entirely of the cursor elements of the query nodes in the sub query. Note that if $q$ has a Solution Extension, $C_q$ is the ancestor of all cursor elements in the sub query tree nodes, and pre($C_q$) is smaller than all other elements of query nodes in the subtree rooted at $q$, based on the strictly nested property of XML data.

element instead of being pushed into $S_B$. In this example, the useful elements are $a_1, b_2, b_3$ and $c_1$. After $c_1$ is processed, the stack encoding is shown in Figure 6(c) and then two path solutions are produced, they are $(a_1, b_2, c_1)$ and $(a_1, b_3, c_1)$.

Thus we have the following observations.

1. Query nodes in a TP are processed with a special order in existing methods, i.e. left-to-right depth-first traversing the TP. For $Q_{11}$, the order is $A, B, C$.

2. Query node $q$ returned by getNext(root) must have a Solution Extension, from which we can get an element $C_q$ for further processing. $C_q$ can be either pushed into stack if it can participate in final answers, or skipped directly as a useless one.

3. If $q$ is the root node or otherwise, $C_q$ satisfies the structural relationship of edge $< \text{parent}(q), q >$ with the top element in stack $S_{\text{parent}(q)}$ (if such element exists), then $C_q$ is a useful element, which means that $C_q$ can participate in final answers.

4. All elements in the same stack (from bottom to top) are guaranteed to lie on a root-to-leaf path according to the given XML document, and elements in different stacks are linked together through pointers (from descendant to ancestor).

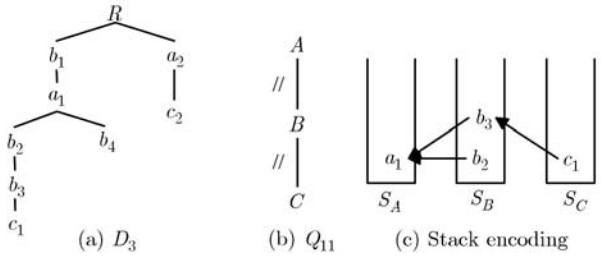



**Figure 6** An example of query processing of the TwigStack algorithm.

### 4.2 Challenges and our solutions

Although TwigStack guarantees that all elements are scanned only once and no redundant output, the four aspects described above hold only for a TP, not a PSTP. Similarly, we need to resolve the following problems when evaluating a PSTP.

4.2.1 Query node processing order. As a PSTP may correspond to multiple TPs, e.g. $Q_9$ in Figure 4 corresponds to 7 TPs, a naive way is processing each one of them using existing methods. Obviously, this will cause high processing cost. In our method, we process a Samepath axis without decomposing it into two P-C or A-D axes, thus we can process a PSTP without considering the derived TPs. In this way, the query node processing order for $Q_9$ is $A, B, C$.

4.2.2 Returning node. For the same reason, when processing a PSTP, Solution Extension cannot be used correctly in getNext as an indicator to tell whether an element is useless. For example, consider $Q_{11}$ and $D_3$ in Figure 6. If the current cursors $C_A, C_B$ and $C_C$ point to $a_2, b_4$ and $c_2$, we can skip $b_4$ directly since $b_4$ appears before $c_2$ and it is not the ancestor of $c_2$. But for $Q_9$ in Figure 4, $Q_{11}$ is just one of the derived TPs of $Q_9$. Although $b_4$ is useless for $Q_{11}$, for $Q_9$, however, we cannot say that $b_4$ is useless by just checking $Q_{11}$, $b_4$ may be useful for other derived TPs since $B$ also appears at leaf node in two derived TPs of $Q_9$. In this case, we should return $B$ for further processing to avoid losing answers of $Q_9$. For this problem, we propose a notion partial solution extension (PSE) to guide the execution of getNext. Intuitively, if $q$ has a PSE, $q$ corresponds to at least one derived TP in which $q$ has a Solution Extension, which means that $C_q$ may participate in final answers of these derived TPs.

4.2.3 Pushing element. In TwigStack, an element $C_q$ corresponding to $q$ returned by getNext(root) can be pushed into stack if $C_q$ can participate in at least one final result, i.e. $C_q$ is a useful element. Similarly, in our method, $C_q$ will be pushed into stack if and only if it is a useful element. Then what is a useful element for PSTP? In TwigStack, we only need to check whether the top element in $S_{\text{parent}(q)}$ satisfies the structural relationship of edge $< \text{parent}(q), q >$ with $C_q$. However, we need to check other related elements for PSTP so as not to lose any result. For example, consider $Q_9$ in Figure 4(b) and $D_4$ in Figure 7(a). Obviously, $a_1, b_1$ and $c_1$ are useful elements since $B[.//A]//C$ and $B//C[.//A]$ are two derived TPs of $Q_9$; thus the three elements are pushed into $S_A, S_B$ and $S_C$. After that, $C_A, C_B$ and $C_C$ point to $a_2, b_2$ and $c_2$, respectively. Because $A$ appears at leaf node of three derived TPs of $Q_9$, after $a_2$ is returned by getNext(root), we need to check whether $a_2$ can participate in final answers with elements in stacks.

Obviously, $a_2$ is a useful element for $Q_9$. For another example, consider $D_3$ in Figure 6 and $Q_9$ in Figure 4. Suppose $C_A, C_B$ and $C_C$ point to $a_2, b_4$ and $c_2$, respectively. Because $B$ also appears at leaf node of some derived TPs, $b_4$ is first returned by getNext(root) for further processing. Although $b_4$ can satisfy the structural constraint of $A \Rightarrow B$ with top($S_A$), i.e. $a_1 \Rightarrow b_4$, it is a useless element for $Q_9$ since no element with tag $C$ can satisfy the structural constraint of $B \Rightarrow C$. We propose a notion, Useful Element, in the next section as a metric to tackle this problem.

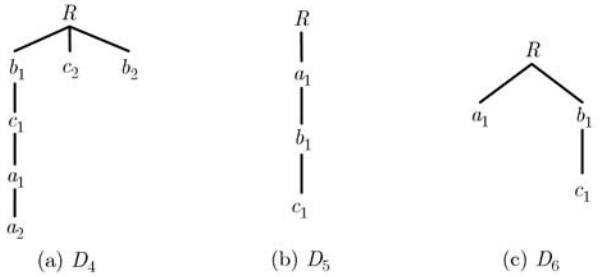



**Figure 7** Three XML documents.

4.2.4 Stack organization. For a PSTP, a query node is the ancestor of another one in some derived TPs; in other derived TPs, however, it may be a descendant of that node. In our method, data elements in the same stack (from bottom to top) lie on a root-to-leaf path in an XML document, and data elements in different stacks are linked together through pointers from elements in the stack of descendant query node to elements in stack of ancestor query node. For example, consider $Q_9$ in Figure 4 and $D_4$ in Figure 7. $a_1, b_1$ and $c_1$ satisfy the structural constraint of $A \Rightarrow B \Rightarrow C$, so they should be pushed into $S_A, S_B$ and $S_C$, respectively. Although $a_1$ is a descendant of $b_1$, the pointer between them starts with $b_1$ and ends at $a_1$.

## 5 Related notions

From the discussion in section 4 we know that the first problem of evaluating PSTPs is: in what condition a query node $q$ should be returned by getNext(root)? For example, $Q_S$ in Figure 5(b) is a PSTP, $B$ and $C$ have the Samepath relationship, so each of them can be a leaf node, which is denoted by (1) with arrows. Suppose $B$ is returned by getNext($A$) in our method. Then $C_B$ appears before $C_C$ and maybe they are not on the same path. We cannot say that, in this case, $C_B$ is useless. So the objective of getNext(root) can be stated as not checking whether an element is useful, but whether it is useless. If it is useless, the element will be skipped directly, otherwise, it will be returned for further processing. In the following definition, hasSE($q$) checks whether $q$ has a Solution Extension.

**Definition 4** (partial solution extension (PSE))**.** Let $Q$ be a PSTP, we say that a query node $q$ of $Q$ has a PSE if and only if $q$ satisfies any one of the following conditions:

1. isLeaf($q$)$\wedge C_q \neq$NULL, or,
2. for each $q' \in$ children($q$)

(1) $q//q' \wedge C_q//C_{q'} \wedge$hasSE($q'$), or,

(2) $q \Rightarrow q' \wedge$hasSE($q'$) $\wedge$ (pre($C_q$) $<$ pre($C_{q'}$) $\vee$ $C_{q'} \Rightarrow C_q$).

Case 1 is straightforward. Case 2 consists of two independent conditions. (1) means that if $q$ and $q'$ have A-D relationship, the current elements of $q$ and $q'$, i.e. $C_q$ and $C_{q'}$, should satisfy the structural constraint of $q//q'$, i.e. $C_q//C_{q'}$, at the same time, $q'$ should have Solution Extension. For (2), consider $D_5$ in Figure 7(b) and $Q_9$ in Figure 4(b), and suppose $C_A, C_B$ and $C_C$ point to $a_1, b_1$ and $c_1$, respectively. As $B$ and $C$ have the Samepath relationship, $C$ has a PSE and $b_1 \Rightarrow c_1$, which is equal to $q \Rightarrow q' \wedge$hasSE($q'$) $\wedge\, C_q \Rightarrow C_{q'}$, so we say that $B$ has a PSE since $b_1$ may participate in final answers. Consider $D_6$ in Figure 7(c) and $Q_9$, and suppose $C_A, C_B$ and $C_C$ point to $a_1, b_1$ and $c_1$, respectively. In this case, $A$ and $B$ have the Samepath relationship, $B$ has a PSE, and $a_1$ and $b_1$ are not on the same path, which is equal to $q \Rightarrow q' \wedge$hasSE($q'$) $\wedge$ pre($C_q$) $<$ pre($C_{q'}$) $\wedge \neg(C_q \Rightarrow C_{q'})$. However, we cannot say that $a_1$ is useless just according to the structural relationship between $a_1$ and $b_1$, because $A$ can be a leaf node in some derived TPs shown in Figure 4(b). In such a case, $A$ has a PSE and the union of the two cases is equal to (2).

By the definition of PSE, we can easily check whether a query node $q$ has PSE. However, $C_q$ may not participate in final answers and we need

to check whether $C_q$ is useful, which forms the second problem, i.e. in what condition should $C_q$ be pushed into stack $S_q$? From (2) with arrow in Figure 5(b) we know that this operation should include checking the satisfiability of all Samepath relationships that are directly related with $B$ from both up and down directions. We propose a notion, Useful Element, to answer this problem, where top($S_q$) returns the top element from stack $S_q$ and isEmpty($S_p$) checks whether $S_p$ is empty. For two elements $e_p$ and $e_q$, isHold($e_p, e_q, \langle p, q\rangle$) is used to check whether $e_p$ and $e_q$ satisfy the structural constraint between query nodes $p$ and $q$.

**Definition 5** (Useful element)**.** An element pointed by $C_q$ ($q$ is returned by getNext(root) in our method) is a useful element if and only if any one of the following conditions holds:

1. isRoot($q$)$\wedge$hasSE($q$), or,
2. isHold(top($S_{\text{parent}(q)}$), $C_q$, $\langle$parent($q$), $q\rangle$)$\wedge$

(1) hasSE($q$), or,

(2) for each child $q'$ of $q$, if isHold($C_q, C_{q'}, \langle q, q'\rangle$) = FALSE, then isHold($C_q$,top($S_{q'}$), $\langle q, q'\rangle$) = TRUE

Intuitively, $C_q$ is useful means that it can participate in final answers. The fact we should understand is that $q$ has a PSE does not means that it must have a Solution Extension. Because the Samepath axis is bidirectional in essence, if $q$ has not a Solution Extension, we need to check for each child node $q'$ of $q$, whether there exists in $S_{q'}$ elements that can satisfy the structural constraint between $q$ and $q'$ with $C_q$.

Case 1 means that if $q$ is the root node and $q$ has Solution Extension, then $C_q$ is a useful element. Case 2 means that if $q$ is not the root node, $C_q$ must satisfies the structural constraint between parent($q$) and $q$ with top($S_{\text{parent}(q)}$); moreover, $q$ must have a Solution Extension (shown as "(1)"), or otherwise, for any child node $q'$, if $C_q$ and $C_{q'}$ do not satisfy the structural constraints between $q$ and $q'$, then $C_q$ and the top element of $S_{p'}$ must satisfy the structural constraints between $q$ and $q'$. For example, consider $D_3$ in Figure 6(a) and $Q_9$ in Figure 4(b), and suppose $C_A, C_B$ and $C_C$ point to $a_2, b_4$ and $c_2$, respectively. From Definition 4 we know that $B$ has a PSE, and the top element $a_1$ in $S_A$ satisfies the structural constraint of $A \Rightarrow B$, i.e. $a_1 \Rightarrow b_4$. However, we still cannot say that $b_4$ is a useful element. As $b_4$ and $c_2$ do not satisfy the structural constraint of $B \Rightarrow C$, $B$ has not a Solution Extension. Further, $b_4$ does not satisfy the second condition "(2)" of "2", i.e. there is no element (in $S_C$) that can satisfy the structural constraint of $B \Rightarrow C$ with $b_4$. Therefore, $b_4$ is a useless element and can be safely discarded.

## 6 PSTP matching

Similarly to the TwigStack algorithm, in our method, each query node $q$ in the given PSTP is associated with a stack $S_q$, a cursor $C_q$ and a data stream $T_q$. $S_q$, $C_q$ and $T_q$ have the same meaning as that of TwigStack, and some functions used in our method are the same as that described in section 4.1.

### 6.1 Algorithm: pTwigStack

As shown in Algorithm 1, in the first phase (lines 1–8), as long as not all elements in element streams of leaf nodes are processed, getNext(root) is called repeatedly in line 2 to get a query node $q$ with a PSE. If $C_q$ is useful (determined by isUsefulEle($q$) in line 3), it will be pushed into $S_q$ in line 4, ModifyPointer($q$) is used to modify related pointers in line 5. In line 6, elements that are not processed and have a smaller numerical id value than pre($C_q$) are processed by calling ProcessOtherEle($q$), i.e. all elements appear before $C_q$ are processed all together with $C_q$ if their corresponding query nodes are descendant of $q$ in the given PSTP. In line 7, $C_q$ is moved to the next element in $T_q$. When an element is popped from stack, all its related path solutions are produced by calling cleanStack(). In line 8, all remaining path solutions will be produced by calling outputPaths(). In the second phase, these path solutions are merge-joined to compute the final answers by calling MergeAllPathSolution() in line 9. Noting that path solutions should be outputted in root-to-leaf order so that they can be easily merged together to form the final answers, so we need to block some path solutions during output, just as showSolutionsWithBlocking[1] does.

**Algorithm 1: pTwigStack**(root)

1: **while** ¬end(root) **do**
2: $q$ = getNext(root)
3: **if** isUsefulEle($q$) **then**
4: Push($C_q, S_q$, NULL)
5: ModifyPointer($q$)
6: ProcessOtherEle($q$)
7: Advance($C_q$)
8: outputPaths()
9: MergeAllPathSolution()

**Function: end**($q$)

1: **return** $\forall q_i \in$ subtreeNodes($q$): isLeaf($q_i$)∧end($C_{q_i}$)

**Procedure: ModifiyPointer**($q$)

1: **for** $p \in$ relatedNodes($q$) **do**
2: **if** $p//q \vee p \Rightarrow q$ **then** top($S_q$).$ptr$=top($S_p$)
3: **if** $q \Rightarrow p$ **then** top($S_p$).$ptr$=top($S_q$)

**Function: isUsefulEle**($q$)

1: bUseful=FALSE; bFlag=TRUE
2: **if** isRoot($q$)∧hasSE($q$) **then** bUseful=TRUE
3: **if** ¬isRoot($q$)∧isHold(top($S_{\text{parent}(q)}$), $C_q$, ⟨parent($q$), $q$⟩)
4: **if** hasSE($q$) **then** bUseful=TRUE
5: **else for each** $q' \in$ children($q$)
6: **if** ¬isHold($C_q, C_{q'}, \langle q, q' \rangle$)∧ ¬isHold($C_q$,top($S_{q'}$), $\langle q, q' \rangle$) **then** bFlag = FALSE
7: **if** bFlag=TRUE **then** bUseful=TRUE
8: **return** bUseful

**Procedure: Push**($C_q, S_q, ptr$)

1: push the pair ($C_q, ptr$) onto stack $S_q$

**Procedure: cleanStack**($S_p, C_q$)

1: Push all useful elements that are descendant of each popped element of $S_p$, which does not satisfy the structural relationship of $\langle p, q \rangle$ with $C_q$, then output related path solutions

**Procedure: ProcessOtherEle**($q$)

1: **for** $p \in$ children($q$) **do**
2: **while** $q \Rightarrow p \wedge$pre($C_q$) < $pre(C_p)$ **do**
3: cleanStack($S_q, C_p$)
4: **if** isUsefulEle($p$)=TRUE **then**
5: Push($C_p, S_p, C_q$)
6: ModifyPointer($p$)
7: ProcessOtherEle($p$)
8: Advance($C_p$)

getNext, as shown in Algorithm 2, is the core function called in pTwigStack, in which we need to consider A-D and the Samepath relationship simultaneously. getNext is used here to get a query node with a PSE, from which we can get an element that may participate in final answers. If $q$ is a leaf node, it will be returned directly in line 1. If not, in lines 2–5, for each child $p$ of $q$, if $p'$ (returned by getNext($p$)) is not equal to $p$, $p'$ is returned in line 4; otherwise, if $p'$ equals $p$ and $p$ has not a Solution Extension, $p$ is directly returned in line 5. If all children of $q$ have Solution Extension, we need to determine whether $q$ has a PSE. In lines 6–7, we find $n_{\min}$ and $n_{\max}$ which have the minimal and maximal numerical id value from all children that have A-D but not the Samepath relationship with $q$. In lines 8–9, $C_q$ is forwarded until $C_q$ and $C_{n\max}$ are on the same path or $C_{n\max}$ is before $C_q$. If pre($C_q$) > pre($C_{n\min}$), $n_{\min}$ is returned in line 10. In lines 11–12, for each child of $q$ that has the Samepath relationship with $q$, if $C_q$ cannot cover or be covered by $C_p$ and $C_p$ appears before $C_q$, $p$ is returned. Finally, if all children of $q$ satisfy the structural constraint with $q$ (hasSE($q$)=TRUE) or $C_q$ appears before $C_p$, i.e. ¬($C_q \Rightarrow C_p$)∧pre($C_q$) <pre($C_p$) (hasSE($q$)=FALSE), $q$ is returned in line 13.

**Algorithm 2: getNext**($q$)

1: **if** isLeaf($q$)=TRUE **then return** $q$
2: **for** $p \in$ children($q$) **do**
3: $p'$ = getNext($p$)
4: **if** $p' \neq p$ **then return** $p'$
5: **if** ¬hasSE($p$) **then return** $p$
6: $n_{\min}$ =minarg$_p${pre($C_p$)|$q//p$}
7: $n_{\max}$ =maxarg$_p${pre($C_p$)|$q//p$}
8: **while** pre($C_q$) <pre($C_{n\max}$)∧¬ ($C_q \Rightarrow C_{n\max}$) **do**
9: Advance($C_q$)
10: **if** pre($C_q$) >pre($C_{n\min}$) **then return** $n_{\min}$
11: **for** $p \in$ children($q$) **do**
12: **if** $q \Rightarrow p \wedge \neg(C_q \Rightarrow C_p) \wedge$pre($C_q$) >pre($C_p$) **then return** $p$
13: **return** $q$

**Example 3.** As shown in Figure 8, (a) is the given XML document $D$, (b) is a PSTP $Q$ and (c)–(f) are four TPs of $Q$. Initially, $C_A, C_B$ and $C_C$ point to $a_1, b_1$ and $c_1$, respectively. The first call of getNext($A$) returns $C$ with a PSE since $A$ and $C$ have A-D relationship and pre($c_1$) <pre($a_1$) ∧ ¬($a_1//c_1$). Because $c_1$ is useless, it is skipped directly and $C_C$ is moved to $c_2$. The second call of getNext($A$) returns $A$ with a PSE, then $a_1$ is pushed into $S_A$ and $b_1$ is also pushed into $S_B$ since pre($b_1$) <pre($a_1$) and $B \in$children($A$) and they are all useful elements. The statuses of $S_A, S_B$ and $S_C$ are shown in Figure 8(g). Though $B$ is a leaf

node, no path solution will be produced after $b_1$ is pushed into $S_B$ since $b_1$ is ancestor of $a_1$. After that, $C_A$ and $C_B$ point to $a_2$ and $b_2$, respectively. Thus $C_A, C_B$ and $C_C$ point to $a_2, b_2$ and $c_2$. In the third call of getNext($A$), $a_2$ is skipped directly and $C_A$ is moved forwardly to $a_3$ since $A$ and $C$ have A-D relationship and pre($a_2$) <pre($c_2$)$\wedge\neg(a_2//c_2)$. Then $C$ is returned with a PSE. Since $c_2$ is useful, it is pushed into stack $S_C$. The statuses of $S_A, S_B$ and $S_C$ are shown in Figure 8(h). The path solution $(a_1, c_2)$ is produced since $C$ is a leaf node and $c_2$ is a descendant of $a_1$. After that, $c_2$ is popped from $S_C$. The next call of getNext($A$) returns $B$ with a PSE, and $b_2$ will be pushed into $S_B$. The statuses of $S_A, S_B$ and $S_C$ are shown in Figure 8(i). The path solution $(a_1, b_2)$ is produced and $b_2$ is popped from $S_B$. After all elements are processed, all related path solutions are produced at the end of the first phase. They are $(a_1, b_1), (a_1, b_2)$ and $(a_1, c_2)$, respectively. In the second phase, all path solutions are merge-joined to get the final results, i.e. $(a_1, b_1, c_2)$ and $(a_1, b_2, c_2)$.

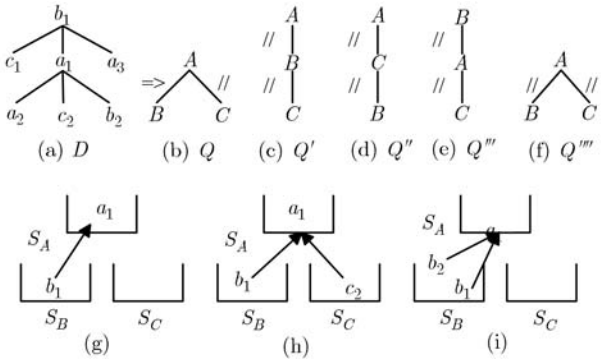



**Figure 8** Document $D$, PSTP $Q$ and its TPs ((a)–(f)), and running examples ((g)–(i)).

When P-C or PC-samepath edges appear in the given PSTP, we just need to take the level information of each element into account, the detailed description is omitted from Algorithm 1 for simplicity.

### 6.2 Analysis of pTwigStack

We first show the correctness of pTwigSatck and then analyze the complexity of pTwigStack.

**Lemma 1.** Let $Q$ be a PSTP, and $q$ be a query node of $Q$. If $q$=getNext(root), $q$ has a PSE.

**Lemma 2.** Any useful element $C_q$ will be pushed into stack $S_q$.

Obviously, Lemma 1 shows that from the query node $q$ returned by getNext(root) we can get an element $C_q$ that may be useful. Lemma 2 means that all useful elements are pushed into stacks. Let $Q$=$A$ op $B$, where op can be A-D ("//"), P-C ("/"), PC-samepath("$\rightarrow$") or AD-samepath("$\Rightarrow$") relationship. Because P-C and PC-samepath relationships can be easily processed based on A-D and AD-samepath relationships, we just show the correctness about A-D and AD-samepath relationships. If op="//", $Q = A//B$ and the subsequent operation is the same as that in TwigStack. If op="$\Rightarrow$", $Q = A \Rightarrow B$. As shown in Figure 9, we need to consider four cases: (a) $C_A//C_B$, which is consistent with $A \Rightarrow B$ and $A$ is returned first, the element processing order is $C_A, C_B$. (b) $C_B//C_A$, which is consistent with $A \Rightarrow B$ and $A$ is returned first. Further, element $C_B$ is processed simultaneously after $A$ is returned, and the element processing order is $C_A, C_B$. This case denotes that all elements appearing before $C_A$ are processed without another call of getNext($A$). (c) pre($C_B$) <pre($C_A$) $\wedge\neg(C_A \Rightarrow C_B)$. $B$ is returned first by getNext($A$) since $C_B$ appears before $C_A$. The element processing order is $C_B, C_A$. (d) pre($C_B$) >pre($C_A$) $\wedge\neg(C_A \Rightarrow C_B)$. In this case, $A$ is returned first by getNext($A$) since $C_A$ appears before $C_B$, and $A$ has a PSE, and the element processing order is $C_A, C_B$. In each case, if the processed element is useful, it will be pushed into $S_q$, otherwise, it will be discarded directly. Thus we have the following theorem.

**Theorem 1.** Let $Q$ be a PSTP. If each edge in $Q$ represents an A-D or AD-samepath relationship, then algorithm pTwigStack guarantees that only useful elements can be pushed into stack and each intermediate path solution can participate in final answers.

Theorem 1 means that each intermediate path solution produced by pTwigStack is useful when considering only A-D and AD-samepath relationship. The proof is simple. From Algorithm 1 we know that if an element is useless (line 3 in Algorithm 1), the cursor pointing to it will be forwarded to the next element (line 7 in Algorithm 1), otherwise, it will be pushed into stack according to Lemma 2. Further, if an element is useful, it must

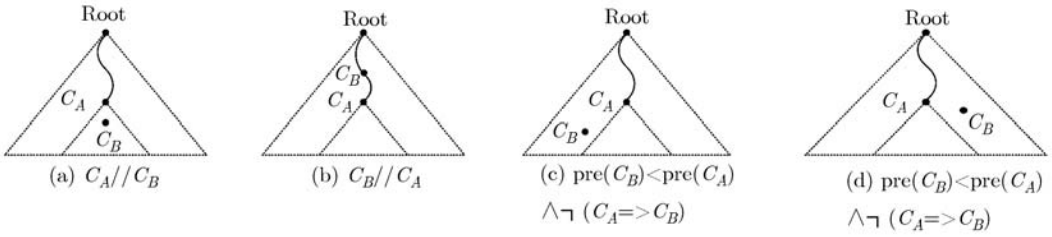



**Figure 9** Cases for pTwigStack.

satisfy the structural constraint of the given query with other elements in the running stacks; thus each intermediate path solution consisting of only useful elements will definitely participate in final answers, i.e. it is useful.

Since all useful elements are pushed into stacks according to Lemma 2, in the procedure cleanStack, path solutions are produced when elements are popped from stacks. Finally, in the second phase of pTwigStack, all path solutions are merge-joined to compute the final answers. So we have the following theorem.

**Theorem 2.** Given a PSTP $Q$ and an XML database $D$, algorithm pTwigStack correctly returns all answers for $Q$ on $D$.

While the correctness holds for any given PSTP $Q$, the I/O optimality holds only for the case where no P-C and PC-samepath edges exist in $Q$ as only useful elements are pushed into stacks. Therefore, we have the following result.

**Theorem 3.** Consider an XML database $D$ and a PSTP $Q$ that has $n$ nodes and just A-D and AD-samepath edges. Algorithm pTwigStack has worst case I/O and CPU time complexities linear in the sum of sizes of the $n$ input lists and the output list. Further, the worst case space complexity of Algorithm pTwigStack is the minimum of (i) the sum of sizes of the $n$ input lists, and (ii) $n$ times the maximum length of a root-to-leaf path in $D$.

Theorem 3 holds only for PSTP with A-D and AD-samepath edges, in the case where the PSTP contains P-C or PC-samepath edges. Algorithm pTwigStack is no longer guaranteed to be I/O and CPU time optimal. In particular, the algorithm might produce a solution for one root-to-leaf path that does not match any solution in another root-to-leaf path.

### 6.3 Optimization

If schema is not considered for query processing, obviously, pTwigStack definitely outperform TwigStack since a PSTP may correspond to multiple TPs. In this paper, we represent the underneath schema $S$ as a directed graph, where each node corresponds to a tag name, and each edge from node $A$ to node $B$ means that in the document complying with $S$, elements with tag $B$ can be children of elements with tag $A$. We say there exists a cycle between $A$ and $B$ in $S$ if there exists at least a path from $A$ to $B$ and vice versa, which means that elements with tag $A$ can be ancestor or descendant of elements with tag $B$. However, if no cycle exists between two query nodes in $S$, the Samepath relationship between the two nodes in a PSTP can be replaced by just one P-C or A-D relationship. If we can make use of such structural information, then query performance can be improved significantly.

Let a PSTP $Q = \{V, E, T\}$, where $V$ is the set of nodes in $Q$, $E \subseteq V \times V$ is the set of edges in $Q$, and $T : E \rightarrow R$ is a type function that maps each edge to a value in the relationship set $R$={'/','//','→','⇒'}. We propose Algorithm 3 to optimize a PSTP using the schema $S$.

In Algorithm 3, hasCycle$(A, B, S)$ is used to check whether there exists a cycle between $A$ and $B$ in schema $S$. For each edge $\langle A, B\rangle \in Q.E$, if $A$ and $B$ are connected by a PC-samepath edge and there is no cycle between $A$ and $B$ (line 3), the relationship between $A$ and $B$ can be replaced by P-C edge (lines 4–5). Similarly, if $A$ and $B$ are connected by an AD-samepath edge and there is no cycle between $A$ and $B$ (line 6), the relationship between $A$ and $B$ can be replaced by A-D edge (lines 7–8). In line 9, the optimized PSTP $Q$ is returned. Compared with the PSTP input into

Algorithm 3, the number of derived TP is reduced significantly, as shown in Figure 10. For simplicity, we call Algorithm 3 the 1st optimization technique.

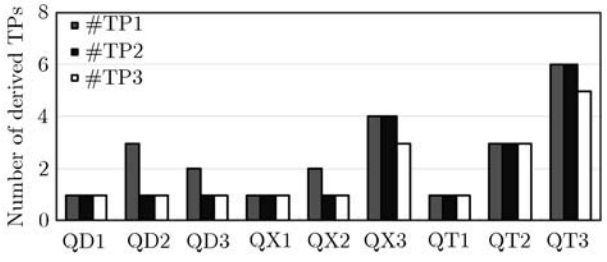



**Figure 10** Number of derived TPs for each query. #TP1, Number of derived TPs without optimization; #TP2, number of derived TPs after using the 1st optimization technique; #TP3, number of remained TPs after using the 2nd optimization technique.

**Algorithm 3: Rewrite**$(Q, S)$ /*$Q$ is a PSTP, $S$ is the schema graph*/

1: **for** $\forall A, B \in Q.V$ **do**
2: **if** $\langle A, B\rangle \in Q.E$ **then**
3: **if** $T(\langle A, B\rangle) =$ ' $\rightarrow$' $\wedge\neg$hasCycle$(A, B, S)$ **then**
4: $T(\langle A, B\rangle) \leftarrow$'/'
5: '$A \rightarrow B$' is replaced by '$A/B$' or '$B/A$'
6: **if** $T(\langle A, B\rangle)$='$\Rightarrow$' $\wedge\neg$hasCycle$(A, B, S)$ **then**
7: $T(\langle A, B\rangle) \leftarrow$ '//'
8: '$A \Rightarrow B$' is replaced by '$A//B$' or '$B//A$'
9: **return** $Q$

**Example 4.** Assume that the DTD schema of the given XML document consists of two rules, namely $\langle$!ELEMENT $A$ $(A, B, C)\rangle$ and $\langle$!ELEMENT $B$ $(A*, C)\rangle$. Obviously, there exists a cycle between $A$ and $B$, not $B$ and $C$. If the given PSTP is $Q_9$ in Figure 4 (b), by Algorithm 3, it can be rewritten as $Q_{9'}$: $A \Rightarrow B//C$, through which we can get four TPs, they are $A//B//C, B//A//C, B//C[.//A]$ and $B[.//A]//C$. Note that after optimization using Algorithm 3, the cost of evaluating $Q_9$ is greatly reduced since many derived TPs of $Q_9$ are no longer existent.

Further, schema information can also be used to check whether a PSTP or derived TP is satisfied, i.e. the answer set is not empty. Thus we can further reduce query processing cost by checking whether the given query is consistent with the schema information (i.e. Definition 6). We implemented such an operation in Algorithm 4, which we call the 2nd optimization technique.

**Definition 6.** Letting a PSTP $Q = \{V, E, T\}$, we say that $Q$ is consistent with schema $S$ if for each edge $\langle A, B\rangle \in Q.E$, we can find two nodes $A'$, $B'$ that satisfy Lable$(A)$ =Lable$(A')$ and Lable$(B)$ =Lable$(B')$ in $S$, such that the relationship of $A'$ and $B'$ satisfies the structural constraint of $T(\langle A, B\rangle)$ according to $S$, which corresponds to four cases: 1) if $T(\langle A, B\rangle)$ ='/', there exists an edge from $A'$ to $B'$; 2) if $T(\langle A, B\rangle)$='//', there exists a path from $A'$ to $B'$; 3) if $T(\langle A, B\rangle)$='$\rightarrow$', there exists an edge from $A'$ to $B'$ or vice versa; and 4) if $T(\langle A, B\rangle)$ ='$\Rightarrow$', there exists a path from $A'$ to $B'$ or vice versa.

**Algorithm 4: isConsistent**$(Q, S)$

1: **for** $\forall A, B \in Q.V$ **do**
2: **if** $\langle A, B\rangle \in Q.E$ **then**
3: find two nodes $A'$, $B'$ corresponding to $A$ and $B$ in $S$
4: **if** $T(\langle A, B\rangle)$ ='/' $\wedge\neg$hasEdge$(A', B', S)$ **then**
5: **return** FALSE
6: **if** $T(\langle A, B\rangle)$ ='//' $\wedge\neg$hasEdge$(A', B', S)$ **then**
7: **return** FALSE
8: **if** $T(\langle A, B\rangle)$ ='$\rightarrow$' $\wedge$ $\neg$(hasEdge$(A', B', S)\vee$hasEdge$(B', A', S)$) **then**
9: **return** FALSE
10: **if** $T(\langle A, B\rangle)$ ='$\Rightarrow$' $\wedge$ $\neg$(hasPath$(A', B', S)\vee$hasPath$(B', A', S)$) **then**
11: **return** FALSE
12: **return** TRUE

Algorithm 4 is used to determine whether the given query is consistent with the underneath schema, where hasEdge$(A, B, S)$ is used to check whether there is an edge from $A$ to $B$ in $S$. And hasPath$(A, B, S)$ is used to check whether there is a path from $A$ to $B$. For each edge $\langle A, B\rangle$ in $Q.E$, we find two nodes $A'$, $B'$ that correspond to $A$ and $B$ in $S$. In lines 4–11, we check whether the relationship between $A'$ and $B'$ in $S$ is satisfied with the structural constraint between $A$ and $B$ in $Q$, which corresponds to the four cases in Definition 6. If any one of the four cases does not hold, Algorithm 4 is terminated with FALSE returned, meaning that $Q$ is not consistent with $S$; otherwise, TRUE is returned in line 12, which means that $Q$ is consistent with $S$.

Assume that the DTD schema is the same to that of Example 4. Consider the four TPs derived from $Q_{9'}$, i.e. $A//B//C, B//A//C, B//C[.//A]$

and $B[.//A]//C$. Since no elements can appear below elements with tag $C$ according to the DTD schema, we can safely discard $B//C[.//A]$ as it is not consistent with the DTD schema. Thus only three TPs are left for further processing.

## 7 Experimental evaluation

### 7.1 Experimental setup

Our experiments were implemented on a PC with Pentium4 2.8 GHz CPU, 512 MB memory, 160 GB IDE disk, and Windows XP professional as the operation system.

We used TwigStack as the basis when implementing the naive method proposed in ref. [12], which we call nTwigStack (nTS). In addition to nTS and pTwigStack (pTS), we also implemented five other algorithms using three optimization techniques including B+ tree index, the 1st and 2nd optimization techniques (i.e. Algorithm 3 and Algorithm 4). B+ tree is used to index element labels so that we can skip useless element labels in label stream like TSGeneric+[2]. The 1st optimization is used to rewrite the given PSTP and the 2nd optimization is used to check whether a given query is consistent with the underneath schema. All these algorithms are listed in Table 1, where nTS-O is an improved version of nTS using the 1st optimization technique, nTS-OB is the combination of nTS, the 1st optimization technique and B+ tree index, and nTS-OBO is the combination of nTS, B+ tree index, the 1st and the 2nd optimization techniques. Similarly, pTS-O is an improved version of pTS using the 1st optimization technique, pTS-OB is the combination of pTS, the 1st optimization technique and B+ tree index. All algorithms were implemented using Visual C++ 6.0.

### 7.2 Datasets and queries

We used XMark (http://monetdb.cwi.nl/xml), DBLP (http://www.cs.washington.edu/research/xmldatasets/www/repository.html) and TreeBank for our experiments. The main characteristics of the three datasets can be found in Table 2. Although PSTP can be used to extract useful information from multiple XML documents with structural heterogeneity, we use one data set with recursive structure to simulate multiple data sources with different structures.

In our experiment, each element is labeled with a triple (start, end, level) and then stored into two separate files, one is sequential file, and the other is random file (disk-based B+ tree index). All labels corresponding to same tag are stored together in a label stream in an ascending order according to start value of each label. Sequential file is used in nTS, nTS-O, pTS and pTS-O algorithms without B+ tree index, and random file is used in nTS-OB, nTS-OBO and pTS-OB algorithms with B+ tree index. Each query node with a distinct tag name corresponds to a separate label stream.

The queries used in our experiment are listed in Table 3. Since the 2nd optimization technique is just used for the derived TPs in our experiment, we show in Table 3 only the changes caused by the 1st optimization technique. The benefits of the 2nd optimization is shown in Figure 10. All these queries can be classified into three categories: 1) PSTPs without the Samepath edges ("⇒" and "→"), e.g. QD1, QX1 and QT1, which we call 1st group PSTPs. 2) PSTPs can be transformed to TPs based on DTD schema using the 1st optimization, e.g. QD2, QD3 and QX2, which we call the 2nd group. 3) PSTPs cannot be transformed to TPs, which means that some Samepath edges cannot be replaced by P-C or A-D edges since there exist cycles in the schema between the nodes connected by these Samepath edges, e.g. QX3, QT2 and QT3, which we call the 3rd group.

**Table 1** Algorithms and optimization techniques used in our experiment

| | nTS | nTS-O | nTS-OB | nTS-OBO | pTS | pTS-O | pTS-OB |
|---|---|---|---|---|---|---|---|
| The 1st optimization | | √ | √ | √ | | √ | √ |
| $B^+$ tree index | | | √ | √ | | | √ |
| The 2nd optimization | | | | √ | | | |

**Table 2** Statistics of XML data sets

| Dataset | Size (M) | Nodes (Million) | Max depth | Average depth |
|---|---|---|---|---|
| DBLP | 127 | 3.3 | 6 | 2.9 |
| XMark | 113 | 1.7 | 12 | 5.5 |
| TreeBank | 82 | 2.4 | 36 | 7.8 |

**Table 3** Queries used in our experiment[a)]

| | Dataset | Status | Queries | Group |
|---|---|---|---|---|
| QD1 | DBLP | BO | //book/author | 1 |
| | | AO | -no change- | |
| QD2 | DBLP | BO | //www[.⇒editor]/url | 2 |
| | | AO | //www[.//editor]/url | |
| QD3 | DBLP | BO | //dblp⇒article/year | 2 |
| | | AO | //dblp//article/year | |
| QX1 | XMark | BO | //site/people/person/name | 1 |
| | | AO | -no change- | |
| QX2 | XMark | BO | /site⇒closed_auctions/emph | 2 |
| | | AO | /site//closed_auctions/emph | |
| QX3 | XMark | BO | //listitem[.//bold]/text[.//emph]⇒keyword | 3 |
| | | AO | -no change- | |
| QT1 | TreeBank | BO | //S/VP/VBD | 1 |
| | | AO | -no change- | |
| QT2 | TreeBank | BO | //S[.⇒JJ]/NP | 3 |
| | | AO | -no change- | |
| QT3 | TreeBank | BO | //S⇒VP/PP[NP/VBN]/IN | 3 |
| | | AO | -no change- | |

a) BO, Before the 1st optimization; AO, after the 1st optimization.

### 7.3 Performance comparison and analysis

We consider the following performance metrics to compare the performance of different algorithms: 1) number of derived TPs, which reflects the effect of using the 1st and 2nd optimization techniques; 2) number of scanned elements; 3) running time; and 4) scalability.

Consider the first metric, i.e. number of derived TPs, as shown in Figure 10, for the 1st group of PSTPs, optimization equals no optimization since no Samepath edge is contained in the three PSTPs. For the 2nd group of PSTPs, the 1st optimization will cause the Samepath edge replaced by A-D edge, the 2nd optimization technique will not bring us any benefits since each PSTP is consistent with the schema. For PSTPs in the 3rd group, the 1st optimization technique does not work in such a case since the three PSTPs are consistent with the schema and the Samepath edge in each PSTP cannot be replaced by A-D or P-C edge, but the 2nd optimization technique does work since some derived TPs of QX3 and QT3, except QT2, are not consistent with the schema; thus they can be safely

discarded without further processing.

Consider the 1st group of PSTPs, as shown in Figure 11(a), the number of scanned elements of nTS, nTS-O, pTS and pTS-O are same as each other, because neither of the two optimization techniques works for the 1st group of PSTPs, and for a given query, each algorithm uses the same set of label streams (sequential files); thus they will read the same amount of element labels. Figure 11(b) shows that our method, pTS and pTS-O, need to afford additional CPU cost since our method has more judging operations. This conclusion also holds for the comparison of nTS-OB, nTS-OBO and pTS-OB. However, as shown in the following, this performance degradation can be safely ignored when compared with the significant performance improvement achieved from the 3rd group of PSTPs.

For the 2nd group of PSTPs, as shown in Figure 12, the number of scanned elements of nTS is more than that of other algorithms since nTS will process each derived TP of the given PSTP without any optimization; as a result, running time of nTS is also very large compared with other algorithms. For the remainder algorithms, we can see from Figure 12 that algorithms with same basic configuration (label streams and optimization techniques) have similar performance because after the 1st optimization, each one of the 2nd group of PSTPs will be transformed to a TP, and the 2nd optimization doesn't work for each PSTP after transformation. Note using B+ tree index to skip useless elements will greatly improve query performance for QD2 rather than QD3. For QD2, only few elements are useful, so the running time of nTS-OB, nTS-OBO and pTS-OB is less than that of nTS-O, pTS and pTS-O. For QD3, although the number of scanned elements is reduced after using B+ tree index, repeatedly searching from root to leaf node makes the I/O cost of nTS-OB, nTS-OBO and pTS-OB larger than that of nTS-O, pTS and pTS-O, which will further result in poor query performance.

For the 3rd group of PSTPs, as shown in Figure 13, for any metrics in this figure, our methods, i.e. pTS, pTS-O and pTS-OB, outperform nTS, nTS-O, nTS-OB and nTS-OBO significantly, because the 1st optimization technique is useless for the 3rd group of PSTPs and each PSTP corresponds

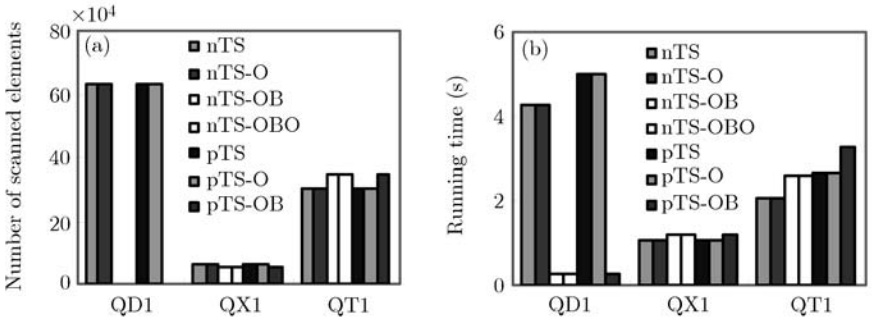



**Figure 11** Performance comparison against the 1st group of PSTPs. (a) Number of scanned elements; (b) runnin time.

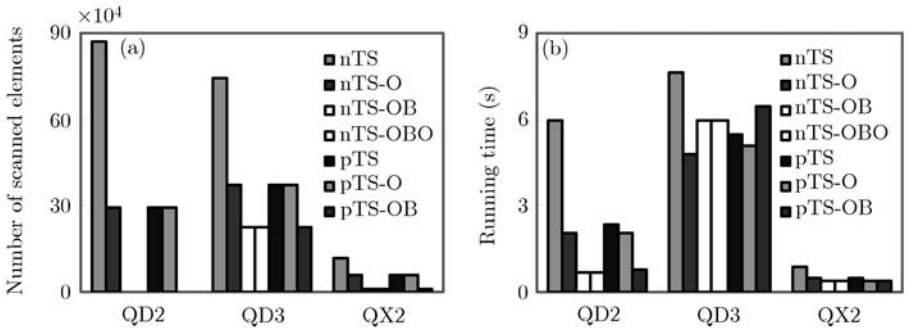



**Figure 12** Performance comparison against the 2nd group of PSTPs. (a) Number of scanned elements; (b) running time.

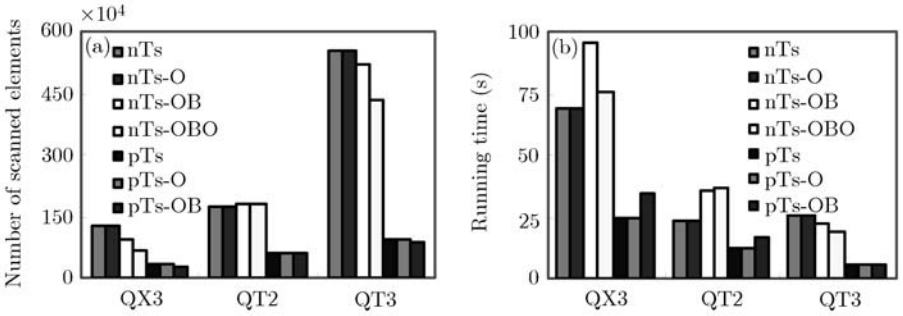



**Figure 13** Performance comparison against the 3rd group of PSTPs. (a) Number of scanned elements; (b) running time.

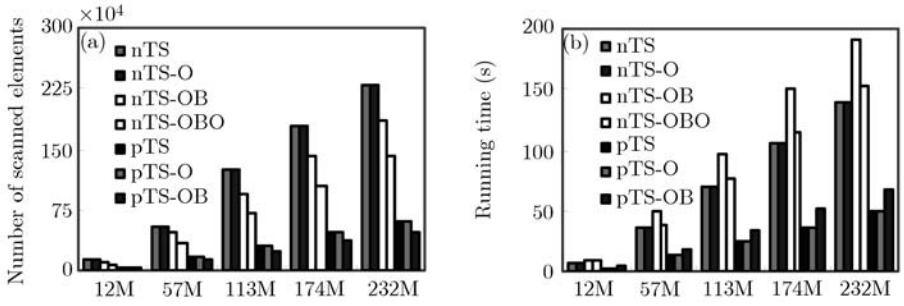



**Figure 14** Performance comparison of seven algorithms over XMark of different sizes using QX3. (a) Number of scanned elements; (b) running time.

to multiple TPs (Figure 10), among which only limited TPs can be safely discarded using the 2nd optimization technique, thus all remainder TPs after the 2nd optimization will be processed one by one using nTS, nTS-O, nTS-OB or nTS-OBO algorithm, as a result, large amount of elements need to be scanned multiple times, which will result in high CPU cost. Although the 2nd optimization technique is useful in such a case, the effect is limited for the 3rd group of PSTPs. Obviously, naive method, nTS, and its improved algorithms, nTS-O, nTS-OB and nTS-OBO, cannot work efficiently for the 3rd group of PSTPs.

Further, we present the performance results about scalability of QX3 in Figure 14, from which we know that our methods can work more efficiently than naive methods when processing PSTP over XML document with different size, because our method processes each element only once, while the performance of naive methods is determined by the number of derived TPs.

From the above experimental results and our analysis we know that when processing PSTPs with Samepath edges, especially the effect of optimization is not remarkable, e.g. QX3, QT2 and QT3, our methods, e.g. pTS, pTS-O and pTS-OB, can work much more efficiently than naive methods, i.e. nTS, and its optimization, i.e. nTS-O, nTS-OB and nTS-OBO. The reason lies in two aspects: 1) our methods guarantee that each element is scanned only once; and 2) our methods guarantee that no useless intermediate paths will be produced when considering only A-D and AD-samepath relationships (Theorem 1). Even if no Samepath edge appears in the query expression, e.g. QD1, QX1 and QT1, or otherwise, the Samepath edges can be replaced by P-C or A-D edges after optimization, e.g. QD2, QD3 and QX2, our method can achieve similar performance to that of the existing methods.

## 8 Related work

TPs can be used to match data fragments in an XML document. MPMGJN[13] was proposed for efficient structural join, Stack-Tree-Desc/Anc[15] improves the query performance of MPMGJN by using stack-based binary structural join algorithm.

Wu et al.[16] studied the problem of binary join order selection for complex queries based on a cost model. All these structure join methods suffer from the large number of intermediate results. To process a TP holistically, many methods[1−4] were proposed to avoid producing large size of useless intermediate results. Among them, TwigStack[1] was proposed to process a TP in a holistic way. When considering only A-D edges, TwigStack guarantees that the CPU time and I/O complexity is independent of the size of partial matches to any root-to-leaf path. Other methods[2−4] made improvements against TwigStack from different aspects. TSGeneric+[2] focused on holistic twig joins on all/partly indexed XML documents to skip some useless elements. Chen et al.[3] proposed iTwigJoin that exploits different data partition strategies to further boost the holism. TJEssential[4] uses a hybrid strategy to avoid redundant operations compared with the methods of refs. [1–3]. All these methods can only be used to a single TP, for a PSTP, however, they cannot work efficiently since a PSTP may correspond to several TPs.

Keyword based methods[8,9] can provide us with the most flexibility. MLCAS was introduced in ref. [9] to reduce meaningless results. XSEarch[8] returns semantically related document fragments that satisfy the user's query. However, we can not specify that several nodes are on the same path without specifying the concrete precedence relationship. Query relaxation based methods[10,11] will produce a large number of relaxed query expressions, thus resulting in too many approximate answers.

In the area of integrating tree-structured data, the Xyleme system[17] exploits XML views to cope with the problem. The Agora system[18] translates query expressions to SQL queries on each local data source. In ref. [19], queries are processed using mapping rules between a global schema and many local data sources, and then evaluated in each data source.

Although the notion of PSTP has been proposed in ref. [12] to provide the users with a more flexible way to express semantic constraints, no existing work has focused on holistic query evaluation for PSTPs. In ref. [20], the authors proposed a method for evaluation of partial path queries, not a query expression with branch node, thus we do not compare them with this method since it is based on different query syntaxes.

## 9 Conclusions

In this paper, we firstly extended XPath language with the new axis, Samepath axis, which allows users to express partial semantic constraints of being on the same path in a concise but effective way; then we proposed a new holistic query processing method, pTwigStack, for semantically querying tree-structured data sources using PSTPs. Our experimental results show that our method can work more efficiently than the existing methods when processing PSTPs. As our method cannot process a PSTP that is not specified with a root node, we will focus on this problem in the near future.


1 Bruno N, Koudas N, Srivastava D. Holistic twig joins: optimal XML pattern matching. In: Michael J F, Bongki M, Anastassia A, eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison: ACM, 2002. 310–321

2 Jiang H, Wang W, Lu H, et al. Holistic twig joins on indexed XML documents. In: Freytag J C, Lockemann P C, Abiteboul S, et al., eds. Proceedings of 29th International Conference on Very Large Data Bases. Berlin: Morgan Kaufmann, 2003. 273–284

3 Chen T, Lu J, Ling T W. On boosting holism in XML twig pattern matching using structural indexing techniques. In: Fatma Ö, ed. Proceedings of the ACM SIGMOD International Conference on Management of Data. Baltimore: ACM, 2005. 455–466

4 Li G, Feng J, Zhang Y, et al. Efficient holistic twig joins in Leaf-to-Root combining with Root-to-Leaf way. In: Ramamohanarao K, Krishna P R, Mohania M K, et al., eds. Proceedings of 12th International Conference on Database Systems for Advanced Applications. Bangkok: Springer, 2007. 834–849

5 Olteanu D. Forward node-selecting queries over trees. ACM Trans Database Syst, 2007, 32(1): 75–111

6 Olteanu D, Meuss H, Furche T, et al. XPath: looking forward. In: Chaudhri A B, Unland R, Djeraba C, et al., eds. EDBT 2002 Workshops XMLDM, MDDE, and YRWS. Prague: Springer, 2002. 109–127

7 Gottlob G, Koch C, Pichler R. The complexity of XPath query evaluation. In: Alin D, ed. Proceedings of the Twenty-third

ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. San Diego: ACM, 2003. 179–190
8 Cohen S, Mamou J, Kanza Y, et al. XSEarch: a semantic search engine for XML. In: Freytag J C, Lockemann P C, Abiteboul S, et al., eds. Proceedings of 29th International Conference on Very Large Data Bases. Berlin: Morgan Kaufmann, 2003. 45–56
9 Li Y, Yu C, Jagadish H V. Schema-Free XQuery. In: Nascimento M A, Özsu M T, Kossmann D, et al., eds. Proceedings of the 30th International Conference on Very Large Data Bases. Toronto: Morgan Kaufmann, 2004. 72–83
10 Sihem A Y, Koudas N, Marian A, et al. Structure and content scoring for XML. In: Böhm K, Jensen C S, Haas L M, et al., eds. Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim: ACM, 2005. 361–372
11 Sihem A Y, Cho S R, Srivastava D. Tree pattern relaxation. In: Jensen C S, Jeffery K G, Pokornÿ J, et al., eds. Proceedings of 8th International Conference on Extending Database Technology. Prague: Springer, 2002. 496–513
12 Theodoratos D, Souldatos S, Dalamagas T, et al. Heuristic containment check of partial tree-pattern queries in the presence of index graphs. In: Yu P S, Tsotras V J, Fox E A, et al., eds. Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management. Virginia: ACM, 2006. 445–454
13 Zhang C, Naughton J F, DeWitt D J, et al. On supporting containment queries in relational database management systems. In: Walid G A, ed. Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Barbara: ACM, 2001. 425–436
14 Tatarinov I, Viglas S, Beyer K S, et al. Storing and querying ordered XML using a relational database system. In: Franklin M J, Moon B, Ailamaki A, eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison: ACM, 2002. 204–215
15 Shurug A K, Jagadish H V, Jignesh M P, et al. Structural joins: a primitive for efficient XML query pattern matching. In: Umeshwar D, ed. Proceedings of the 18th International Conference on Data Engineering. San Jose: IEEE Computer Society, 2002. 141–152
16 Wu Y, Jignesh M P, Jagadish H V. Structural join order selection for XML query optimization. In: Dayal U, Ramamritham K, Vijayaraman T M, eds. Proceedings of the 19th International Conference on Data Engineering. Bangalore: IEEE Computer Society, 2003. 443–454
17 Cluet S, Veltri P, Vodislav D. Views in a large scale XML repository. In: Apers P M G, Atzeni P, Ceri S, et al., eds. Proceedings of 27th International Conference on Very Large Data Bases. Roma: Morgan Kaufmann, 2001. 271–280
18 Manolescu I, Florescu D, Kossmann D. Answering XML queries on heterogeneous data sources. In: Apers P M G, Atzeni P, Ceri S, et al., eds. Proceedings of 27th International Conference on Very Large Data Bases. Roma: Morgan Kaufmann, 2001. 241–250
19 Christophides V, Cluet S, Siméon S. On wrapping query languages and efficient XML integration. In: Chen W, Naughton J F, Bernstein P A, eds. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Texas: ACM, 2000. 141–152
20 Souldatos S, Wu X, Theodoratos D, et al. Evaluation of partial path queries on XML data. In: Silva M J, Laender A H F, Baeza-Yates R A, et al., eds. Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management. Lisbon: ACM, 2007. 21–30

IEEE International Conference on Data Engineering (ICDE2009):
517-528, March 29- April 3, 2009, Shanghai, China

# Effective XML Keyword Search with Relevance Oriented Ranking

Zhifeng Bao #, Tok Wang Ling #, Bo Chen #
#School of Computing
National University of Singapore
{baozhife, lingtw, chenbo}@comp.nus.edu.sg

Jiaheng Lu *
*School of Information and DEKE, MOE
Renmin University of China
jiahenglu@gmail.com

*Abstract*— **Inspired by the great success of information retrieval (IR) style keyword search on the web, keyword search on XML has emerged recently. The difference between text database and XML database results in three new challenges: (1) Identify the user search intention, i.e. identify the XML node types that user wants to search for and search via. (2) Resolve keyword ambiguity problems: a keyword can appear as both a tag name and a text value of some node; a keyword can appear as the text values of different XML node types and carry different meanings. (3) As the search results are sub-trees of the XML document, new scoring function is needed to estimate its relevance to a given query. However, existing methods cannot resolve these challenges, thus return low result quality in term of query relevance.**

**In this paper, we propose an IR-style approach which basically utilizes the statistics of underlying XML data to address these challenges. We first propose specific guidelines that a search engine should meet in both search intention identification and relevance oriented ranking for search results. Then based on these guidelines, we design novel formulae to identify the search for nodes and search via nodes of a query, and present a novel XML TF*IDF ranking strategy to rank the individual matches of all possible search intentions. Lastly, the proposed techniques are implemented in an XML keyword search engine called XReal, and extensive experiments show the effectiveness of our approach.**

## I. Introduction

The extreme success of web search engines makes keyword search the most popular search model for ordinary users. As XML is becoming a standard in data representation, it is desirable to support keyword search in XML database. It is a user friendly way to query XML databases since it allows users to pose queries without the knowledge of complex query languages and the database schema.

Effectiveness in term of result relevance is the most crucial part in keyword search, which can be summarized as the following three issues in XML field.

**Issue 1**: It should be able to effectively identify the type of target node(s) that a keyword query intends to search for. We call such target node as a *search for node*.

**Issue 2**: It should be able to effectively infer the types of condition nodes that a keyword query intends to search via. We call such condition nodes as *search via nodes*.

**Issue 3**: It should be able to rank each query result in consideration of the above two issues.

The first two issues address the search intention problem, while the third one addresses the relevance based ranking problem w.r.t. the search intention. Regarding to Issue 1 and Issue 2, XML keyword queries usually have ambiguities in interpreting the search for node(s) and search via node(s), due to two reasons below.

- **Ambiguity 1**: A keyword can appear both as an XML tag name and as a text value of some other nodes.
- **Ambiguity 2**: A keyword can appear as the text values of different types of XML nodes and carry different meanings.

For example see the XML document in Figure 1, keywords *customer* and *interest* appear as both an XML tag name and a text value (e.g. value of the title for book B1), and *art* appears as a text value of interest, address and name node.

Regarding to Issue 3, the search intention for a keyword query is not easy to determine and can be ambiguous, because the search via condition is not unique; so how to measure the confidence of each search intention candidate, and rank the individual matches of all these candidates are challenging.

Although many research efforts have been conducted in XML keyword search [1], [2], [3], [4], [5], none of them has addressed and resolved the above three issues yet. For instance, one widely adopted approach so far is to find the smallest lowest common ancestor (SLCA) of all keywords [3]. Each SLCA result of a keyword query contains all query keywords but has no subtree which also contains all the keywords. Since [4], [5] etc. are variations of SLCA, we use SLCA as a typical existing approach in the rest discussion. Those SLCA-based approaches only take the tree structure of XML data into consideration, without considering the semantics of the query and XML data.

In particular, regarding to Issue 1 and 2, SLCA may introduce answers that are either irrelevant to user search intention, or answers that may not be meaningful or informative enough. E.g. when a query "Jim Gray" that intends to find Jim Gray's publications on DBLP [6] is issued, SLCA returns only the *author* elements containing both keywords. Besides, SLCA also returns publications written by two authors where "Jim" is a term in 1st author's name and "Gray" is a term in 2nd author, and publications with *title* containing both keywords. It is reasonable to return such results because search intention may not be unique; however they should be given a lower rank, as they are not matches of the major search intention. Regarding to Issue 3, no existing approach has studied the problem of relevance oriented result ranking in depth yet. Moreover, they don't perform well on pure keyword query

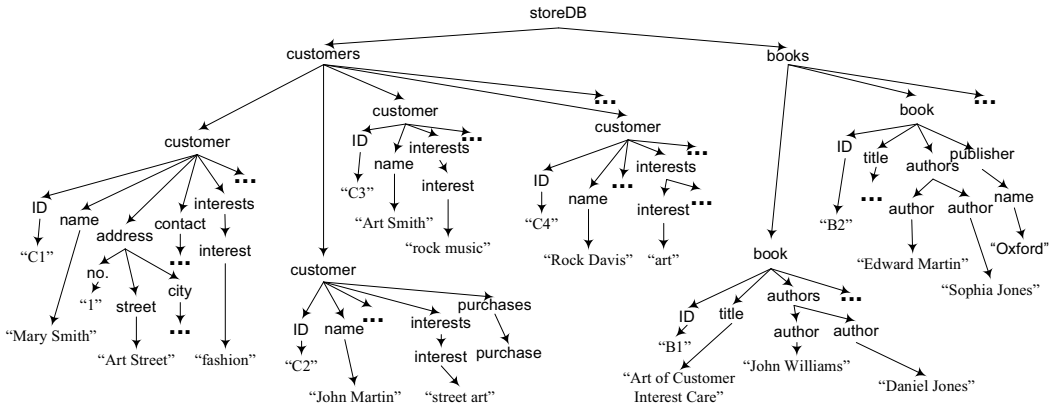


Fig. 1. Portion of data tree for an online bookstore XML database

when the schema information of XML data is not available [4]. The actual reason is, none of them can solve the keyword ambiguity problems, i.e. Ambiguity 1 and Ambiguity 2, as demonstrated by the following example.

*Example 1:* Consider a keyword query "*customer interest art*" issued on the bookstore data in Figure 1, and most likely it intends to find the customers who are interested in art.

If adopting SLCA, we will get 5 results, which include the title of book B1 and the customer nodes with IDs from C1 to C4 (as these four customer nodes contain "customer", "interest" and "art" in either the tag names or node values) in Figure 1. Since SLCA cannot well address the search intention, all these 5 SLCA results are returned without any ranking applied. However, only C4 is desired which should be put as the top ranked one, and C2 is less relevant, as his interest is "street art" rather than "art", while C1 and C3 are irrelevant.□

Inspired by the great success of IR approach on web search (especially its distinguished ranking functionality), we aim to achieve similar success on XML keyword search, to solve the above three issues without using any schema knowledge. The main challenge we are going to solve is how to extend the keyword search techniques in text databases (IR) to XML databases, because the two types of databases are different. *First*, the basic data units in text databases searched by users are flat documents. For a given query, IR systems compute a numeric score for each document and rank the document by this score. In XML databases, however, information is stored in hierarchical tree structures. The logical unit of answers needed by users is not limited to individual leaf nodes containing keywords, but a subtree instead. *Second*, unlike text database, it is difficult to identify the (major) user search intention in XML data, especially when the keywords contain ambiguities mentioned before. *Third*, effective ranking is a key factor for the success of keyword search. There may be dozens of candidate answers for an ordinary keyword query in a medium-sized database. E.g. in Example 1, five subtrees can be the query answers, but they are not equally useful to user. Due to the difference in basic answer unit between document search and database search, in XML database we need to assign a single ranking score for each subtree of certain category with a fitting size, in order to rank the answers effectively.

Statistics is a mathematical science pertaining to the collection, analysis, interpretation or explanation of data; it can be used to objectively *model a pattern* or *draw inferences* about the underlying data being studied. Although keyword search is a subjective problem that different people may have different interpretations on the same keyword query, statistics provides an objective way to distinguish the major search intention(s).

This motivates us to design a best efforts heuristic approach that provides an objective way to measure the query result relevance; thus we model the search engine as a domain expert who automatically interprets user's all possible search intention(s) through analyzing the statistics knowledge of underlying data. In this paper we propose a novel IR-style approach which well captures XML's hierarchical structure, and works well on pure keyword query independent of any schema information of XML data. In particular, the original TF*IDF similarity [7] is extended to handle both semi-structured and unstructured data, and a keyword search system prototype called XReal is implemented to achieve effective identification of user search intention and relevance oriented ranking for the search results in the presence of keyword ambiguities.

*Example 2:* We use the query in Example 1 again to explain how XReal infers user's desired result and puts it as a top-ranked answer. XReal interprets that user desires to search for customer nodes, because all three keywords have high frequency of occurrences in customer nodes. Similarly, since keywords "interest" and "art" have high frequency of occurrences in subtrees rooted at interest nodes, it is considered with high confidence that this query wants to search via interest nodes, and incorporate this confidence into our ranking formula. Besides, customers interested in "art" should be ranked before those interested in (say) "street art". As a result, C4 is ranked before C2, and further before customers with address in "art street"(e.g. C1) or named "art" (e.g. C3).□

To our best knowledge, we are the first that exploit the statistics of underlying XML database to address search intention identification, result retrieval and relevance oriented ranking as a single problem for XML keyword search. Our

main contributions are summarized as follows:

1) This is the first work that addresses the keyword ambiguity problem. We also identify three crucial issues that an effective XML keyword search engine should meet.
2) We define our own XML TF (term frequency) and XML DF (document frequency), which are cornerstones of all formulae proposed later.
3) We propose three important guidelines in identifying the user desired *search for* node type, and design a formula to compute the confidence level of a certain node type to be a desired *search for* node based on the guidelines.
4) We design formulae to compute the confidence of each candidate node type as the desired *search via* node to model natural human intuitions, in which we take into account the pattern of keywords co-occurrence in query.
5) We propose a novel relevance oriented ranking scheme called *XML TF*IDF similarity* which can capture the hierarchical structure of XML, and resolve Ambiguity 1 and Ambiguity 2 in a heuristic way; and also distinguish the similarity computation for leaf nodes and internal nodes in XML data. Moreover, our approach is able to handle both semi-structured and unstructured data.
6) We implement the proposed techniques in a keyword search engine prototype called XReal. Extensive experiments show its effectiveness, efficiency and scalability.

The rest of the paper is organized as follows. We present the related work in Section II, and preliminary on IR and data model in Section III. Section IV infers user search intention, and Section V discusses relevance oriented ranking. Section VI presents the search algorithms. Experimental evaluation is given in Section VII and we conclude in Section VIII.

## II. Related Work

Extensive research efforts have been conducted in XML keyword search to find the smallest sub-structures in XML data that each contains all query keywords in either the tree data model or the directed graph (i.e. digraph) data model.

In tree data model, LCA (lowest common ancestor) semantics is first proposed and studied in [8], [2] to find XML nodes, each of which contains all query keywords within its subtree. Subsequently, SLCA (smallest LCA [9], [3]) is proposed to find the smallest LCAs that do not contain other LCAs in their subtrees. GDMCT (minimum connecting trees) [5] excludes the subtrees rooted at the LCAs that do not contain query keywords. Sun et al. [10] generalize SLCA to support keyword search involving combinations of AND and OR boolean operators. XSeek [4] generates the return nodes which can be explicitly inferred by keyword match pattern and the concept of entities in XML data. However, it addresses neither the ranking problem nor the keyword ambiguity problem. Besides, it relies on the concept of entity (i.e. object class) and considers a node type $t$ in DTD as an entity if $t$ is "*"-annotated in DTD. As a result, customer, phone, interest, book in Figure 1, are identified as object classes by XSeek. However, it causes the multi-valued attribute to be mistakenly identified as an entity, causing the inferred return node not as intuitive as possible. E.g. phone and interest are not intuitive as entities. In fact, the identification of entity is highly dependent on the semantics of the underlying database rather than its DTD, so it usually requires the verification and decision from database administrator. Therefore, the adoption of entities for keyword search should be optional although this concept is very useful.

In digraph data model, previous approaches are heuristics-based, as the reduced tree problem on graph is as hard as NP-complete. Li et al. [11] show the reduction from minimal reduced tree problem to the NP-complete Group Steiner Tree problem on graphs. BANKS [12] uses bidirectional expansion heuristic algorithms to search as small portion of graph as possible. BLINKS [13] proposes a bi-level index to prune and accelerate searching for top-k results in digraphs. Cohen et al. [14] study the computation complexity of interconnection semantics. XKeyword [15] provides keyword proximity search that conforms to an XML schema; however, it needs to compute candidate networks and thus is constrained by schemas.

On the issue of result ranking, XRANK [2] extends Google's PageRank to XML element level, to rank among the LCA results; but no empirical study is done to show the effectiveness of its ranking function. XSEarch [1] adopts a variant of LCA, and combines a simple tf*idf IR ranking with size of the tree and the node relationship to rank results; but it requires users to know the XML schema information, causing limited query flexibility. EASE [16] combines IR ranking and structural compactness based DB ranking to fulfill keyword search on heterogenous data. Regarding to ranking methods, TF*IDF similarity [7] which is originally designed for flat document retrieval is insufficient for XML keyword search due to XML's hierarchical structure and the presence of Ambiguity 1 and Ambiguity 2. Several proposals for XML information retrieval suggest to extend the existing XML query languages [17], [18], [19] or use XML fragments [20] to explicitly specify the search intention for result retrieval and ranking.

## III. Preliminaries

### A. TF*IDF cosine similarity

TF*IDF (Term Frequency * Inverse Document Frequency) similarity is one of the most widely used approaches to measure the relevance of keywords and document in keyword search over flat documents. We first review its basic idea, then address its limitations for keyword search in XML. The main idea of TF*IDF is summarized in the following three rules.

- **Rule 1**: A keyword appearing in many documents should not be regarded as being more important than a keyword appearing in a few.
- **Rule 2**: A document with more occurrences of a query keyword should not be regarded as being less important for that keyword than a document that has less.
- **Rule 3**: A normalization factor is needed to balance between long and short documents, as Rule 2 discriminates against short documents which may have less chance to contain more occurrences of keywords.

To combine the intuitions in the above three rules, the TF*IDF similarity is designed:

$$\rho(q,d) = \frac{\sum_{k \in q \cap d} W_{q,k} * W_{d,k}}{W_q * W_d} \quad (1)$$

where $q$ represents a query, $d$ represents a flat document and k is a keyword appearing in both $q$ and $d$. A larger value of $\rho(q,d)$ indicates $q$ and $d$ are more relevant to each other. $W_{q,k}$ and $W_{d,k}$ represent the weights of $k$ in query $q$ and document $d$ respectively; while $W_q$ and $W_d$ are the weights of query $q$ and document $d$. Among several ways to express $W_{q,k}$, $W_{d,k}$, $W_q$ and $W_d$, the followings are the conventional formulae:

$$W_{q,k} = \ln(N/(f_k + 1)) \quad (2)$$

$$W_{d,k} = 1 + \ln(f_{d,k}) \quad (3)$$

$$W_q = \sqrt{\sum_{k \in q} W_{q,k}^2} \quad (4)$$

$$W_d = \sqrt{\sum_{k \in d} W_{d,k}^2} \quad (5)$$

where $N$ is the total number of documents, and document frequency $f_k$ in Formula 2 is the number of documents containing keyword $k$. Term frequency $f_{d,k}$ in Formula 3 is the number of occurrences of $k$ in document $d$.

$W_{q,k}$ is monotonical decreasing w.r.t. $f_k$ (*Inverse Document Frequency*) to reflect Rule 1; while $W_{d,k}$ is monotonical increasing w.r.t. $f_{d,k}$ (*Term Frequency*) to reflect Rule 2. The logarithms used in Formula 2 and 3 are designed to normalize the raw document frequency $f_k$ and raw term frequency $f_{d,k}$. Finally, $W_q$ and $W_d$ are increasing w.r.t. the size of $q$ and $d$, playing the role of normalization factors to reflect Rule 3.

However, the original TF*IDF is inadequate for XML, because it is not able to fulfill the job of search intention identification or resolve keyword ambiguities resulted from XML's hierarchical structure, as Example 3 shows.

*Example 3:* Suppose a keyword query *"art"* is issued to search for *customer*s interested in "art" in Figure 1's XML data. Ideally, the system should rank *customer*s who do have "art" in their nested *interest* nodes before those who do not have. Moreover, it is desirable to give customer (*A*) who is only interested in art a higher rank than another customer (*B*) who has many interests including art (e.g. C4 in Figure 1).

However, it causes two problems if directly adopting original TF*IDF to XML data. (1) If the structures in *customer* nodes are not considered, customer *A* may have a lower rank than *B* if *A* happens to have more keywords in its subtrees (analog to long document in IR) than *B*. (2) Even worse, suppose a customer *C* is not interested in art but has *address* in "art street". If $C$ has less number of keywords than $A$ and $B$ in XML data, then $C$ may have higher rank than $A$ and $B$.

### B. Data model

We model XML document as a rooted, labeled tree, such as the one in Figure 1. Our approach exploits the prefix-path of a node rather than its tag name for result retrieval and ranking. Note that the existing works [4], [21] rely on DTD while our approach works without any XML schema information.

*Definition 3.1:* (**Node Type**) The type of a node $n$ in an XML document is the prefix path from root to $n$. Two nodes are of same node type if they share the same prefix path.

In Definition 3.1, the reason that two nodes need to share same prefix path instead of their tag name is, there may be two or more nodes of the same tag name but of different semantics (i.e. in different contexts) in one document. E.g. In Figure 1, the *name* of publisher and the *name* of customer are of different node types, as they are in different contexts.

To facilitate our discussion later, we use the tag name instead of the prefix path of a node to denote the node type in all examples throughout this paper. Besides, we distinguish an XML node into either a value node or a structural node, to separate the content part from leaf node.

*Definition 3.2:* (**Value Node**) The text values contained in the leaf node of XML data (i.e. #PCDATA) is defined as a *value node*.

*Definition 3.3:* (**Structural Node**) An XML node labeled with a tag name is called a *structural node*. A structural node that contains other structural nodes as its children is called an *internal node*.

*Definition 3.4:* **(Single-valued Type)** A given node $t$ is of *single-valued type* if each node of type $t$ has at most one occurrence within its parent node.

*Definition 3.5:* **(Multi-valued Type)** A given node $t$ is of *multi-valued type* if each node of type $t$ has more than one occurrence within its parent node.

*Definition 3.6:* **(Grouping Type)** An internal node $t$ is defined as a *grouping type* if each node of type $t$ contains child nodes of only one multi-valued type.

XML nodes of *single-valued type* and *multi-valued type* can be easily identified when parsing the data. A node of single-valued (or multi-valued, or grouping) type is called a single-valued (or multi-valued, or grouping) node. E.g. in Figure 1, address is a *single-valued node*, while interest is a multi-valued node and interests is a grouping node for interest.

In this paper, for ease of presentation later, we assume every multi-valued node has a grouping node as its parent, as we can easily introduce a dummy grouping node in indexing without altering the data. Note a grouping node is also a single-valued node. Thus, the children of an internal node are either of same multi-valued type or of different single-valued types.

### C. XML TF & DF

Inspired by the important role of data statistics in IR ranking, we try to utilize it to resolve ambiguities for XML keyword search, as it usually provides an intuitionistic and convincing way to model and capture human intuitions.

*Example 4:* When we talk about *"art"* in the domain of database like Figure 1, we in the first place consider it as a value in interest of customer nodes or category (or title) of book nodes. However, we seldom first consider it as a value of other node types (e.g. street with value "Art Street").

The reason for this intuition is, usually there are many nodes of interest type and category type containing "art" in their text values (or subtrees) while "art" is usually infrequent in street nodes. Such intuition (based on domain knowledge) always can be captured by statistics of the underlying database.

Similarly, when we talk about *"interest"* here, we in the first place consider it as a node type instead of a value of

the title of book nodes with intuition thinking. Besides the simple reason that "interest" matches the XML tag interest, it can also be explained from statistical point of view, i.e. all interest nodes contain keyword "interest" in their subtrees.

The importance of statistics in XML keyword search is formalized as follows.

*Intuition 1:* The more XML nodes of a certain type $T$ (and their subtrees) contain a query keyword $k$ in either their text values or tag names, it is more intuitive that nodes of type $T$ are more closely related to the query w.r.t. keyword $k$.

In this paper, we maintain and exploit two **important** basic statistics terms, $f_{a,k}$ and $f_k^T$.

*Definition 3.7:* ***(XML TF)*** $f_{a,k}$: The number of occurrences of a keyword $k$ in a given *value node* $a$ in the XML database.

*Definition 3.8:* ***(XML DF)*** $f_k^T$: The number of $T$-typed nodes that contain keyword $k$ in their subtrees in the XML database.

Here, $f_{a,k}$ and $f_k^T$ are defined in an analogous way to term frequency $f_{d,k}$ (in Formula 3) and document frequency $f_k$ (in Formula 2) used in original TF*IDF similarity; except that we use $f_k^T$ to distinguish statistics for different node types, as the granularity on which to measure similarity in XML scenario is a subtree rather than a document. Therefore, $f_{a,k}$ and $f_k^T$ can be directly used to measure the similarity between a value node (with parent node of type $T$) and a query based on the intuitions of original TF*IDF. Besides, $f_k^T$ is also useful in resolving ambiguities, as Intuition 1 shows. We will discuss how these two sets of statistics are used for relevance oriented ranking for XML keyword search in presence of ambiguities.

## IV. Inferring Keyword Search Intention

In this section, we discuss how to interpret the search intentions of keyword query according to the statistics in XML database and the pattern of keyword co-occurrence in a query.

### A. Inferring the node type to search for

The desired node type to search for is the first issue that a search engine needs to address in order to retrieve the relevant answers. Given a keyword query $q$, a node type $T$ is considered as the desired node to search for only if the following three guidelines hold:

**Guideline 1**: $T$ is intuitively related to every query keyword in $q$, i.e. for each keyword $k$, there should be some (if not many) $T$-typed nodes containing $k$ in their subtrees.

**Guideline 2**: XML nodes of type $T$ should be informative enough to contain enough relevant information.

**Guideline 3**: XML nodes of type $T$ should not be overwhelming to contain too much irrelevant information.

Guideline 2 prefers an internal node type $T$ at a higher level to be the returned node, while Guideline 3 prefers that the level of $T$-typed node should not be very near to the root node. For instance let's refer to Figure 1: according to Guideline 2, leaf nodes of type interest, street etc. are usually not good candidates for desired returned nodes, as they are not informative. According to Guideline 3, nodes of type customers and books are not good candidates as well, as they are too overwhelming as a single keyword search result.

By incorporating the above guidelines, we define $C_{for}(T, q)$, which is the confidence of a node type $T$ to be the desired search for node type w.r.t. a given keyword query $q$ as follows:

$$C_{for}(T, q) = \log_e(1 + \prod_{k \in q} f_k^T) * r^{depth(T)} \quad (6)$$

where $k$ represents a keyword in query $q$; $f_k^T$ is the number of $T$-typed nodes that contain $k$ as either values or tag names in their subtrees; $r$ is some reduction factor with range (0,1] and normally chosen to be 0.8, and $depth(T)$ represents the depth of $T$-typed nodes in document.

In Formula 6, the first multiplier (i.e. $\log_e(1 + \prod_{k \in q} f_k^T)$) actually models Intuition 1 to address *Guideline 1*. Meanwhile, it effectively addresses *Guideline 3*, since the candidate overwhelming nodes (i.e. the nodes that are near the root) will be assigned a small value of $\prod_{k \in q} f_k^T$, resulting in a small confidence value. The second multiplier $r^{depth(T)}$ simply reduces the confidence of the node types that are deeply nested in the XML database to address *Guideline 2*. In addition, we use product rather than sum of $f_k^T$ (i.e. $\prod_{k \in q} f_k^T$) in the first multiplier to combine statistics of all query keywords for each node type $T$. The reason is, the search intention of each query usually has a unique desired node type to search for, so using product ensures that a node type needs to be intuitively related to all query keywords in order to have a high confidence as the desired type. Therefore, if a node type $T$ cannot contain all keywords of the query, its confidence value is set to 0.

*Example 5:* Given a query *"customer interest art"*, node type customer usually has high confidence as the desired node type to search for, because the values of three statistics $f^{\text{customer}}_{\text{"customer"}}$, $f^{\text{customer}}_{\text{"interest"}}$ and $f^{\text{customer}}_{\text{"art"}}$ (i.e. the number of subtrees rooted at customer nodes containing "customer", "interest" and "art" in either nested text values or tags respectively) are usually greater than 1. In contrast, node type customers doesn't have high confidence since $f^{\text{customers}}_{\text{"customer"}} = f^{\text{customers}}_{\text{"interest"}} = f^{\text{customers}}_{\text{"art"}} = 1$. Similarly, node type interest doesn't have high confidence since $f^{\text{interest}}_{\text{"customer"}}$ usually has small value. E.g. in Figure 1's XML data, $f^{\text{interest}}_{\text{"customer"}} = 0$.

Finally, with the confidence of each node type being the desired type, the one with the highest confidence is chosen as the desired search for node, when the highest confidence is significantly greater than the second highest. However, when several node types have comparable confidence values, either users can be offered a choice to decide the desired one, or the system will do a search for each convincing candidate node. Although not always fully automatic, our inference approach still provides a guidance for the system-user interaction for ambiguous keyword queries in absence of syntax.

### B. Inferring the node types to search via

Similar to inferring the desired search for node, *Intuition 1* is also useful to infer the node types to search via. However, unlike the search for case which requires a node type to be related to all keywords, it is enough for a node type to have high confidence as the desired search via node if it is closely related to some (not necessarily all) keywords, because a query

may intend to search via more than one node type. E.g. we can search for customer(s) named "Smith" and interested in "fashion" with query *"name smith interest fashion"*. In this case, the system should be able to infer with high confidence that name and interest are the node types to search via, even if keyword "interest" is probably not related to name nodes.

Therefore, we define $C_{via}(T, q)$, which is the confidence of a node type $T$ to be a desired type to search via as below:

$$C_{via}(T, q) = \log_e(1 + \sum_{k \in q} f_k^T) \quad (7)$$

where variables $k$, $q$ and $T$ have the same meaning as those in Formula 6. Compared to Formula 6, we use sum of $f_k^T$ instead of product, as it is sufficient for a node type to have high confidence as the search via node if it is related to some of the keywords. In addition, if all nodes of a certain type $T$ do not contain any keyword $k$ in their subtrees, $f_k^T$ is equal to 0 for each $k$ in q, resulting in a zero confidence value, which is also consistent with the semantics of SLCA. Then, the confidence of each possible node type to search via will be incorporated into *XML TF*IDF similarity* (which will be discussed in Section V-B) to provide answers of high quality.

### C. Capturing keyword co-occurrence

While statistics provide a macro way to compute the confidence of a node type to search via; it alone is not adequate to infer the likelihood of an individual value node to search via for a given keyword in the query.

*Example 6:* Consider a query "*customer name Rock interest Art*" searching for customers whose name includes "Rock" and interest includes "Art". Based on statistics, we can infer that name and interest-typed nodes have high confidence to *search via* by Formula 7, as the frequency of keywords "name" and "interest" are high in node types name and interest respectively. However, statistics is not adequate to help the system infer that the user wants "Rock" to be a value of name and "Art" to be a value of interest, which is intuitive with the help of keyword co-occurrence in the query. Therefore, purely based on statistics, it is difficult for search engine to differ customer C4 (with name "Art" and interest "Rock") from C3 (with name "Rock" and interest "Art") in Figure 1.

Motivated from the above example, the pattern of keyword co-occurrence in a query provides a micro way to measure the likelihood of an individual value node to search via, as a compliment of statistics. Therefore, for each query-matching value node in XML data, in order to capture the co-occurrence of keywords matching the node types and keywords matching the value nodes, the following distances are defined.

Given a keyword query $q$ and a certain value node $v$, if there are two keywords $k_t$ and $k$ in $q$, such that $k_t$ matches the type of an ancestor node of $v$ and $k$ matches a keyword in $v$, then we define the following distances.

*Definition 4.1:* (**In-Query Distance (IQD)**) The *In-Query Distance* $Dist_q(q, v, k_t, k)$ between keyword $k$ and node type $k_t$ in query $q$ with respect to a value node $v$ is defined as the position distance between $k_t$ and $k$ in $q$ if $k_t$ appears before $k$ in $q$; Otherwise, $Dist_q(q, v, k_t, k) = \infty$.

Note the position distance of two keywords $k_1$ and $k_2$ in a query $q$ is the difference of $k_1$'s position and $k_2$'s position in the query. The above definition assumes there is no repeated $k_t$ and $k$ in a query $q$. When there are multiple occurrences of $k_t$ and/or $k$ (e.g. *query "name smith address smith street"*), we define $Dist_q(q, v, k_t, k)$ as the minimal value for all possible combinations of each occurrence of $k_t$ and $k$.

*Definition 4.2:* (**Structural Distance (SD)**) The *structural Distance* $Dist_s(q, v, k_t, k)$ between $k_t$ and $k$ w.r.t. a value node $v$ is defined as the depth distance between $v$ and the nearest $k_t$-typed ancestor node of $v$ in XML document.

*Definition 4.3:* (**Value-Type Distance (VTD)**) The *Value-Type Distance* $Dist(q, v, k_t, k)$ between $k_t$ and $k$ w.r.t. a value node $v$ is defined as

$$\max(Dist_q(q, v, k_t, k), Dist_s(q, v, k_t, k)).$$

In general, the smaller the value of $Dist(q, v, k_t, k)$ is, it is more likely that $q$ intends to search via the node $v$ with value matching keyword $k$. Therefore, we define the confidence of a value node $v$ as the node to search via w.r.t. a keyword $k$ appearing in both query $q$ and $v$ as follows.

$$C_{via}(q, v, k) = 1 + \sum_{k_t \in q \cap ancType(v)} \frac{1}{Dist(q, v, k_t, k)} \quad (8)$$

*Example 7:* Consider the query $q$ in Example 6 again with same search intention. Let $n_3$ and $i_3$ represent the value nodes under name (i.e. Art Smith) and interest (i.e. rock music) respectively of customer C3. Similarly, let $n_4$ and $i_4$ be the values nodes under name and interest of customer C4. Now $Dist_q(q, n_3, \text{name}, Art) = 3$; $Dist_s(q, n_3, \text{name}, Art) = 1$; as a result $Dist(q, n_3, \text{name}, Art) = 3$ and $C_{via}(q, n_3, Art)$ = 4/3. Similarly, $C_{via}(q, i_3, Rock)$ = 1; $C_{via}(q, n_4, Rock)$ = 2; and $C_{via}(q, i_4, Art)$ = 2. We can see that the values of customer C4 are larger than those of customerC3.

## V. Relevance Oriented Ranking

In this section, we first summarize some unique features of keyword search in XML, and address the limitations of traditional TF*IDF similarity for XML. Then we propose a novel XML TF*IDF similarity which incorporates the confidence formulae we have designed in Section IV, to resolve the keyword ambiguity problem in relevance oriented ranking.

### A. Principles of keyword search in XML

Compared with flat documents, keyword search in XML has its own features. In order for an IR-style ranking approach to smoothly apply to it, we present three principles that the search engine should adopt.

**Principle 1**: When searching for XML nodes of desired type $D$ via a *single-valued node type* $V$, ideally, only the values and structures nested in $V$-typed nodes can affect the relevance of $D$-typed nodes as answers, whereas the existence of other typed nodes nested in $D$-typed nodes should not. In other words, the size of the subtree rooted at a $D$-typed node $d$ (except the subtree rooted at the search via node) shouldn't affect $d$'s relevance to the query.

*Example 8:* When searching for customer nodes via street nodes using a keyword query "*Art Street*", a customer node

(e.g. customer $C1$ in Figure 1) with the matching keyword *"street"* shouldn't be ranked lower than another customer node (e.g. customer $C3$ in Figure 1) without the matching keyword *"street"*, regardless of the sizes, values and structures of other nodes nested in $C1$ and $C3$. Note this is different from the original TF*IDF similarity that has strong intuition to normalize the relevance score of each document with respect to its size (i.e. to normalize against long documents).

**Principle 2**: When searching for the desired node type $D$ via a *multi-valued node type* $V'$, if there are many $V'$-typed nodes nested in one node $d$ of type $D$, then the existence of one query-relevant node of type $V'$ is usually enough to indicate, $d$ is more relevant to the query than another node $d'$ also of type $D$ but with no nested $V'$-typed nodes containing the keyword(s). In other words, the relevance of a $D$-typed node which contains a query relevant $V'$-typed node should not be affected (or normalized) too much by other query-irrelevant $V'$-typed nodes.

*Example 9:* Consider when searching for customers interested in art using the query "*art*", a customer with "art"-interest along with many other interests (e.g. $C4$ in Figure 1) should not be regarded as less relevant to the query than another customer who doesn't have "art"-interest but has "art street" in address (e.g. $C1$ in Figure 1).

**Principle 3**: The `order` of keywords in a query is usually important to indicate the search intention.

The first two principles look trivial if we know exactly the search via node. However, when the system doesn't have exact information of which node type to search via (as user issues pure keyword query in most cases), they are important in designing the formula of XML TF*IDF similarity; we will utilize them in designing Formula for $W_a^q$ in section V-B.2.

### *B. XML TF*IDF similarity*

We propose a recursive Formula 9, which captures XML's hierarchical structure, to compute XML TF*IDF similarity between an XML node of the desired type to search for and a keyword query. It first (*base case*) computes the similarities between leaf nodes $l$ of XML document and the query, then (*recursive case*) it recursively computes the similarities between internal nodes $n$ and the query, based on the similarity value of each child $c$ of $n$ and the confidence of $c$ as the node type to search via, until we get the similarities of search for nodes:

$$\rho_s(q,a) = \begin{cases} \frac{\sum_{k \in q \cap a} W_{q,k}^{T_a} * W_{a,k}}{W_q^{T_a} * W_a} & \text{(a) } a \text{ is value node (base case)} \\ \frac{\sum_{c \in chd(a)} \rho_s(q,c) * C_{via}(T_c,q)}{W_a^q} & \text{(b) } a \text{ is internal node (recursive case)} \end{cases} \tag{9}$$

where $q$ represents a keyword query; $a$ represents an XML node; and the result $\rho_s(q,a)$ represents the similarity value between $q$ and $a$.

We first discuss the intuitions behind Formula 9 briefly.
(1) In the base case, we compute the similarity values between XML leaf nodes and a given query in a similar way to original TF*IDF, since leaf nodes contain only keywords with no further structure.
(2) In the recursive case: on one hand, if an internal node $a$ has more query relevant child nodes while another internal node $a'$ has less, then it is likely that $a$ is more relevant to the query than $a'$. This intuition is reflected as the numerator in Formula 9(b). On the other hand, we should take into account the fan-out (size) of the internal node as normalization factor, since the node with large fan-out has a higher chance to contain more query relevant children. This is reflected as the denominator of Formula 9(b).

Next, we will illustrate how each factor in Formula 9 contributes to the XML structural similarity in Section V-B.1 (for base case) and V-B.2 (for recursive case).

*1) Base case of XML TF*IDF:* Since XML leaf nodes contain keywords with no further structure, we can adopt the intuitions of original TF*IDF to compute the similarity between a leaf node and a keyword query by using statistics terms $f_k^T$ and $f_{a,k}$ which have been explained in Section III-C.

However, unlike Rule 1 in original TF*IDF which models and assigns the same weight to a query keyword w.r.t. all documents (i.e. $W_{q,k}$ in Formula 2), we model and distinguish the weights of a keyword w.r.t. different XML leaf node types (i.e. $W_{q,k}^{T_a}$ in Formula 10).

*Example 10:* Keyword *"road"* may appear quite frequently in street nodes of Figure 1 while infrequently in other nodes. Thus it is necessary to distinguish the (low) weight of "road" in address from its (high) weight in other nodes. Similarly, we distinguish the weights of a query w.r.t. different XML node types (i.e. $W_q^{T_a}$), rather than fixed weight for a given query for all flat documents.

Now let's take a detailed look at Formula 9. In the base case for XML leaf nodes, each $k$ represents a keyword appearing in both query $q$ and value node $a$; $T_a$ is the type of $a$'s parent node; $W_{q,k}^{T_a}$ represents the weight of keyword $k$ in $q$ w.r.t. node type $T_a$. $W_{a,k}$ represents the weight of $k$ in leaf node $a$; $W_q^{T_a}$ represents the weight of $q$ w.r.t. node type $T_a$; and $W_a$ represents the weight of $a$. Following the conventions of original TF*IDF, we propose the formulas for $W_{q,k}^{T_a}$, $W_{a,k}$, $W_q^{T_a}$ and $W_a$ in Formula 10, 11, 12 and 13 respectively:

$$W_{q,k}^{T_a} = C_{via}(q,a,k) * \log_e\left(1 + N_{T_a}/(1 + f_k^{T_a})\right) \tag{10}$$

$$W_{a,k} = 1 + \log_e\left(f_{a,k}\right) \tag{11}$$

$$W_q^{T_a} = \sqrt{\sum_{k \in q} (W_{q,k}^{T_a})^2} \tag{12}$$

$$W_a = \sqrt{\sum_{k \in a} W_{a,k}^2} \tag{13}$$

In Formula 10, $N_{T_a}$ is the total number of nodes of type $T_a$ while $f_k^{T_a}$ is the number of $T_a$-typed nodes containing keyword $k$; $C_{via}(q,a,k)$ is the confidence of node $a$ to be a search via node w.r.t. keyword $k$ (explained in Section IV-C). In Formula 11, $f_{a,k}$ is the number of occurrences of $k$ in value node $a$. Similar to Rule 1 and Rule 2 in original TF*IDF, $W_{q,k}^{T_a}$ is monotonical decreasing w.r.t. $f_k^{T_a}$, while $W_{a,k}$ is

monotonical increasing w.r.t. $f_{a,k}$. $W_a$ is normally increasing w.r.t. the size of $a$, so put it as part of denominator to play a role of normalization factor to balance between leaf nodes containing many keywords and those with a few keywords.

*2) Recursive case of XML TF*IDF:* The recursive case of Formula 9 recursively computes the similarity value between an internal node $a$ and a keyword query $q$ in a bottom-up way based on two intuitions below.

*Intuition 2:* An internal node $a$ is relevant to $q$, if $a$ has a child $c$ such that the type of $c$ has high confidence to be a *search via* node w.r.t. $q$ (i.e. large $C_{via}(T_c, q)$), and $c$ is highly relevant to $q$ (i.e. large $\rho_s(q, c)$).

*Intuition 3:* An internal node $a$ is more relevant to $q$ if $a$ has more query-relevant children when all others being equal.

In the recursive case of Formula 9, $c$ represents one child node of $a$; $T_c$ is the node type of $c$; $C_{via}(T_c, q)$ is the confidence of $T_c$ to be a search via node type presented in Formula 7; $\rho_s(q, c)$ represents the similarity between node $c$ and query $q$ which is computed recursively; $W_a^q$ is the overall weight of $a$ for the given query $q$.

Next, we explain the similarity design of an internal node $a$ in Formula 9: we first get a weighted sum of the similarity values of all its children, where the weight of each child $c$ is the confidence of $c$ to be a *search via* node w.r.t. query q. This weighted sum is exactly the numerator of formula 9, which also follows *Intuition 2* and *3* mentioned above. Besides, since *Intuition 3* usually favors internal nodes with more children, we need to normalize the relevance of $a$ to $q$. That naturally leads to the use of $W_a^q$ (Formula 14) as the denominator.

*3) Normalization factor design:*

$$W_a^q = \begin{cases} \sqrt{\sum_{c \in chd(a)} (C_{via}(T_c, q) * B + DW(c))^2} & \text{(a) if } a \text{ is grouping node} \\ \sqrt{\sum_{T \in chdType(T_a)} C_{via}(T, q)^2} & \text{(b) otherwise} \end{cases} \quad (14)$$

Formula 14 presents the design of $W_a^q$, which is used as a normalization factor in the recursive case of XML TF*IDF similarity formula. $W_a^q$ is designed based on *Principle 1* and *Principle 2* pointed out in section V-A.

Formula 14(a) presents the case that internal node $a$ is a *grouping node*; then for each child $c$ of $a$ (i.e. $c \in chd(a)$), $B$ is considered as a Boolean flag: $B = 1$ if $\rho_s(q, c) > 0$ and $B = 0$ otherwise; $DW(c)$ is a small value as the default weight of $c$ which we choose $DW(c) = 1/\log_e(e - 1 + |chd(a)|)$ if $B = 0$ and $DW(c) = 0$ if $B = 1$, where $|chd(a)|$ is the number of children of $a$, so that $W_a^q$ for grouping node $a$ grows with the number of query-irrelevant child nodes, but grows very slowly to reflect *Principle 2*. Note $DW(c)$ is usually insignificant as compared to $C_{via}(T_c, q)$.

Now let's explain the reason that we design Formula 14(a).

The intuition for the formula of *grouping node* $a$ comes from *Principle 2*, so we don't count $C_{via}(T_c, q)$ in the normalization unless $c$ contains some query keywords within its subtree. In this way, the similarity of $a$ to $q$ will not be significantly normalized (or affected) even if $a$ has many query-irrelevant child nodes of the same type. At the same time, with the default weight $DW(c)$, we still provide a way to distinguish and favor a grouping node with small number of children from another grouping node with many children, in case that the two contain the same set of query-relevant child nodes. Informally speaking, the more compact the meaningful answer is, the higher the rank it is given.

When internal node $a$ is a *non-grouping node*, we compute $W_a^q$ based on the type of $a$ rather than each individual node. In Formula 14(b), $chdType(T_a)$ represents the node types of the children of $a$, and it computes the same $W_a^q$ for all $a$-typed nodes even if each individual $a$-typed node may have different set of child nodes (e.g. some customer nodes have nested address while some do not have).

This design has two advantages. *First*, it models *Principle 1* to achieve a normalization that the size of the subtree of individual node $a$ does not affect the similarity of $a$ to a query.

*Example 11:* Given a query $q$ *"customer Art Street"*, since address has high confidence to be searched via (i.e. $C_{via}(address, q)$), $C1$ (with address in "Art Street") will be ranked before $C2$ (with interest in "street art") according to the normalization in Formula 14(b). However, if we compute the normalization factor based on the size of each individual node, then the high confidence for address node doesn't contribute to the normalization factor of $C2$ (who even doesn't have address and street nodes etc.). As a result, $C2$ has a good chance to be ranked before $C1$ due to its small size which results in small normalization factor.

*Second*, Formula 14(b)'s design has advantage in term of computation cost. With $W_a^q$ for non-grouping node computed based on node types instead of data nodes, we only need to compute $W_a^q$ for all $a$-typed nodes once for each query, instead of repeatedly computing $W_a^q$ for each $a$-typed node in the data.

Note in the base case, a keyword $k$ is less important in $T$-typed nodes if more $T$-typed nodes contain $k$. However, now we consider $T$-typed nodes are more important for keyword $k$ (i.e. larger $C_{via}(T, k)$). These two, which seem contradictive, are in fact the key to accurate relevance based ranking.

*Example 12:* Consider when searching for customers with query "*customer art road*", statistics will normally give more weights to address than other node types because of the high frequency of keyword "road" in address. But if no customer node has address in "art road" but some have address in "art street", then these customer nodes will be ranked before customers with address containing "road" without "art". Because the keyword "road" has a lower weight than "art" in address nodes due to its much higher frequency.

## VI. Algorithms

### A. Data processing and index construction

We parse the input XML document during which we collect the following information for each node $n$ visited: (1) assign a Dewey label $DeweyID$ [22] to $n$; (2) store the prefix path $prefixPath$ of $n$ as its node type in a global hash table, so that any two nodes sharing the same $prefixPath$ have the

same node type; (3) in case $n$ is a leaf node, we create a value node $a$ (mentioned in section III-B) as its child and summarize two basic statistics data $f_{a,k}$ (in Definition 3.7) and $W_a$ (in Formula 13) at the same time. Besides, we also build two indices in order to speedup the keyword query processing.

The first index built is called keyword inverted list, which retrieves a list of value nodes in document order whose values contain the input keyword. In particular, we have designed and evaluated three candidates for the inverted list: (1) Dup, the most basic index which stores only the dewey id and XML TF $f_{a,k}$; (2) DupType, which stores an extra node type (i.e. its prefix path) compared to Dup; (3) DupTypeNorm, which stores an extra normalization factor $W_a$ (in Formula 13) associated with this value node compared to DupType. DupTypeNorm provides the most efficient computation of XML TF*IDF, as it costs the least index lookup time; in contrast Dup and DupType need extra index lookup to gather the value of $W_{a,k}$ (see formula 11) to compute $W_a$ online.

Given a keyword $k$, the inverted list returns a set of nodes $a$ in document order, each of which contains the input keyword and is in form of a tuple $<DeweyID,\ prefixPath,\ f_{a,k},\ W_a>$. Each term here has been explained as above. In order to facilitate the explanations of the algorithm, we name such tuple as "*Node*". It supports the following operations:

- getDeweyID(a,k) returns the Dewey id of value node $a$.
- getPrefix(a,k) returns the prefix path of $a$ in XML data.
- getFrequency(a,k) returns the value of $f_{a,k}$.

**Algorithm 1**: KWSearch($keywords[m]$, $IL[m]$, $F[m]$)

```
1  Let max = 0; T_for = null
2  List L_for = getAllNodeTypes()
3  foreach T_n∈L_for do
4      C_for(T_n, keywords) = getSearchForConfidence(T_n,keywords)
5      if (C_for(T_n) > max) then
6          max = C_for(T_n); T_for = T_n
7  LinkedList rankedList
8  N_for = getNext(T_for)
9  while (!end(IL[1]) || ... || (!end(IL[m])) do
10     Node a = getMin(IL[1],IL[2],...,IL[m])
11     if (!isAncestor(N_for, a)) then
12         ρ_s(keywords,N_for) = getSimilarity(N_for,keywords)
13         rankedList.insert(N_for, ρ_s(keywords,N_for))
14         N_for = getNext(T_for)
15     if (isAncestor(N_for, a)) then
16         ρ_s(keywords, a) = getSimilarity(a,keywords)
17     else
18         ρ_s(keywords, a) = 0
19 return rankedList;
```

The second index built is called frequency table, which stores the frequency $f_k^T$ for each combination of keyword $k$ and node type $T$ in XML document. Its worst case space complexity is O(K*T), where K is the number of distinct keywords and T is the number of node types in XML database. Since the number of node types in a well designed XML database is usually small (e.g. 100+ in DBLP 370MB and 500+ in XMark 115MB), the frequency table size is comparable to inverted list. It is indexed by keywords using Berkeley DB B+-tree [23], so the index lookup cost is O(log(K)). It supports getFrequency(T,k) which returns the value of $f_k^T$.

Note that values returned by these operations are important to compute the result of the formulae presented in Section V.

### B. Keyword search & ranking

Algorithm 1 presents a flowchart of keyword search and result ranking. The input parameters $keywords[m]$ is a keyword query containing $m$ keywords. Based on the inverted lists built after pre-processing the XML document, we extract the corresponding lists $IL[1]$, ..., $IL[m]$ for each keyword in the query. $F$ is the frequency table mentioned in section VI-A. In particular, Algorithm 1 executes in three steps.

First, it identifies the search intention of the user, i.e. to identify the most desired search for node type (line 1-6). In particular, it first collects all distinct node types in XML document (line 2). Then for each node type, we compute its confidence to be a search for node through Formula 6, and choose the one with the maximum confidence as the desired search for node type $T_{for}$ (line 3-6).

Second, for each search for node candidate $N_{for}$, it computes the XML TF*IDF similarity between $n$ and the given keyword query (line 7-18). We maintain a $rankedList$ to contain the similarity of each search for node candidate (line 7). $N_{for}$ is initially set to the first node of type $T_{for}$ in document order (line 8). The computation of XML TF*IDF similarity between an XML node and the given query is computed recursively in a bottom-up way (line 9-18): for each $N_{for}$, we first extract node $a$ which occurs first in document order (line 10), then compute the similarity of all leaf nodes $a$ by calling Function $getSimilarity()$, then go one level up to compute the similarity of the lowest *internal node* (line 15-18), until it reaches up to $N_{for}$, which is actually the root of all nodes computed before. Then it computes the similarity between current $N_{for}$ and the query (line 12), insert a pair ($N_{for}$, $\rho$) into $rankedList$ (line 13), and move the cursor to next $N_{for}$ by calling function $getNext()$ and calculate the similarity of next $N_{for}$ in the same way (line 14).

Third, it returns the ranked list of all search for node candidates by their similarity to the query (line 19).

**Function** `getSimilarity`(*Node a, q[n]*)

```
1  if (isLeafNode(a)) then
2      foreach k ∈ q ⋂ a do
3          C_via(q, a, k) = getKWCo-occur(q,a,k);
4          W_{q,k}^{T_a} = getQueryWeight(q,k,a);
5          W_{q,k}^{T_a} = C_via(q, a, k) * W_{q,k}^{T_a};
6          W_{a,k} = 1+log_e(f_{a,k});
7          sum += W_{q,k}^{T_a} * W_{a,k};
8      ρ_s(q, a) = sum/(W_q^{T_a} *getWeight(a));
9  if (isInternalNode(a)) then
10     W_a^q = getQWeight(a,q);
11     foreach c∈child(a) do
12         T_c = getNodeType(c);
13         C_via(T_c,q) = getSearchViaConfidence();
14         sum += getSimilarity(c, q) * C_via(T_c,q);
15     ρ_s(q, a) = sum/W_a^q;
16 return ρ_s(q, a);
```

Function $getSimilarity()$ presents the procedure of computing XML TF*IDF similarity between a document node $a$ and a given query $q$ of size $n$. There are two cases to consider. `Case 1`: $a$ is a leaf node (line 1-8). For each keyword $k$ in both $a$ and $q$, we first capture whether $k$ co-occurs with keyword $k_t$ matching some node type. Line 3-8 present the

calculation details of $\rho_s(q, a)$ in Formula 9(a). The statistics in line 3,5,6 are illustrated in Formula 8, 10 and 11 respectively. `Case 2`: $a$ is an internal node (line 9-15). We compute $a$'s similarity $\rho_s(q, a)$ w.r.t. query $q$ by exactly following Formula 9(b). $\rho_s(q, a)$ is computed by a sum of the product of the similarity of each of its child $c$ and the confidence value of $c$ as a search via node (line 11-14). Finally, $\rho_s(q, a)$ is normalized by a factor $W_a^q$ (line 15), which is the weight of internal node $a$ w.r.t. $q$. Lastly, we return the similarity value (line 16).

Moreover, XReal can work on both semi-structured and unstructured data, since unstructured data is a special case of semi-structured data with no structure, and XML TF*IDF ranking formula 9(a) for value node can be easily simplified to original TF*IDF Formula 1 by ignoring the node type.

## VII. Experiments

We have performed comprehensive experiments to compare the effectiveness, efficiency and scalability of XReal with SLCA and XSeek. XReal and SLCA are implemented in Java and run on a 3.6GHz Pentium 4 machine with 1GB RAM running Windows XP; the binary file of XSeek is generously provided by its author. We have tested both synthetic and real datasets. The synthetic dataset is generated using XMark benchmark [24] with size 115MB; WSU and eBay from Washington XML Data Repository [25] and DBLP 370MB are used as real datasets. Berkeley DB Java Edition [23] is used to store the keyword inverted lists and frequency table.

| | Query | Intention | *XReal* | *SLCA* / XSeek |
|---|---|---|---|---|
| **DBLP (370MB)** | | | | |
| $QD_1$ | Java, book | book | book | book; title / book; article |
| $QD_2$ | author, Chen, Lei | inproceedings | inproceedings | author |
| $QD_3$ | Jim, Gray, article | article | article | article |
| $QD_4$ | xml, twig | inproceedings | inproceedings | title / inproceedings |
| $QD_5$ | Ling, tok, wang, twig | inproceedings | inproceedings | inproceedings |
| $QD_6$ | vldb, 2000 | inproceedings | inproceedings | inproceedings |
| **WSU (16.5MB)** | | | | |
| $QW_1$ | 230 | place | course; place | room; crs / course |
| $QW_2$ | CAC, 101 | course | course | course |
| $QW_3$ | ECON | course | course | prefix / course |
| $QW_4$ | Biology | course | course | title / course |
| $QW_5$ | place, TODD | course | course | place / course |
| $QW_6$ | days, TU, TH | course | course | days / course |
| **eBay (0.36MB)** | | | | |
| $QE_1$ | 2, days | auction_info | listing | time_left / listing |
| $QE_2$ | cpu, 933 | listing | listing | cpu / listing |
| $QE_3$ | Hard, drive, CA | listing | listing | description / listing |

Fig. 2. Test on inferring the *search for* node

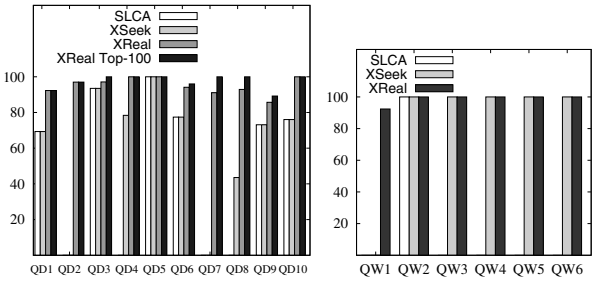
The effectiveness test contains two parts: (1) the quality of inferring the desired *search for* node; (2) the quality of our ranking approach.

### A. Search effectiveness

*1) Infer the search for node:* To test XReal's accuracy in inferring the desired search for node, we make a survey of 20 keyword queries, most of which do not contain an explicit search for node. To get a fairly objective view of user search intentions in real world, we post this survey online and ask for 46 people to write down their desired search for and search via nodes. We summarize their answers and choose the queries that more than 80 percentage of users agree on a same search intention. The final queries are shown in Figure 2, and some queries contain ambiguities: e.g. $QD_1$ and $QD_3$ have both Ambiguity 1 and Ambiguity 2; $QD_2$, $QD_6$ and $QW_1$ have Ambiguity 2. The 4th column contains the search for node inferred by XReal while the 5th column contains the majority node types returned by SLCA and XSeek, as the semantics of SLCA cannot guarantee all results are of the same node type.

We find XReal is able to infer a desired search for node in most queries, especially when the search for node is not given explicitly in the query (e.g. $QD_2$, $QD_4$, $QW_2$, $QE_1$), or its choice is not unique (e.g. $QD_1$, $QD_3$), or both cases such as $QW_1$. XSeek just blindly infers the return nodes of individual keyword matches case by case, rather than addressing the major search intention(s), whereas XReal does so before it goes to find individual matches.

In addition, if more than one candidate have comparable confidence to be a search for node, XReal returns all possible candidates (for user to decide) or returns a ranked list for each such candidate if user interaction is not preferred. E.g. in $QW_1$, both place and course can be the return node, as the frequency of "230" in subtrees of course and place are comparable. The search for node models a real world object, so we choose to return sub-trees rooted at the desired search for node, and provide links to the descendants of subtrees for user interested in particular parts of the subtree to explore.




(a) DBLP (b) WSU

Fig. 3. Precision Comparison(%)

*2) Precision, Recall & F-measure:* To measure the search quality, we evaluate all queries in Figure 2, and summarize two metrics, i.e. precision and recall borrowed from IR field. The results are shown in Figure 3 and 4. Precision measures the percentage of the output subtrees that are desired; recall measures the percentage of the desired subtrees that are output. We obtain the correct answers by running the schema-aware XQuery and additionally verifying the correctness manually.

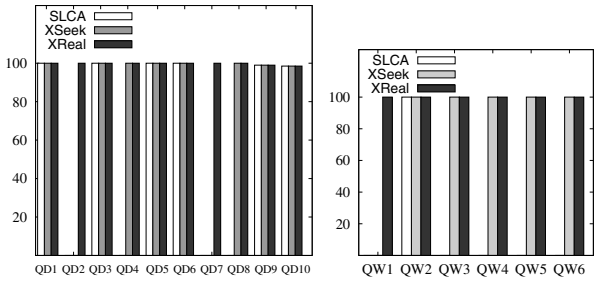
To evaluate XReal's performance on large real datasets, we include four more queries for DBLP: $QD_7$ "Philip Bernstein"; $QD_8$ "WISE"; $QD_9$ "ER 2005"; $QD_{10}$ "LATIN 2006". Each of these queries have Ambiguity 2 problem, e.g. "WISE" can be the *booktitle*, *title* of inproceedings, or a value of *author*.

As users are always interested in top-k results, so we compute XReal's *top-100* precision besides the overall precision

of SLCA, XSeek and XReal on DBLP and WSU, shown in Figure 3; results on eBay are not shown due to space limit. We have four main observations as below.

(1) XReal achieves higher precision than SLCA and XSeek for the queries that contain ambiguities (e.g. $QD_1$-$QD_4$, $QD_6$-$QD_{10}$, $QW_1$). E.g. in $QD_3$ which intends to find articles written by author "Jim Gray". Since "article" can be either a tag name or a value of title node, and "Jim" and "Gray" can appear in one author or two different authors, SLCA and XSeek generate many false positive results and lead to low accuracy, while XReal addresses these ambiguities well. As another example in $QD_9$ which intends to find the inproceedings of *ER* conference in year 2005. As "ER" appears in both *booktitle* and *title*, and "2005" appears in both *title* and *year*, XSeek returns not only the intended results, but also other inproceedings whose title contains both keywords; but XReal correctly interprets the search intention.

(2) We find SLCA suffers a zero precision and recall from the pure keyword value query, e.g. $QD_4$, $QD_7$, $QD_8$, $QW_1$ and $QW_3$ etc. Because the result returned by SLCA contains nothing relevant except the SLCA node. E.g. for $QD_8$ SLCA returns the *booktitle* or *title* nodes containing "WISE", while user wants to find the inproceedings of "WISE" conference. However, XReal detects keyword "WISE" has large occurrences as a *booktitle* of inproceedings and correctly captures the search intention. XSeek suffers a zero precision in $QD_2$ and $QD_7$, mainly because it mistakenly decides "*author*" as an entity, while the query intends to find the publications.




(a) DBLP (b) WSU

Fig. 4. Recall Comparison(%)

(3) XReal Performs as well as XSeek (in both recall and precision) when queries have no ambiguity in XML data (e.g. $QD_5$, $QW_4$-$QW_6$, etc).

(4) By comparing the overall and *top-100* precision of XReal on DBLP in Figure 3(a), *XReal Top-100* has a higher precision, which indirectly proves our ranking strategy works well w.r.t. the user search intention on large datasets.

TABLE I

F-MEASURE COMPARISON

| *F-measure* | *SLCA* | *XSeek* | *XReal* | *XReal top-100* |
|---|---|---|---|---|
| DBLP | 0.272 | 0.3461 | 0.4748 | 0.4799 |
| WSU | 0.0083 | 0.4162 | 0.4967 | 0.497 |
| EBAY | 0 | 0.4002 | 0.4002 | 0.4002 |

Furthermore, we adopt *F-measure* used in IR as the weighted harmonic mean of precision and recall. We compute the average precision and recall of all queries in Figure 2 (plus $QD_7$-$QD_{10}$) for each dataset, adopting formula $F = precision * recall/(precision + recall)$ to get F-measure in Table I. We find XReal beats SLCA and XSeek on all datasets, and achieves almost a perfect value of F which is 0.5 on WSU.

TABLE II

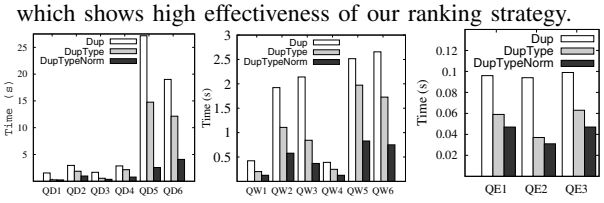
RANKING PERFORMANCE OF XREAL

| Dataset | *Top-1 Number/Total Number* | *R-Rank* | *MAP* |
|---|---|---|---|
| DBLP | 27/30 | 0.946 | 0.925 |
| WSU | 8/10 | 0.85 | 0.803 |
| eBay | 9/10 | 0.9 | 0.867 |
| XMark | 7/10 | 0.791 | 0.713 |

### B. Ranking effectiveness

To evaluate the effectiveness of XReal's ranking strategy, we use three measures widely adopted in IR field. (1) *Number of top-1 answers that are relevant*. (2) *Reciprocal rank* (R-rank). For a given query, the reciprocal rank is 1 divided by the rank at which the first correct answer is returned, or 0 if no correct answer is returned. (3) *Mean Average Precision (MAP)*. A precision is computed after each relevant answer is retrieved, and MAP is the average value of such precisions. The first two measure how good the system returns one relevant answer, while the third one measures the overall effectiveness for top-k answers returned, k=40 for DBLP (as DBLP data has very large size) and k=20 for others.

We evaluate a set of 30 randomly generated queries on DBLP, and 10 queries on WSU, eBay and XMark, with an average of 3 keywords. The average values of these metrics are recorded in Table II. We find XReal has an average R-rank greater than 0.8 and even over 0.9 on DBLP. Besides, XReal returns the relevant result in its top-1 answer in most queries, which shows high effectiveness of our ranking strategy.




(a) DBLP (b) WSU (c) eBay

Fig. 5. Response time on individual queries

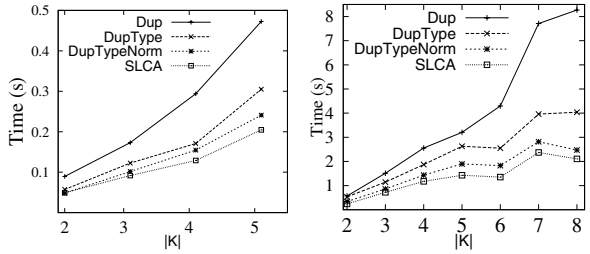
### C. Efficiency

We compare the query response time of XReal adopting three indices for keyword inverted list mentioned in section VI-A, i.e. *Dup*, *DupType* and *DupTypeNorm*, measured by the timestamp difference between a query is issued and result is returned. Throughout section VII, XReal refers to the one adopting *DupTypeNorm*. Figure 5 shows the time on hot cache for queries listed in Figure 2. *DupTypeNorm* outperforms the other two on all three real datasets, about 2 and 4 times faster than *DupType* and *Dup* respectively. Because *DupTypeNorm* stores the dewey id, node type and normalization factor (for value nodes) together, thus it needs less number of index lookups to fulfill the similarity computation in Formula 9. Such advantage is significant when the number of keywords is large or query result size is large, e.g. $QD_5$ and $QD_6$ in Figure 5(a).

### D. Scalability

Among the existing keyword search systems SLCA[3], GDMCT[5] and XSEarch[1], SLCA is recognized as the most

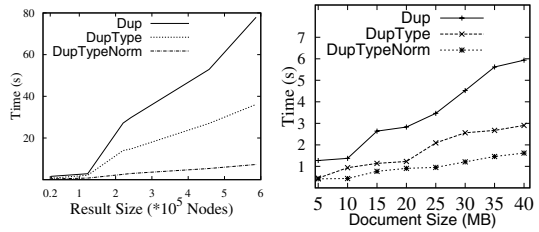
efficient one so far, so we compare XReal with SLCA on DBLP and XMark. For each dataset, we test a set of 50 randomly generated queries, each guarantees to have at least one SLCA result and contains $|K|$ number of keywords, where $|K|$ = 2 to 8 for DBLP and $|K|$ = 2 to 5 for XMark. The response time is average time of the corresponding 50 queries in four executions on hot cache, as shown in Figure 6. From Figure 6(a) and 6(b), we find XReal is nearly 20% slower than SLCA on both datasets which is acceptable, because SLCA only find all the SLCA nodes resulting in low accuracy, while XReal does extra search intention identification, precise result retrieval and ranking; and XReal finds extra results (satisfying the boolean OR semantics). So this overhead is worthwhile. We also find, the response time of each proposed index increases as number of keywords increases. In particular, the one using *DupTypeNorm* index costs less time than *DupType*, in turn less than *Dup*. XReal adopting *DupTypeNorm* index scales as well as SLCA, especially when $|K|$ varies from 5 to 8 for DBLP (Figure 6(b)).




(a) XMark (b) DBLP

Fig. 6. Response time on different number of keywords |K|

Besides, we evaluate the scalability of those indices by drawing the relationship between the response time and query result size (in term of number of nodes returned). A range of 15 queries with various result sizes run over DBLP, and the result is shown in Figure 7(a). We can see *DupTypeNorm* again outperforms the other two, and scales linearly w.r.t. the query result size. Similarly, we test the response time of a query "*location united states item*" on XMark data of size 5MB up to 40MB. As shown in Figure 7(b), both *DupTypeNorm* and *DupType*'s response time increases linearly w.r.t. the data size.




(a) DBLP (b) XMark

Fig. 7. Response time w.r.t. result/document size

## VIII. Conclusion

In this paper, we study the problem of effective XML keyword search which includes the identification of user search intention and result ranking in the presence of keyword ambiguities. As statistics provide an objective way to model patterns and draw inferences on the underlying data, we utilize them to infer user's search intention and rank the query results. In particular, we define XML TF (term frequency) and XML DF (document frequency). Based on these two statistic terms, we design formulas to compute the confidence level of each candidate node type to be a *search for/search via node*. Then, we propose a novel *XML TF*IDF similarity* ranking scheme which takes the above confidence levels and the co-occurrence of keywords into consideration, and well captures the hierarchical structure of XML document. Moreover, we are the first paper to investigate and solve the keyword ambiguity problem. As a result, we implement an XML keyword search engine prototype called XReal, which exploits only data statistics to combine search intention identification, search result retrieval and relevance oriented ranking together as a single problem in XML keyword search. Extensive experiment results show XReal is much more effective than the existing approaches. For future work, we are now investigating the ranking strategy for XML documents with ID/IDREFs.

## References


[1] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSEarch: A semantic search engine for XML," in *Proc. of VLDB Conference*, 2003, pp. 45–56.

[2] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK:ranked keyword search over XML documents," in *SIGMOD*, 2003.

[3] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest LCAs in XML databases," in *SIGMOD*, 2005, pp. 537–538.

[4] Z. Liu and Y. Chen, "Identifying meaningful return information for xml keyword search," in *SIGMOD Conference*, 2007.

[5] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava, "Keyword proximity search in XML trees," in *TKDE*, 2006, pp. 525–539.

[6] M. Ley, "Dblp computer science bibliography record," http://www.informatik.uni-trier.de/ ley/db/.

[7] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[8] A. Schmidt, M. L. Kersten, and M. Windhouwer, "Querying xml documents made easy: Nearest concept queries." in *ICDE*, 2001, pp. 321–329.

[9] Y. Li, C. Yu, and H. V. Jagadish, "Schema-free XQuery," in *VLDB2004*.

[10] C. Sun, C. Y. Chan, and A. K. Goenka, "Multiway slca-based keyword search in xml data," in *WWW*, 2007, pp. 1043–1052.

[11] W. S. Li, K. S. Candan, Q. Vu, and D. Agrawal, "Retrieving and organizing web pages by information unit," in *WWW*, 2001, pp. 230–244.

[12] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional expansion for keyword search on graph databases," in *Proc. of VLDB Conference*, 2005, pp. 505–516.

[13] H. He, H. Wang, J. Yang, and P. S. Yu, "Blinks: ranked keyword searches on graphs," in *SIGMOD Conference*, 2007, pp. 305–316.

[14] S. Cohen, Y. Kanza, B. Kimelfeld, and Y. Sagiv, "Interconnection semantics for keyword search in xml," in *CIKM*, 2005, pp. 389–396.

[15] V. Hristidis, Y. Papakonstantinou, and A. Balmin, "Keyword proximity search on XML graphs," in *ICDE*, 2003, pp. 367–378.

[16] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: Efficient and adaptive keyword search on unstructured, semi-structured and structured data," in *SIGMOD*, 2008.

[17] N. Fuhr and K. Großjohann, "Xirql: A query language for information retrieval in xml documents," in *SIGIR*, 2001, pp. 172–180.

[18] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit, "Flexpath: flexible structure and full-text querying for xml," in *SIGMOD conference*, 2004.

[19] A. Theobald and G. Weikum, "The index-based xxl search engine for querying xml data with relevance ranking," in *EDBT*, 2002, pp. 477–495.

[20] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer, "Search xml documents via xml fragments," in *SIGIR*, 2003, pp. 151–158.

[21] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective keyword search for valuable lcas over xml documents," in *CIKM*, 2007, pp. 31–40.

[22] V. Vesper., "Let's do dewey," http://www.mtsu.edu/ vvesper/dewey.html.

[23] "Berkeley DB," http://www.sleepycat.com/.

[24] "http://www.xml-benchmark.org/."

[25] "http://www.cs.washington.edu/research/xmldatasets."

# 高效的XML关键字查询改写和结果生成技术

黄静　陆嘉恒　孟小峰

(中国人民大学信息学院 北京 100872)

(huangjingruc@ruc.edu.cn)

# Efficient XML Keyword Query Refinement with Meaningful Results Generation

Huang Jing,　Lu Jiaheng,　and Meng Xiaofeng

(*School of Information, Renmin University of China, Beijing* 100872)

**Abstract** Keyword search method provides users with a friendly way to query XML data, but a user's keyword query may often be an imperfect description of their intention. Even when the information need is well described, a search engine may not be able to return the results matching the query as stated. The task of refining the user's original query is first defined to achieve better result quality as the problem of keyword query refinement in XML keyword search, and guidelines are designed to decide whether query refinement is necessary. Four refinement operations are defined, namely term deletion, merging, split and substitution. Since there may be more then one query refinement candidates, proposes the definition of refinement cost, which is used as a measure of semantic distance between the original query and refined query, and also a dynamic programming solution to compute refinement cost. In order to achieve the goal of finding the best refined queries and generate their associated results within a one-time node list scan, a stack-based algorithm is proposed, followed by a generalized partition-based optimization, which improves the efficiency a lot. Finally, extensive experiments have been done to show efficiency and effectiveness of the query refinement approach.

**Keywords** XML; Keyword Search; Query Refinement;Query Rewriting; Query Suggestion; SLCA

**摘要**　用户使用关键字查询时,可能不能准确的表达他们的意图,即使用户正确的表达了查询意图,查询引擎也可能不能准确地返回查询结果.针对这一问题,重点研究了在 XML 关键字查询中如何进行有效的查询改写并生成有意义的结果.提出四种查询改写操作和查询改写代价的概念,给出了动态规划的方法计算查询改写代价.为了找出最优的查询改写,给出了基于栈的查询改写和结果生成算法,并提出了基于划分的优化算法.最后通过丰富的实验对提出的方法进行了验证.

**关键词** XML; 关键字查询; 查询改写; 查询重写; 查询推荐; SLCA

中图法分类号　TP391

## 0 引言

关键字查询为用户提供了友好便捷的查询方式,如何使用关键字查询从XML数据中获取所需信息已经成为学术界近期研究的一个热点问题[1-5].这些工作主要研究如何过滤无关的查询结果来提高查准率.本文关注的是另一个方面:当查询没有结果返回或是返回太少结果时,如何通过改写原始查询,使得新的查询获得好的查全率.这种情况是普遍存在于关键字查询中的,由于用户可能不能准确表达查询意图,输入的查询可能存在拼写错误或不相关的词,这样使得某些关键字在文档中找不到匹配的结点,导致没有结果返回.

收稿日期:

基金项目：国家自然科学基金项目(60833005, 60573091), 国家 863 计划(2007AA01Z155, 2009AA011904, 2009AA01Z133),教育部博士点基金项目(200800020002),教育部重点项目(109004)

本文通讯作者：陆嘉恒(jiahenglu@ruc.edu.cn)

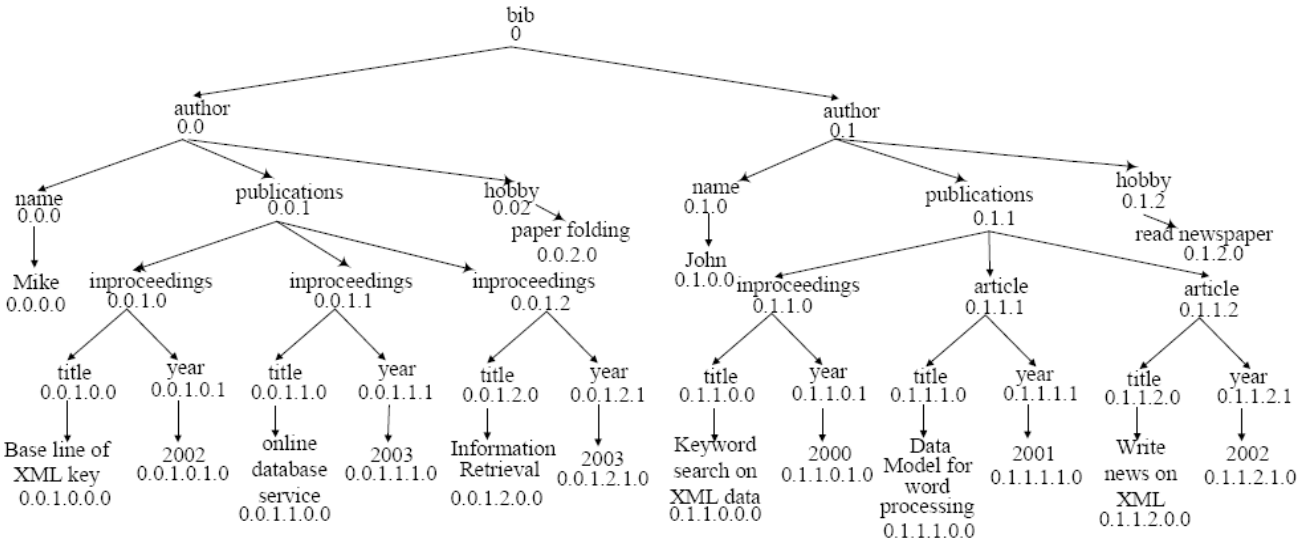



Fig. 1 Example XML document

图 1　带 Dewey 编码的 XML 文档

即使用户准确地表达了查询意图,搜索引擎也可能不能准确地返回用户想要的结果,例如,对于图 1 所示的 XML 文档,用户输入查询“paper, XML”以查找关于 XML 的文章,文档中包含 inproceedings(0.0.1.0)、inproceedings(0.1.1.0)和 article(0.1.1.2)三个解,但由于关键字“paper”在文档中无法找到匹配的结点,以至于没有结果返回.根据文献[6]中的调查统计,用户使用关键字搜索时,大概有 10%-15%的查询存在错误,有 40%-51%的查询要经过修改才能获得所需的信息.我们将改写用户输入的原始查询来获得更好查询结果的问题定义为**查询改写**。

就我们所知,目前的研究工作还没有涉及XML关键字查询改写问题,本文第一次提出了在XML关键字查询中,当用户输入的查询没有结果返回或者只有整个文档作为结果返回时,自动地将原始查询改写成语义相同或相近的查询,并返回有意义的结果.本文采用 SLCA[1]作为关键字查询语义计算结果.例如,对于图 1 所示的XML文档,提交查询“proceedings, XML”和“XML, John, 2003”,前者没有任何结果返回,后者将返回根结点,如果将这两个查询分别改写成“inproceedings, XML”和“XML, John”重新进行查询,前者的查询结果是 inproceedings(0.0.1.0) 和 inproceedings(0.1.1.0);后者的查询结果是author(0.1),明显可以看出改写后的查询获得了更好的查询结果.因此,查询改写不仅可以提高查询质量,而且可以作为查询推荐,帮助用户更快地找到所需的信息.

本文将查询改写和结果生成作为一个完整的问题进行研究,贡献可以总结为以下几点:

1. 形式化的定义了XML关键字查询改写问题,并提出了是否需要查询改写的判断标准.
2. 提出了查询改写的四种操作并设计了动态规划的方法来计算改写原始查询的代价.
3. 提出了基于栈的查询改写算法,计算最小代价的查询改写,同时生成相应的查询结果,只需一趟扫描关键字倒排索引,并能处理不需要查询改写的情况.
4. 进一步设计了基于划分的优化算法,与现有的语义和查询算法具有良好的正交性.
5. 进行了全面的实验,结果表明了本文查询改写方法的有效性和算法的高效性.

本文其余部分的内容安排如下:第 1 部分介绍查询改写问题和四种查询改写操作,并介绍计算查询改写代价的动态规划算法.基于栈的查询算法和基于划分的优化方法在第 2 部分给出.第 3 部分是实验和结果分析.最后,第 4 部分总结本文的工作.

## 1 查询改写

本节给出XML关键字查询改写问题的定义,并介绍四种查询改写操作和查询改写代价计算方法.

### 1.1 XML 关键字查询改写

**定义 1:** 对 XML 文档 $D$ 提交查询 $Q$,如果没有结果返回或者基于 SLCA 语义返回整个文档作为结果,那么就说查询 $Q$ 需要被改写.

例如,对于图 1 所示的 XML 文档来说,表 1 左栏所示的原始查询都是需要被改写.分析这些查询,可以发现主要包括了以下四种情况:查询中包含拼写错误的词或文档中不存在的词;查询中错误地将多个词合并;查询中错误的将一个词拆分;查询中包含太多关键字

以至限制过于严格或包含与查询意图无关的词.

针对这四种情况,我们提出四种相应的查询改写操作:词替换、词拆分、词合并和词删除.表 1 给出了一些查询改写实例,原始查询在图 1 所示的 XML 文档中找不到解或只返回根结点,经过这四种操作之后得到改写后的查询,这些查询能够在图 1 所示的文档中得到非根结点的 SLCA.

Table 1 Examples of query refinement

表 1　查询改写实例

| Original query | Refined query |
|---|---|
| $Q_1$：IR，2003，Mike | $RQ_1$：Information Retrieval，2003，Mike |
| $Q_2$：Mike，publication | $RQ_2$：Mike，publications |
| $Q_3$：Database，paper | $RQ_3$：Database，inproceedings |
| $Q_4$：XML，John，2003 | $RQ_4$：XML，John |
| $Q_5$：hobby，news，paper | $RQ_5$：hobby，newspaper |
| $Q_6$：on，line，data，base | $RQ_6$：online，database |

对于一个查询,可能有多种改写方法,例如,对于表 1 中查询$Q_1$来说,有两种改写方法,一种是使用替换操作,用“Information Retrieval”替换“IR”;另一种是使用删除操作,直接删除“IR”,使得查询变成“2003, Mike”.很明显应该采用第一种改写,因为它的语义更接近原始查询.对于多种查询改写方案,如何确定最优的方案,我们提出查询改写代价*refineCost*($Q$, $RQ$),其中$Q$是原始查询,$RQ$是改写后的新查询,它在一定程度上表示了新查询与原查询在语义上差异.

**定义 2**: (**查询改写代价**)假设查询$Q$通过一系列改写规则$R_1$, $R_2$, … ,$R_r$变成新查询$RQ$, 记作: $Q \xrightarrow{R_1 R_2 \dots R_r} RQ$,其中$R_i$(1≤i≤r)称为改写规则,有如下形式: $k_1, k_2, ..., k_n \xrightarrow{op} k_1', k_2', ..., k_m'$,其中$op$为四种操作之一, $k_i \in Q$或$k_i$=Ø(1≤i≤n), $k_j' \in RQ$或$k_j'$=Ø(1≤j≤m).我们定义查询改写代价*refineCost*($Q$, $RQ$)=$\sum_{i=1}^{r} \mathrm{cost}(R_i)$, 其中$cost(R_i)$为应用规则$R_i$的代价.

因此,查询改写代价就是改写查询时应用的所有改写规则的代价之和.改写规则的生成不是本文的研究重点,可以使用已有的字典(如WordNet)生成.规则的代价可以由系统定义,也可以由用户定义.本文假设使用替换和删除操作的规则代价定义为 2,其他为 1,对表 1 来说,我们可以得到*refineCost*($Q_4$, $RQ_4$)=2; *refineCost*($Q_5$, $RQ_5$)=1.

查询改写的目的是获得有意义的解,根据定义 1,在本文就是非根结点的 SLCA.因此好的查询改写不仅是代价最小的,也要使得新查询有非根结点的解.

**定义 3**:给定 XML 文档 $D$ 和查询 $Q$, $RQ$ 称为 $Q$ 的最优改写,如果不存在$Q$的查询改写$RQ'$, *refineCost*($Q$, $RQ'$)< *refineCost*($Q$, $RQ$), $RQ'$在 $D$ 上有非根结点的 SLCA;且 $RQ$ 在 $D$ 上有非根结点的 SLCA.

因此, XML 关键字查询改写问题就是计算最优查询改写和相应的查询结果.

### 1.2 查询改写代价的动态规划算法

应用不同的改写规则,一个查询可能有多种改写结果,因此需要一个高效的方法计算查询改写代价.

这个问题可以定义为:给定两个关键字序列$S$={ $k_1$, $k_2$,…,$k_s$}, $T$={ $k_1'$, $k_2'$,…,$k_t'$}和改写规则集合$R$,返回将$S$改写成$T$的代价,即*refineCost*($S$, $T$).为了解决这个问题,首先要解决子问题:对于任意 1≤i≤s,计算将$S$的子序列$S$[1, $i$]={ $k_1$, $k_2$,…,$k_i$}改写成$T$的代价.我们提出了一个动态规划方法,如下面方程所示:

$$C[i] = \min \begin{cases} C[i\text{-}1] & \text{if } k_i \in T \\ C[i\text{-}1] + cost\ of\ deletion\ k_i & \text{if } k_i \notin T \\ C[i - |\,l(r_i)\,|] + cost\ of\ applying\ rule\ r_i & \text{if } k_i \notin T \text{ and for } \forall r_i \in R(k_i) \end{cases}$$

在这个方程中,$C[i]$=*refineCost*($S$[1, $i$], $T$),我们的目标是计算$C[s]$,它等于*refineCost*($S$, $T$).对于每个关键字 $k_i \in S$, 对应一个规则集合 R($k_i$)={r| r=<*$k_i$→$k_m$', …,$k_n$'>,其中,*$k_i$是以关键字$k_i$结尾的S的子序列, $k_m$', …,$k_n$' $\in T$}.对于每个规则$r$,用|$l$($r$)|表示规则左边包含关键字的个数.

我们初始化 $C$[0]=0,也就是改写一个空查询的代价为 0.在迭代计算 $C$[i](1≤i≤s)时,考虑以下三种情况,并将得到的最小值赋给 $C$[i]:

（1） 当关键字$k_i$同时出现在$S$和$T$中时,不需要任何改写代价;

（2） 当关键字$k_i$不出现在$T$中,增加删除代价.

（3） 当关键字$k_i$不出现在$T$中,但是有规则$r$可以使用时,追溯至$C$[i-|$l$($r$)|],重新计算$C$[i]的值.如果有多个规则可以使用,取使得$C$[i]值最小的那个.

## 2 查询改写和结果生成算法

给定查询 $Q$,如果根据定义 1 需要进行改写,查询引擎最好能够自动地进行查询改写并计算新查询的结果.我们的目标是只扫描关键字倒排索引一次,在执行原始查询的过程中,有效的计算最优的查询改写和结果.如果不需要改写,则返回原始查询的结果.

### 2.1 基于栈的算法

我们修改了文献[1]中基于栈的算法来解决本文中的问题.主要数据结构栈的设计如下:栈中每个元素由(id, keywords)对组成.假设一个从栈底到栈元素en的id分别为$id_1$, $id_2$,…,$id_m$,那么元素en代表了Dewey编码为$id_1.id_2$.….$id_m$的结点.keywords是布尔值数组.栈中元素的keywords[i]=T表示以该元素表示的结点包含关键字$k_i$,否则为F.根据改写规则,将互相替换的关键字共享同一列;将拆分操作后涉及的多个关键字共享一列,这列中只有当所有的关键字都被包含时keywords才赋值T.

由于在扫描完关键字索引之前,不确定是否需要改写和最优改写,为了只处理结点一次,我们需要保存当前最优解,成为查询改写候选,当发现更优候选时,进行替换,最后得到整体最优解.具体算法如图 2 所示,算法中的 *result* 在运行过程中保留了当前最优解.

```
Input: query Q={k1, k2,…, kn} and XML document D
Output: result={(RQ1 ={k1', k2',…, kt'}, SLCA(RQ1)),
                (RQ2 ={k1'', k2'',…, kt''}, SLCA(RQ2)),…}
1   result ← Ø;   stack ← Ø;
2   result.cost=deletion.cost*sizeof(Q);
3   {K1, K2, …, Km} ← getTerms();
4   {S1, S2, …, Sm}← getNodeLists();
5   while Si is not empty or stack is not empty do
6       vs = getSmallestNode();
7       p = lca (stack, vs );
8       while (stack.size > p.length) do
9           entry = stack.pop();
10          if (entry!=root) then
11              for (i =1→m) do
12                  if (entry.keywords[i] = = true) then
13                      tempR ← K[i] ;
14                      stack.top.keywords[i] = true;
15                  end if
16              end for
17              tempR.cost = refineCost(Q, tempR);
18              if (tempR.cost ≤ result.cost) then
19                  add {tempR, entry} into result ;
20                  result.cost = tempR.cost;
21              end if
22          end if
23      end while
24      for (j = p.length → vs.length) do
25          stack.push(vs[p], vs[length]);
26      end for
27      stack.top.keywords[k] = true;
28  end while
29  return result;
```

Fig. 2 Stack-based algorithm for query refinement

图 2 基于栈的查询改写和结果生成算法

这个算法与文献[1]中的基于栈的算法有如下明显的区别:首先,我们需要根据当前查询和可以使用的规则,获取涉及的所有关键字,对于没有倒排索引的关键字进行删除(3-4 行);其次,根据改写条件,我们摒弃了以根结点为解的情况(行 10);第三,对于每次出栈的非根结点元素 *entry* 都检验其包含的关键字,形成改写候选 *tempR*,并计算此时的改写代价(11-17 行);第四,如果改写候选的代价是当前最小的,就将 *tempR* 和 *entry* 放入 *result* 中,并修改最小代价阀值(18-21 行),这里在 19 行需要检验 *entry* 是否是 *tempR* 的 SLCA,只有 SLCA 才放入 *result* 中.因为当算法找到一个当前解时,由于不能确定这个解是否是整体最优解,并没有将栈中剩下的元素的 keywords 置为 F.最后,*result* 中的结果就是最优查询改写和对应的查询结果.如果原始查询不需要改写,那么 *result* 中就是原始查询的结果.

由于在扫描完关键字索引之前不能确定最优的查询改写,因此,在最坏的情况下,这个算法需要将所有的查询改写候选计算一遍.因此,算法复杂度是 $O(d\sum_{i=1}^{m}|S_i|)$,其中,$m$是使用改写规则后涉及的关键字个数, $S_i$是直接包含关键字$k_i$的结点集合, $d$是XML树的深度.

为了减少无用的候选计算,我们提出了基于划分的优化方法.这个方法能尽早确定最优查询改写,能避免不必要的 SLCA 计算.

### 2.2 基于划分的优化算法

根据定义 1,XML 文档根结点可以排除在 SLCA 的计算之外,除了根结点之外的结点都能看作是有意义的 SLCA,于是,基于划分的优化方法不再将文档整体考虑,而是先对文档进行分割,在文档片断上推断查询改写,然后对最优的改写进行计算.

**定义 4**:(**文档划分**)将XML文档$D$除去根结点$r$得到的子树片断$D_1$, $D_2$,…,$D_m$称为$D$的一个划分,记作$D(r, D_1, D_2,…,D_m)$,它具有以下特点:

(1) $r \cup D_1 \cup D_2 \cup \ldots \cup D_m = D$

(2) $\forall i, j \in [1, m], i \neq j, D_i \cap D_j = \emptyset$

根据定义 4,图 1 中的 XML 文档划分为两部分:一个是以 author(0.0)为根结点的子树,另一个是以 author(0.1)为根结点的子树.

如果要查询$Q$在文档$D$上有非根结点的SLCA,那么$Q$必然在$D$的某个子树片断$D_i$上有解.根据这个性质和定义 3 可以得出:对于文档$D(r, D_1, D_2,…,D_m)$和查询$Q$,如果$RQ$是最优查询改写,那么$\exists i \in [1, m]$, 对于$\forall k \in RQ$, $k \in D_i$.

因此,寻找最优查询改写可以先将文档进行划分,然后找出包含最小代价查询改写的子树片断,并计算改写后的查询在此子树片断上的查询结果即可.将文档进行划分以后,便可轻松获得查询改写候选:由于同

属于一个片断的结点 Dewey 编码前两位相同,因此,我们以这两位编码作为一个片断的标识,记作 *pid*,在倒排索引中用 *pid* 可以找出同属一个片断的结点,也可以得出这个片断包含的关键字,即为一个改写候选.获得所有查询改写候选后,只要选取代价最小的候选在相应的文档片断上计算 SLCA 即是最优查询改写和结果,而不用计算其他候选的 SLCA,这就是基于划分的优化方法.具体算法如图 3 所示.

**Input:** query $Q=\{k_1, k_2,\cdots, k_n\}$ and XML document $D$
**Output:** $result=\{(RQ_1=\{k_1', k_2',\cdots, k_t'\}, \mathrm{SLCA}(RQ_1)), (RQ_2=\{k_1'', k_2'',\cdots, k_t''\}, \mathrm{SLCA}(RQ_2)),\cdots\}$

```
1  result ← Ø;
2  result.cost=deletion.cost*sizeof(Q);
3  {K1, K2, ⋯, Km}← getTerms();
4  {S1, S2, ⋯, Sm}← getNodeLists();
5  while Si is not empty do
6      vs = getSmallestNode();    /* vs ∈ Ss */
7      pid = getDocPartitionID( vs );
8      {S1', S2' , ⋯, Sm' }← getKLPartition(pid);
9      for (i =1→m) do
10         if (Si' ≠ null) then
11             RQ ← Ki ;
12         end if
13     end for
14     RQ.cost=refineCost(RQ,Q);
15     if (RQ.cost ≤ result.cost) then
16         slcas = computeSLCAs ({S1', S2' , ⋯, Sm' }, RQ);
17         result ← {RQ, slcas};
18         result.cost = RQ.cost;
19     end if
20     remove (S1', S2' , ⋯, Sn') from {S1, S2, ⋯, Sm};
21     RQ ← Ø;
22 end while
23 return result;
```

Fig. 3 Partition-based algorithm for query refinement

图 3 基于划分的优化算法

首先根据原始查询和改写规则获得相关的关键字,并根据倒排索引获得包含这些关键字的结点列表(3-4 行).然后从结点列表中找到编码最小的结点,取这个结点 Dewey 编码的前两位 *pid* 就可以作为文档片断的标识,从结点列表中取出编码前两位为 *pid* 的所有结点子列表,也即为同一文档片断的结点(6-8 行).如果关键字对应的子列表不为空,则说明此片断包含此关键字,由此可以确定改写候选,然后通过 1.2 节介绍计算改写代价的方法计算候选改写的代价(9-14 行).对于比当前代价阀值小的改写候选用已有的方法计算 SLCA,修改代价阀值(15-19 行).将结点子列表移除(行 20).这个过程一直循环到结点列表中没有任何结点为止,返回结果.

这个优化的算法在推断查询改写候选时方法简单快捷,只需取结点编码前两位并判断结点集合是否为空即可.可以跳跃某些结点不进行SLCA计算,以达到优化的目的.在计算SLCA时,可以直接运用现有工作中的算法[1, 7].基于划分的思想具有良好的扩展性,可以运用到基于其他语义的查询改写算法中.

## 3 实验与分析

我们进行了丰富实验,验证了本文查询改写方法的有效性和算法的高效性.

所有实验都是在处理器为Intel双核 2.0GHz,内存为 1GB,操作系统为Windows XP Professional机器上运行,实现语言为 Java. 我们在 SIGMOD Record[8] 和 DBLP[9]两个数据集上进行了实验,大小分别为 500KB 和 130MB.

### 3.1 改写质量

我们使用一些需要改写的查询来验证本文查询改写方法的有效性,改写规则是根据本文提出的四种改写操作人工生成的.原始查询没有任何结果返回,因此本文对比改写后的查询和相应的结果个数来评价改写方法的质量.表 2 和表 3 列出了一些改写示例.原始查询没有结果返回或返回整个文档作为结果,改写的查询结果个数如表所示.

Table 2 Sample query refinement on SIGMOD Record

表 2 SIGMOD Record 上查询改写示例

| id | Original query | Refined query | Result size |
|---|---|---|---|
| $Q_1$ | data, base, Codd | database, Codd | 2 |
| $Q_2$ | data, mining, jia, wei, han | data, mining,, han, jiawei, | 3 |
| $Q_3$ | paper, net, work | article, network | 6 |
| $Q_4$ | real, time, systems | real-time, systems | 9 |
| $Q_5$ | keyword, search, database, rank | search, database | 11 |
| $Q_6$ | adhoc, article | article, ad, hoc | 1 |

Table 3 Sample query refinement on DBLP

表 3 DBLP 上查询改写示例

| id | Original query | Refined query | Result size |
|---|---|---|---|
| $Q_7$ | xml, model, 1995, view | model, 1995, view | 3 |
| | | xml, model, view | 1 |
| $Q_8$ | data, mining, 2006, jia, wei, han | data, mining,, han, jiawei, | 14 |
| $Q_9$ | paper, net, work, 1990 | article, network, 1990 | 59 |
| | | inproceedings, network, 1990 | 88 |
| $Q_{10}$ | fulltext, search | search, full, text | 4 |
| $Q_{11}$ | keyword, search, rank, xml | keyword, search, xml | 2 |

| | | | |
|---|---|---|---|
| $Q_{12}$ | refinement, keyword , search, IR | keyword ,search | 9 |
| | | keyword, refinement | 5 |

从表 2 和表 3 的示例可以看出,本文的改写方法可以对原始查询进行有效的改写,改写后的查询在语义上与原始查询具有最大的相似性,并返回有意义的结果.本文的算法可以支持四种操作的任意组合,可以根据需要来选择.有些查询具有多种最优改写,例如$Q_7$、$Q_9$和$Q_{12}$,算法将返回所有这些改写和它们对应的结果.

### 3.2 执行时间

我们通过对查询改写的执行时间来比较本文两个算法的效率.图 4 和图 5 分别是表 2 和表 3 所示的查询改写的执行时间.其中,“stack-refine”代表本文基于栈的改写算法,“partition-refine”代表本文基于划分的优化算法.

从图 4 和图 5 可以看出,基于划分的优化算法比基于栈的算法效率更高,因为它可以避免不必要的 SLCA 计算.

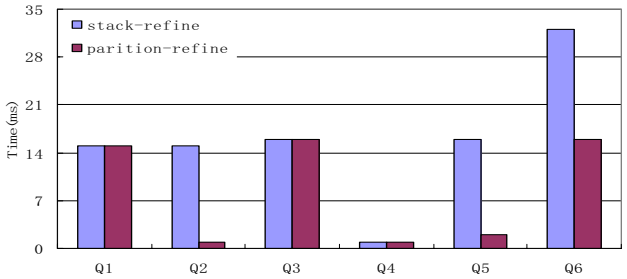



Fig. 4 Processing time on SIGMOD Record

图 4　SIGMOD Record 上查询改写示例执行时间

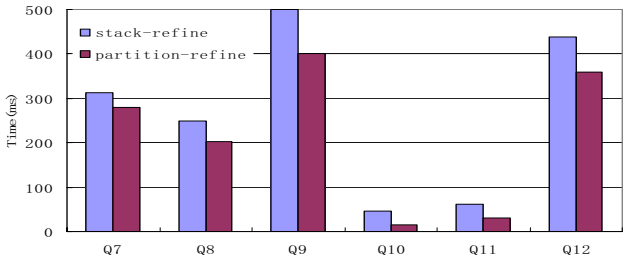



Fig. 5 Processing time on DBLP

图 5　DBLP 上查询改写示例执行时间

为了验证本文方法处理正常查询的效率,我们选择了一些不需要改写的查询与文献[1]提出的基于栈的计算 SLCA 的算法进行了效率对比.图 6 是在 DBLP 上的平均执行时间对比.其中“stack-SLCA”代表文献[1]中基于栈的算法.

从图 6 可以看出,本文的两个算法都能处理不需要改写的查询.基于栈的查询改写算法要比基于栈的计算 SLCA 算法耗费更多一些时间,因为它考虑了可能出现需要改写的情况,会生成改写候选,并计算改写代价.从整体来看,改写算法提高了查询的质量,平均多耗费 10%的时间在可以接受的范围内.而基于划分的算法则比其他两个算法效率都高,因为,它可以避免在某些无解的划分片断上的计算.

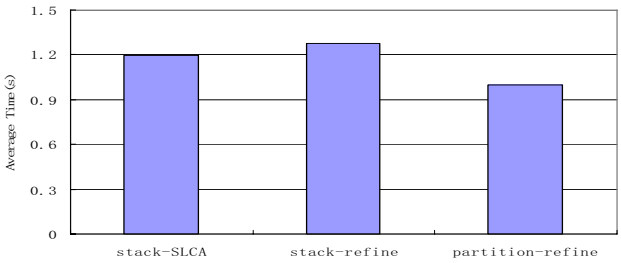



Fig. 6 Average processing time on DBLP

图 6　DBLP 上平均执行时间

### 3.3 可扩展性

我们分别在 10M, 20M, 70M 和 130M 的 DBLP 文档上执行一些查询,并计算平均执行时间来验证本文两个算法的可扩展性,结果如图 7 所示.可以看出,随着文档大小的递增,平均执行时间基本呈线性增长,因此我们的算法具有良好的可扩展性.

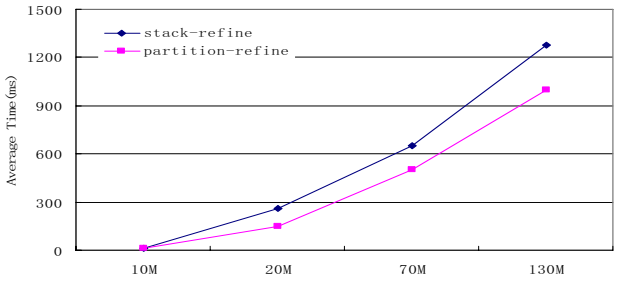



Fig. 7 Processing time with increasing document size

图 7　平均执行时间随数据集大小变化关系

## 4 结论

本文研究了 XML 关键字查询改写问题,查询改写不仅能够提高查询质量,并且能够帮助用户更快的确定查询目标.本文针对用户输入的没有查询结果的查询,通过词删除、替换、拆分和合并四种操作进行改写.对于多个查询改写通过改写代价来衡量它们的优劣,并设计了动态规划方法计算改写代价.为了有效计算最优查询改写并生成相应的结果,我们提出了基于栈的算法和基于划分的优化算法.这两个算法都只需扫描结点列表一次,并能处理不需要改写的情况.特别是基于划分的思想能应用于其他查询语义,据有很好的扩展性.实验结果进一步验证了本文方法的有效性.未来工作可以考虑在原始查询增加关键字的操作来改写查询.

## 参 考 文 献

[1] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Database [C] // Proc. of ACM SIGMOD 2005. New York: ACM Press, 2005: 527-538

[2] S. Cohen, J. Mamou, Y. Kanza and Y. Sagiv. XSEarch: A Semantic Search Engine for XML [C] //Proc. of VLDB 2003. Berlin: Morgan Kaufmann, 2003: 45-56

[3] G. Li, J. Feng. J, Wang, and L. Zhou. Effective Keyword Search for Valuable LCAs over XML Documents [C] //Proc. of ACM CIKM 2007. Lisbon: ACM Press, 2007: 31-40

[4] S. Cohen, Y. Kanza, B. Kimelfeld, and Y. Sagiv. Interconnection Semantics for Keyword Search in XML [C] //Proc. of ACM CIKM 2005. Bremen: ACM Press, 2005: 389-396

[5] Huang Jing, Xu Junjin, Zhou Junfeng and Meng Xiaofeng. MLCEA: An Entity Based Semantics for XML Keyword Search [J]. Journal of Computer Research and Development, 2008, 45(Supplement): 372-377(in Chinese)

(黄静, 徐俊劲, 周军锋, 孟小峰. MLCEA:一种基于实体的 XML 关键字查询语义[J]. 计算机研究与发展, 2008, 45(增刊):372-377)

[6] A. Spink, B.J. Jansen, D. Wolfram, and T. Saracevic. From E-sex to E-commerce: Web Search Changes [J] IEEE Computer, 2002, 35(3):107-109

[7] Chong Sun, Chee-Yong Chan, Amit K. Goenka. Multiway SLCA-based Keyword search in XML Data [C] //Proc. of WWW 2007.Banff: ACM Press, 2007: 1043-1052

[8] UW XMLRepository [OL]. [2009-04-20] http://www.cs.washington.edu/research/xmldatasets/www/repository.html

[9] DBLP XML Repository [OL]. [2009-04-20].http://dblp.uni-trier.de/xml/

**Huang Jing** (huangjingruc@ruc.edu.cn), femail, born in 1984, Master. Her main research interests include XML database and XML keyword search.

黄静(huangjingruc@ruc.edu.cn),女, 1984 年生,硕士,主要研究方向为 XML 数据库、XML 关键字查询

**Lu Jiaheng,** male, born in 1975, Associate Professor. His main research interests include XML data management, approximate string matching and cloud computing technology.

陆嘉恒,男, 1975 年生,副教授,主要研究方向为 XML 数据管理、字符串模糊匹配和云计算技术等。

**Meng Xiaofeng,** male, born in 1964, Professor and Ph.D. supervisor, He is the Secretary General of Database Society of the China Computer Federation (CCF DBS). His main research interests include Web data integration, XML database, mobile data management and flash database.

孟小峰, 男,1964 年生,教授,博士生导师,现为中国计算机学会理事、中国计算机学会数据库专委会秘书长. 主要研究领域为 Web 数据集成,XML 数据库,移动数据管理,闪存数据库等.

**Research Background**

This research was partially supported by the grants from the Natural Science Foundation of China under grant number 60833005, 60573091; National High-Tech Research and Development Plan of China (No:2007AA01Z155, 2009AA011904, 2009AA01Z133) and Key Project in Ministry of Education(No: 109004).

As XML is gradually becoming the standard in exchanging and representing data, effective and efficient methods to query XML data has become an increasingly important problem. XQuery can convey complex semantics meanings and therefore retrieve precisely the desired results. However, in order to write the right query, the user should know the underlying data schema and master the complex query syntax. Keyword search enables users to easily access XML data without the need to learn a structured query language and to study complex data schemas. So keyword search over XML data has attracted a lot of research efforts. However, no existing work has touched the field of automatic query refinement for XML keyword search yet. In this paper, we first define the problem of keyword query refinement in XML keyword search, in which four refinement operations are defined, namely term deletion, merging, split and substitution. In order to achieve the goal of finding the best refined queries and generating their associated results within a one-time node lists scan, we propose a stack-based algorithm, followed by a partition-based optimization.

In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009), page1457-1460, November 2-6, 2009, Hong Kong, China

# RS-Wrapper: Random Write Optimization for Solid State Drive

Da Zhou
Information School
Renmin University of China
Beijing 100872, China
cadizhou@gmail.com

Xiaofeng Meng
Information School
Renmin University of China
Beijing 100872, China
xfmeng@ruc.edu.cn

## ABSTRACT

Solid State Drive (SSD), emerging as new data storage media with high random read speed, has been widely used in laptops, desktops, and data servers to replace hard disk during the past few years. However, poor random write performance becomes the bottle neck in practice. In this paper, we propose to insert unmodified data into random write sequence in order to convert random writes into sequential writes, and thus data sequence can be flushed at the speed of sequential write. Further, we propose a clustering strategy to improve the performance by reducing quantity of unmodified data to read. After exploring the intrinsic parallelism of SSD, we also propose to flush write sequences with the help of the simultaneous program between planes and parallel program between devices for the first time. Comprehensive experiments show that our method outperform the existing random-write solution up to one order of magnitude improvement.

**Categories and Subject Descriptors:** H.2.2 Database Management: Physical Design-Access methods

**General Terms:** Algorithm, Design, Performance

**Keywords:** Flash Memory, Database, Random Write, Parallelism

## 1. INTRODUCTION

SSD, emerging as a new electronic storage device, is widely adopted in laptops and personal computers during the past few years. This mainly benefits from the high read performance of SSD, especially random read performance. As we know, SSD does not has mechanical part like the magnetic head of Hard Disk Drive (HDD), therefore there is no latency for random read of SSD. As a result, random read has similar speed with sequential read. This characteristic improves the read performance of system fundamentally. Besides this, SSD has other attractive characteristics, such as low power consumption, high shock resistance and lightweight form. All of these advantages make SSD as outstanding data storage instead of HDD.

However, the random write performance of SSD, especially small random writes, is very poor as shown in table 1. Read and sequential write is faster than random write in two orders. The costly erase operation of flash memory lays down the main reason for the slow performance of random writes. Erase is peculiar to flash memory. In a word, erase operation has two important characteristics: erase before rewrite and high cost. We must erase the whole block if we want to rewrite pages of it. On the other hand, the cost of erase is 1.5ms, while that of write is only 0.22ms according to Micron electronics datasheet[3]. Besides the cost of erase itself, large quantity of data need to be transferred before erase operation is executed. Although erase operation has high cost and leads to low write performance, every small updates will still trigger erase operation in the worst situation. FTL embedded in SSD can reduce the number of erase by implementing out-of-place update with the help of Physical-to-Logical mapping[4], but the efficiency is very low according to random write performance as shown in table 1. As a result, low performance of random write becomes the bottleneck of wider applications of SSD.

**Table 1: IO Performance Values of Mtron SSD[5]**

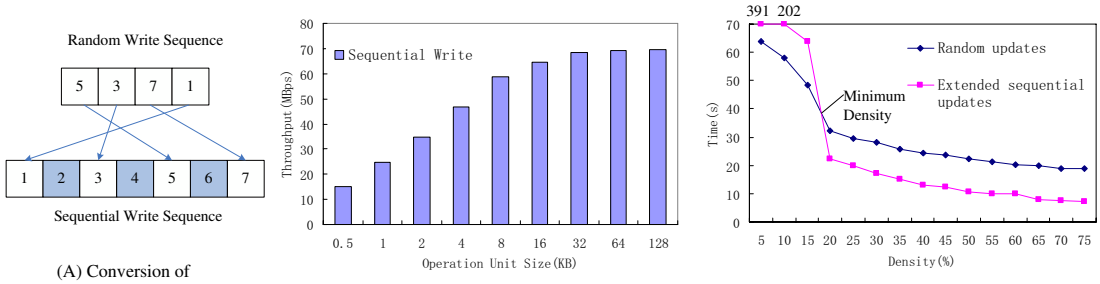
| Sequential Read | Sequential Write | Random Read | Random Write |
|---|---|---|---|
| 11,100 | 16,600 | 11,200 | 120 |

In this paper we propose a novel and efficient method for avoiding random write. Based on the key observation that SSD has high sequential write performance, our method avoids random writes by converting random write sequence into sequential write sequence in order to take full advantage of high performance of sequential write. We novelly insert unmodified data into the random write sequence, which locate between the lower and upper limit of the random write sequence. Finally we flush the constructed sequential write sequence into SSD instead of original random write sequence. Compared with flushing random write sequence, cost of writing the converted sequential write sequence is much lower. Therefore the write performance is enhanced obviously. Density and granularity of write sequence are two key factors of the efficiency of our method in this paper. We also propose to reduce the cost of getting addresses and reading unmodified data further by cluster when density is low than MD(Minimum Density).

We also propose to improve write performance by intrinsic parallelism of SSD further. SSD contains several flash memory devices and one chip is made up of a number of planes. According to this, we explore the simultaneous program between planes and parallel program between devices for the first time. With help of intrinsic parallelism of SSD, we partition sequential write sequence and flush partitions in parallel. Results of comparison experiments show the write performance are enhanced obviously, especially for sequence with low density.

The rest of this paper is organized as follows. Section 2 explains


*CIKM'09,* November 2–6, 2009, Hong Kong, China.




(A) Conversion of Static Random Write Sequence

(B) IO Performance Test of Sequential Write

(C) Write Performance Test on Density

**Figure 1: Conversion of static random write sequence and performance experiments**

the basic idea. Section 3 expresses our method in detail with data stream. We also enhance performance further in section 4. Experiments are shown in section 5. Finally, section 6 concludes.

## 2. STATIC RANDOM WRITE SEQUENCE

In this section, we first describe the basic idea of our method by static random write sequence, and then discuss how density and granularity impact the efficiency of our method. Finally, we optimize our method according to density.

### 2.1 Converting Random Write Sequence

Static random write sequence is defined as a random write sequence without changes during the course of conversion. We insert unmodified data into the random write sequence, which locate between the lower and upper limit of the random write sequence. So we flush the constructed sequential write sequence into SSD instead of original random write sequence. Therefore the cost of write will be reduced due to high sequential write performance. For example, the random write sequence has four data items as shown in Fig. 1(A). The addresses of these items are 5, 2, 7 and 1. It may take four random writes if we flush this write sequence into SSD directly. According to table 1, the cost will be 33 ms. However, after we extend the random sequence with unmodified data items, item 2, 4 and 6, we can flush the sequential write sequence as shown in Fig. 1(A) sequentially. As a result, we only cost seven sequential writes. The cost of flushing is only 0.4 ms ideally. Although the approach is straightforward in concept, when one actually attempts to implement such a facility, one is faced with myriad options and difficult decisions every step of the way. One of important problems is how to decide the granularity and density of random write sequence. We will explore the impaction as follows.

### 2.2 Granularity and Density

Granularity is defined as the write unit when we flush the converted sequential write sequence into SSD. As shown in Fig. 1(B), the throughput becomes higher when we increase the write unit size. The results shows that the write performance of SSD is not taken full use of when the write unit size is less than 32KB. Therefore, 32KB is the optimum granularity for our SSD to perform sequential write. According to this experiment, we define this special write unit as *OSWG*(Optimum Sequential Write Granularity). Different SSDs have different OSWGs.

Density is defined as the length ratio of random write sequence to the converted sequential write sequence. Density is direct ratio to the performance. The less the density is, the more unmodified data items are inserted, and the larger the cost of reading and flushing these data is. According to different density, we get runtime of flushing random write sequence and the converted sequential write sequence. According to the experiment results as shown in Fig. 1(C), the larger density is, the higher the efficiency of our method is. We define the density of intersection point as *MD*. Our method outperforms random write about 100%∼150% when density is larger than MD. Different SSDs have different MDs. The MD of the Mtron SSD in our lab is 18%.

### 2.3 Optimization

According to Fig. 1(C), our method is worse than random write when density is less than MD. For example, when the density is 10%, the runtime of our method is as high as 201.7 seconds. In our method, the runtime of getting addresses of each data item, reading unmodified data items and flushing converted sequential write sequence are 6.4, 50.7 and 144.6 seconds, respectively. the In this case, we need to read too much unmodified data, and then flush them into SSD. In a word, the large quantity of unmodified data lead to low performance of our method.

A sub-sequence is defined as a *cluster* if the density of it is larger than MD. After sorting the random write sequence according to addresses, we generate clusters as follows. Firstly we decide the first data item as a cluster. Secondly, this cluster and its next data item are treated as a new sub-sequence. If the density of it is larger than MD, the next data item are grouped into this cluster. If not, the next data item will be grouped as a new cluster. This course cycles until all data items are grouped into clusters. After getting clusters, we convert these clusters into sequential write sequence instead of the whole write sequence. We re-execute the experiments with the help of clusters when density is less than MD. Experiments show our method outperforms random write about 10%. This attributes to obvious decrease of the quantity of unmodified data to read, and then the costs of random read and sequential write decrease.

## 3. DATA STREAM

Section 2 explains our basic idea by static random write sequence. However real applications usually generate writes continuously as data stream. According to characteristics of dynamic write sequence, we first load and initialize the write sequence, and then generate initial clusters. Lastly the final clusters are evolved from initial clusters and flushed into SSD. The above steps repeat until the end of data stream. We will explain each steps in following.

### 3.1 Initialization

After loading writes from data stream, we need to calculate density. The address of each data item is used to calculate the density of write sequence. Therefore, we need to get the address of each data item in the first step. Actually, this step takes full ad-

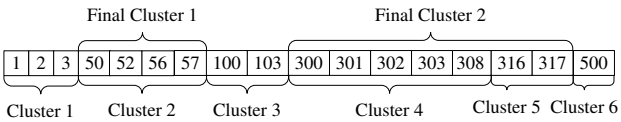



**Figure 2: Initialized random write sequence. Data items are grouped into initial clusters , and initial clusters are merged into final clusters.**

vantage of characteristics of SSD. During the course of getting an address of a data item, we need to look up it from index, and so several random read operations must be executed. Compared with HDD, SSD gets high performance because of its high random read performance, and then the cost of getting address is low for SSD. After getting addresses of data items, we need to sort the write sequence in cache according to addresses and generate initial clusters as shown in Fig. 2. The numbers denote the addresses of data items.

## 3.2 Final Cluster

The design objective of final cluster is to avoid small granularity of write. Final cluster is defined as the random write sub-sequence, the length of which is larger than OSWG and the density of which is larger than MD. The basic design principle is the length should be as long as possible in order to take full advantage of high sequential write speed of SSD. Therefore we will get final clusters by merging initial clusters. There are two cases according to the maximum length of initial clusters: The maximum length is no less or less than OSWG. We will discuss them in following.

*No Less Than OSWG.* The initial clusters are defined as candidate clusters if their lengths are no less than OSWG. In order to lengthen the final cluster, we need to merge adjacent initial clusters into candidate clusters with the grantee of density. The detail step of merger is described as following. For each candidate cluster, we first merge preceding initial clusters into candidate cluster. If the density of merged cluster is less than *MD*, the course of the merger will be terminated. In the same way, the following initial clusters are merged. For example, as to cluster 4 in Fig. 2, preceding clusters will be merged into cluster 4 firstly. Because the density of cluster 3∪cluster 4 is less than MD, the course of merging preceding clusters is terminated. As for following initial clusters, cluster 5 will be merged with cluster 4. At last we get final cluster 2.

*Less than OSWG.* We need to merge initial clusters into final clusters when lengths of all initial clusters are less than OSWG. [cluster i, cluster j] means the cluster union from cluster i to j. In order to get the maximum length efficiently, we use the method of top to down. For example, all initial clusters are shown in Fig. 3. Firstly, the density of [clusters 1, cluster 6] is calculated. If the density is larger than MD, the course of merger will be terminated. If not, the course will continue. For example, the density of [clusters 1, cluster 5] is calculated. This course will be continued until the cluster union only contain two clusters. As shown in Fig. 3, final clusters will be gotten in the last step. The density of [cluster 1, cluster 2] is larger than MD and length is larger than OSWG, they are merged into final cluster 1. So does the Final cluster 2.

# 4. OPTIMIZATION WITH INTRINSIC PARALLELISM

In this section we will explore the intrinsic parallelism of SSD, and then optimize our method further.

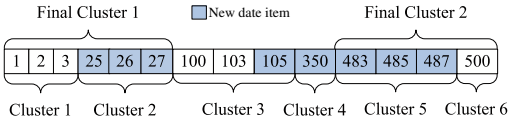



**Figure 3: Random write sequence with lengths of all initial clusters are less than OSWG.**

## 4.1 Intrinsic Parallelism

After disassembling the SSD used in my lab, we can see that sixteen flash devices are arrayed on the circuit board, and the part number is MT29F8G08DAA. Therefore we assure that parallelism exists between flash devices because each device can be operated individually. Besides this, parallelism also exists in the interior of one flash device. According to specifications[3], one flash device contains two CE#s(Chip Enable) which has two planes. Both CE#s and planes can be accessed in parallel. As to smaller storage unit, plane is made up from blocks which contain pages.

In order to testify the parallelism, we design five experiments. Each experiment will write 131072 pages of data into SSD. Experiment 1 sequentially writes pages in a single plane. Experiment 2, 3 and 4 alternately and sequentially write pages between two planes, two CE#s and two devices. Finally, experiment 5 sequentially write pages according page NO. We allocate storage area sequentially in logic layer and suppose it is physically sequential. The experiment results show the reasonableness of our assumption.

The runtime of experiment 1, 2, 3, 4 and 5 are 28.2, 25.8, 23.2, 23 and 25.2 seconds, respectively. The write performance of experiment 1 is less than that of experiment 2 about 10%. The reason is experiment 1 only can write data one by one. However, experiment 2 can write data into two planes of the same CE# at the same time by TWO-PLANE PROGRAM. So do experiments 3 and 4. Finally we find that sequential write almost has the same performance with experiments 2. Because even-numbered and odd-numbered blocks belong to different planes, sequential write also utilizes the TWO-PLANE PROGRAM to speed up write as experiment 2. But sequential write does not utilize intrinsic parallelism between CE#s and devices, so the performance is lower than experiment 3 and 4.

## 4.2 Optimization

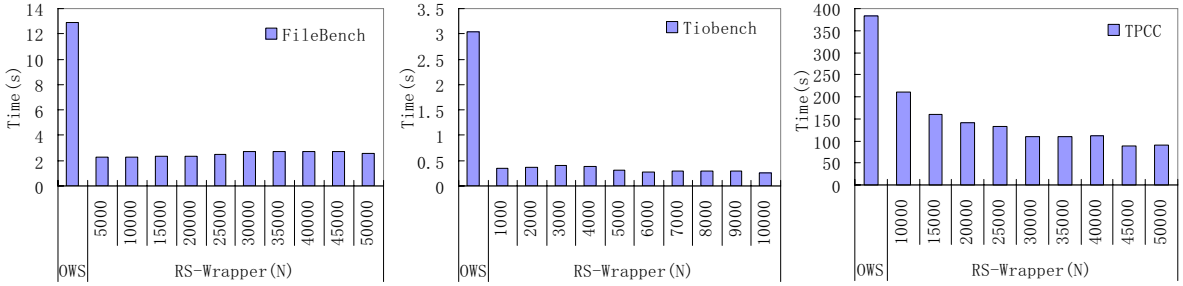
According to above experiments, we propose to speed up our method further with intrinsic parallelism between CE#s and devices. Data items in write sequence are programmed alternately into flash devices if they belong to different CE#s or devices. For example, we suppose the final cluster 1 and 2 in Fig. 2 belong to different devices. As to previous method, we will flush final cluster 1 firstly, and then final cluster 2. However, we change the write sequence according to the parallelism of SSD. Final cluster 1 and 2 will be written alternately. The write sequence will be organized as 50, 300, 51, 301.....57, 307, 308, 309.....317. After re-organizing the write sequence, the parallelism between devices is triggered, and then the write performance is further improved.

# 5. PERFORMANCE EVALUATION

In this section we will firstly introduce the hardware platform and benchmarks in our experiments. In the next, comparison experiment results are shown and analyzed.

## 5.1 Experiments Setup and Workloads

We implement our experiments on a desktop PC powered by Intel Core 2 Pentium 4 Duo CPU 2.83GHz running Linux fedora 8




(A) Experiment on IO trace of FileBench (B) Experiment on IO trace of Tiobench (C) Experiment on IO trace of TPCC

**Figure 4: Comparison experiments about performance of RS-Wrapper.**

with 2GB main memory. The kernel version is Linux-2.6.23. The SSD used in our experiments is 16GB Mtron SSD(MSD-SATA 3035-016). We use three benchmark tests to testify the IO performance enhancement brought by our RS-Wrapper. We also use blktrace[1] to trace the IO activities at the block level. After running benchmarks, we get the IO activities, *OWS* (Original Write Sequence). In our comparison experiments, we firstly write OWS into SSD directly. Secondly, we re-organize OWS by our RS-Wrapper and flush the new sequences. For each benchmark, a series of experiments are run by varying the length of write sequence.

## 5.2 File System Benchmarks

FileBench[2] is a framework of workload for measuring file system performance. In our experiments, the number of files is set 50,000, 50,000 and 25,000 for creatfiles, deletefiles and copyfiles respectively. Besides this, both the file size and IO size are set as 2 kilobytes. We also convert the OWS by our method when the length of sequence varies from 5,000 to 50,000. The runtime is plotted in Fig. 4 (A). Our method outperforms OWS about 600% as a whole. The main reason is that IO operations of a file are basically sequential. In this case, the density of final cluster is very high in our method. Therefore, our RS-Wrapper gains high performance.

## 5.3 File IO Benchmarks

IOzone[6] is a file system benchmark tool. In order to simulate the real workload mostly, we select the full automatic mode. The IO operations cover all tested file operations for record sizes of 4KB to 16MB for file sizes of 64KB to 512MB. The comparison experiment results are shown in Fig. 4(B). With the length of write sequence varying from 1,000 to 10,000, our RS-Wrapper is faster than OWS about 800% as a whole. As for IOzone, the IO activities are random and converge in a limited range. In common test, the maximum size is 512MB. Therefore, every writes are random, and thus the write cost of OWS is very high. However, converted sequential sequence has high density. Therefore our method takes full advantage of, and then performance is enhanced obviously.

## 5.4 TPCC Benchmark

In order to test the write performance enhancement on databases, we select the typical write intensive benchmark, TPC-C. In this experiment, write sequence comes from the disk IO operations by running TPC-C on PostgreSQL database system version 8.3.5. The operation system is Red Hat Linux 2.6.27. After setting the number of warehouses as 50, page size as 8KB, the number of threads as 200, we run TPC-C 30 minutes. We modify postgreSQL to record disk IO operations when executing the routines of writing data to disk. In the same way, we flush IO trace by two ways. The experiment results are shown as Fig. 4(C). Our method outperforms OWS about 300% when length N is no less than 25000. The high performance is obtained because our method converges random writes into high density clusters and flushes them in parallel.

# 6. CONCLUSIONS

SSD is applied widely due to its high random read speed and sequential access performance. However, poor random write leads to the low performance of write-intensive applications on SSD. We novelly propose to extend random write sequence into sequential write sequence by inserting unmodified data into write sequence. Firstly, we explore the impact of density and length on performance of our method with static random write sequence. Secondly, we optimize our method with cluster which reduces the quantity of data to read and flush during the course of conversion. Thirdly, we improve the performance in further by merging initial clusters into final clusters. Finally, we improve write performance with its intrinsic parallelism. The experiments show our method improves write performance of SSD obviously under all tested workloads.

# 7. ACKNOWLEDGMENT

We would like to thank Yinan Li and Prof. Qiong Luo for their help in the experiments, Prof. Lei Chen and Jianliang Xu for their constructive comments, and Prof. Jiaheng Lu for his revision. This research was supported by the grants from the Natural Science Foundation of China under grant number 60833005.

# 8. REFERENCES

[1] J. Axboe, A. D. Brunelle, and N. Scott. blktrace(8) - linux man page, 2006. `http://linux.die.net/man/8/blktrace`.

[2] R. McDougall, J. Crase, and S. Debnath. Filebench: File system microbenchmarks, 2006. `http://www.opensolaris.org/os/community/performance/filebench/`.

[3] Micron. Nand flash memory mt29f4g08aaa, mt29f8g08baa, mt29f8g08daa, mt29f16g08faa, 2007. `http://download.micron.com/pdf/datasheets/flash/nand/4gb_nand_m40a.pdf`.

[4] Mtron. Solid gear | mtron ssd technology, 2008. `http://www.solidgear.sg/technology/mtron-ssd-technology.php`.

[5] Mtron. Solid state drive msd-sata 3035 product specification, 2008. `http://mtron.net/Upload_Data/Spec/ASiC/MOBI/SATA/MSD-SATA3035_rev0.4.pdf`.

[6] W. Norcott. Iozone filesystem benchmark, 2006. `http://www.iozone.org/`.

In Proceedings of 12th International Conference on Extending Database Technology (EDBT2009), page588-599, March 23-26 2009,Saint-Petersburg,Russia

# A Sequential Indexing Scheme for Flash-Based Embedded Systems

Shaoyi Yin [*,**,***]
*INRIA Rocquencourt
78153 Le Chesnay - France
Fname.Lname@inria.fr

Philippe Pucheral [*,**]
**PRiSM Lab, Univ. of Versailles
78035 Versailles - France
Fname.Lname@prism.uvsq.fr

Xiaofeng Meng [***]
***Renmin Univ. of China
100872 Beijing – China
{yinshaoy, xfmeng}@ruc.edu.cn

## ABSTRACT

NAND Flash has become the most popular stable storage medium for embedded systems. As on-board storage capacity increases, the need for efficient indexing techniques arises. Such techniques are very challenging to design due to a combination of NAND Flash constraints (for example the block-erase-before-page-rewrite constraint and limited number of erase cycles) and embedded system constraints (for example tiny RAM and resource consumption predictability). Previous work adapted traditional indexing methods to cope with Flash constraints by deferring index updates using a log and batching them to decrease the number of rewrite operations in Flash memory. However, these methods were not designed with embedded system constraints in mind and do not address them. In this paper, we propose a new alternative for indexing Flash-resident data that specifically addresses the embedded context. This approach, called PBFilter, organizes the index structure in a purely sequential way. Key lookups are sped up thanks to two principles called Summarization and Partitioning. We instantiate these principles with data structures and algorithms based on Bloom Filters and show the effectiveness of this approach through a comprehensive performance study.

## 1. INTRODUCTION

Smart cards were equipped with kilobytes of EEPROM stable storage in the 90's and megabytes of NAND Flash in the 00's; mass-storage cards are coming soon that will link a microcontroller to gigabytes of NAND Flash memory [9]. All categories of smart objects (e.g., sensors, smart phones, cameras and mp4 players) benefit from the same storage capacity improvement thanks to high density NAND Flash. Smart objects are more versatile than ever and are now effective to manage medical, scholastic and other administrative folders, agendas, address books, photo galleries, transportation and purchase histories, etc. As storage capacity increases, the need for efficient indexing techniques arises. This motivates manufacturers of Flash modules and smart objects to integrate file management and even database techniques into their firmware.

Designing efficient indexing techniques for smart objects is very challenging, however, due to conflicting hardware constraints and design objectives.



On the one hand, although it is excellent in terms of shock resistance, density and read performance, NAND Flash exhibits specific hardware constraints. Read and write operations are done at a page granularity, as with traditional disks, but writes are more time and energy consuming than reads. In addition, a page cannot be rewritten without erasing the complete block containing it, which is a costly operation. Finally, a block wears out after about $10^5$ repeated write/erase cycles. As a result, updates are usually performed "out-of-place" entailing address translation and garbage collection overheads. The more RAM is devoted to buffering and caching and the lazier garbage collection is, the better the performance.

On the other hand, smart object manufacturers are facing new constraints in terms of energy consumption (to increase device autonomy/lifetime), microcontroller size (to increase tamper-resistance) and storage capacity (to save production costs on billion-scale markets)[2]. In this context, performance competes with energy, RAM and Flash memory consumption. Co-design rules are therefore essential to help manufacturers calibrate the hardware resources of a platform and select the appropriate data-management techniques to match the requirements of on-board data-centric applications.

State of the art Flash-based storage and indexing methods were not designed with embedded constraints in mind and poorly adapt to this context. Database storage models dedicated to NAND Flash have been proposed in [11, 13] without specifically addressing the management of hot spot data in terms of updates, like indexes. Other work addressed this issue by adapting B+Tree-like structures to NAND Flash [4, 16, 18]. While different in their implementation, these methods rely on a similar approach: delaying index updates using a log dedicated to the index, and batching them with a given frequency so as to group updates related to the same index node. We refer to these methods as *batch methods*. The benefit of batch methods is that they decrease write cost, which is considered the main problem with using Flash in the database context. However, all these methods maintain additional data structures in RAM to limit the negative impact of delayed updates on lookup cost. All these methods also perform "out-of-place" updates, reducing Flash memory usage and generating address translation and garbage collection overheads. Such indirect costs have proven high and unpredictable [16].

Rather than adapting traditional index structures to Flash memory, we believe that indexing methods must be completely rethought if we are to meet the requirements of the embedded context, namely:

- *Low_RAM*: accommodate as little RAM as possible
- *Low_Energy*: consume as little energy as possible
- *Low_Storage*: optimize the Flash memory usage
- *Adaptability*: make resource consumption adaptable to the

performance requirements of on-board applications

– *Predictability*: make performance and resource consumption fully predictable

Low_RAM emphasizes the specific role played by RAM in the embedded context. Due to its poor density, and as it competes with other hardware resources on the same silicon die, RAM is usually calibrated to its bare minimum [2]. Hence, the less RAM an indexing method consumes, the wider the range of devices that can be targeted. Low_Energy is also critical but concerns only autonomous devices. The objective of Low_Storage is to minimize not only the amount of Flash memory occupied by the index structure, but also, and above all, of Flash memory wasted by the obsolete data produced by index updates and leading to overprovisioning Flash memory. Adaptability conveys the idea that optimal performance is not the ultimate goal; rather optimality is reached when no resource is unduly consumed to get better performance than that strictly required by on-board applications. In other words, Adaptability means that Low_RAM, Low_Energy and Low_Storage must be considered in light of the applications' performance expectations. Finally, Predictability is a prerequisite to co-design.

In this paper, we propose a Flash-based indexing method, called PBFilter, specifically designed to answer these requirements. PBFilter organizes the index structure in a purely sequential way to minimize the need for buffering and caching and to avoid the unpredictable side effects incurred by "out-of-place" updates. But how to look up a given key in a sequential list with acceptable performance? We answer this question using two principles. Summarization consists of building an index summary used at lookup time to quickly determine the region of interest in the index. This introduces an interesting source of tuning between the compression ratio of the summary and its accuracy. Partitioning consists of vertically splitting the index list and/or its summary in such a way that only a subset of partitions need to be scanned at lookup time. This introduces a second trade-off between lookup performance and RAM consumption. The key idea behind Summarization and Partitioning is speeding up lookups without hurting sequential writes in Flash memory.

PBFilter gracefully accommodates files up to a few million tuples, a reasonable limit for embedded applications. PBFilter is optimized to support append-oriented files but deletion and updates can be supported without compromising the five requirements above.

The paper is organized as follows. Section 2 reviews the main characteristics of NAND Flash, studies the related work and introduces the metrics of interest for this study. Section 3 details the PBFilter indexing scheme. Section 4 presents an instantiation of the PBFilter scheme with partitioned Bloom Filter summaries. Section 5 presents a comprehensive performance study and Section 6 is the conclusion.

# 2. PROBLEM STATEMENT

## 2.1 NAND Flash Characteristics

Embedded Flash devices come today in various form factors such as compact flash cards, secure digital cards, smart cards and USB tokens. They share several common characteristics. A typical NAND Flash array is divided into blocks, in turn divided into pages (32-64 pages per block), and divided again into sectors (usually 4 sectors per page). Read and write operations usually happen at page granularity, but can also apply at sector granularity if required. A page is typically 512-2,048 bytes. A page can only be rewritten after erasing the entire block containing it (usually called the block-erase-before-rewrite constraint). Page write cost is higher than read, both in terms of execution time and energy consumption, and the block erase requirement makes writes even more expensive. A block wears out after about $10^5$ repeated write/erase cycles, requiring write load to be evenly spread out across the memory.

These hardware constraints make update management complex, although this complexity is slightly mitigated by the decomposition of a page into sectors. Sectors can be written independently (albeit sequentially) in the same page, allowing one write per sector before the block must be erased. To avoid erasing blocks too frequently, "out-of-place" updates are usually performed by using a Flash Translation Layer (FTL) which combines: (1) a translation mechanism relocating the pages and making their address invariant through indirection tables and (2) a garbage collection mechanism that erases blocks, either lazily (waiting for all the pages of the block to become obsolete) or eagerly (moving the active pages still present in the block before erasing it).

As extensively studied in [16], the execution time and energy consumption of read and write operations vary greatly among the Flash devices. The high discrepancies between the platforms are partly due to the raw chip characteristics and partly to the firmware managing the FTL which is usually proprietary and constitutes the primary source of performance unpredictability [16].

## 2.2 Related Work

Some work [11, 13] has adopted the idea of log-structured file systems (LFS) [17] to design or improve database storage models dedicated to NAND Flash, without proposing new index methods. For example, the primary objective of IPL [13] is to hide Flash peculiarities from the upper layers of the DBMS. Updates in Flash are delayed using a log stored in each physical block and an accurate version of each page is rebuilt at load time. Updates are physically applied to a page when the corresponding log region becomes full. While elegant, this general method is not well suited to managing hot spot data in terms of updates, like indexes, because of frequent log overflows.

Other work has specifically considered the indexing problem in NAND Flash. Hash-based and tree-based index families can be distinguished. So far, little attention has been paid to hash-based methods in Flash. This is probably because hashing only performs well when a large number of buckets can be used and when the RAM can accommodate one buffer per bucket, which is a rare situation in the targeted context. One exception is Microhash designed to speed up lookups in sensor devices [22]. However, this method is not general and only applies to sensed data varying within a small range (e.g., temperature).

Within the tree-based family, one work has also considered indexing sensed data [14]. This work proposes a tiny index called TINX based on a specific unbalanced binary tree. The performance demonstrated by the authors (e.g., 2,500 page reads to retrieve one record from 0.6 million records) disqualifies this method for large files. To the best of our knowledge, all other papers suggest adaptations of the well-known B+Tree structure. Regular B+Tree techniques built on top of FTL have been shown

to be poorly adapted to the characteristics of Flash memory [18]. Indeed, each time a new record is inserted into a file, its key must be added to a B+Tree leaf node, causing an out-of-place update of the corresponding Flash page. To avoid such updates, BFTL [18] constructs an "index unit" for each inserted primary key and organizes the index units as a kind of log. A large buffer is allocated in RAM to group the various insertions related to the same B+Tree node in the same log page. To maintain the logical view of the B+Tree, a node translation table built in RAM keeps, for each B+Tree node, the list of log pages which contain index units belonging to this node. In order to limit the size of these lists and therefore RAM consumption as well as lookup cost, each list is compacted when a certain threshold (e.g., 5 log pages in the list) is reached. At this time, logged updates are batched to refresh the physical image of the corresponding B+Tree node.

FlashDB [16] combines the best of Regular B+Tree and BFTL using a self-tuning principle linked to the query workload. JFFS3 proposes a slightly different way of optimizing B+Tree usage [4]. Key insertions are logged in a journal and are applied in the B+Tree in a batch mode. A journal index is maintained in RAM (recovered at boot time) so that a key lookup applies first to the journal index and then to the B+Tree.

In short, all B+Tree-based methods rely on the same principle: (1) delay index updates using a log and batch them with the purpose of grouping updates related to the same index node; (2) build a RAM index at boot time to speed up lookup of a key in the log; (3) commit log updates with a given commit frequency (CF) in order to limit log size. The differences between batch methods mainly include the way index nodes and log are materialized, which affect the way CF is managed.

In their attempt to decrease the number of writes, batch methods are in line with the Low_Energy requirement introduced in Section 1. By allowing trading reads, RAM and Flash memory usage for writes using CF, they also provide an answer to Adaptability. However, all batch methods fail in satisfying Low_RAM. Indeed, the higher the CF, the greater the RAM consumption. However, the primary objective of batch methods is to decrease the number of writes in Flash memory, leading to a higher CF. Section 5 will demonstrate that good write performance for batch methods requires RAM consumption incompatible with most embedded environments (in any case, not the objective they claim). Regarding Predictability, even if the number of writes is reduced, writes still generate out-of-place updates in Flash memory. This results in an indirect and unpredictable garbage collection cost linked to the strategy implemented in the underlying FTL [16]. Flash memory usage is also difficult to predict because it depends on the distribution of obsolete data in the pages occupied by the index.

## 2.3 Metrics of Interest

In light of the preceding discussion, more complete and accurate metrics appear necessary to help in assessing the adequacy of an indexing method for the embedded context. To this end, we propose the following metrics to capture the five requirements introduced in Section 1:

– *RAM consumption*: as already stated, RAM consumption is of utmost importance in the embedded context, since several devices (e.g., smart cards, sensors and smart tokens) are equipped with RAM measured in kilobytes [2]. This metric, denoted hereinafter *RAM*, comprises the buffers to read from and write to the Flash memory as well as the main memory data structures required by the indexing method.

– *Read/write cost*: this metric distinguishes between read cost *R* of executing a lookup and read cost *IR* and write cost *W* for inserting keys into the index. Depending on the objective, the metric can be execution time (wrt Adaptability) or energy consumption (wrt Low_Energy). To address both concerns, R, IR and W will be expressed in terms of number of operations. Note that this metric does not directly capture the Adaptability requirement, but rather tells whether the performance expected by on-board applications can be achieved.

– *Flash memory usage*: the objective is to capture the Flash memory usage, both in terms of space occupancy and effort to reclaim obsolete data. We distinguish between two values: *VP* is the total number of valid pages occupied by the index (i.e., pages containing at least one valid item); *OP* is the total number of pages containing only obsolete data and which can be reclaimed without copying data. Comparing these two values with the raw size of the index (total size of the valid items only) gives an indication of the quality of the Flash memory usage and the effort to reclaim stale space, independent of any FTL implementation.

– *Predictability*: as claimed in the introduction, performance and resource consumption predictability is a prerequisite for co-design. Predictability is mandatory in calibrating the RAM and Flash memory resources of a new hardware platform to the performance requirements of the targeted on-board applications. Another objective is to predict the limit (i.e., in terms of file size or response time) of an on-board application on existing hardware platforms. Finally, predictability is also required to build accurate query optimizers. To avoid making this metric fuzzy by reducing it to a single number, we express it qualitatively using two dimensions: (1) whether the indexing method is dependent on an underlying FTL or can bypass it, (2) whether the values measured for RAM, read/write cost and Flash memory usage can be accurately bounded independent of their absolute value and of the uncertainty introduced by the FTL, if any.

This paper aims to define a Flash-based indexing method that behaves satisfactorily in all of these metrics at once.

# 3. PBFILTER INDEXING SCHEME

As an alternative to the batch indexing methods, PBFilter performs index updates eagerly and makes this acceptable by organizing the complete database as a set of sequential data structures, as presented in Figure 1. The primary objective is to transform database updates into append operations so that writes are always produced sequentially, an optimal scenario for NAND Flash and buffering in RAM.

The database updating process is as follows. When a new record is inserted, it is added at the end of the record area (RA). Then, a new index entry composed by a couple <key, pt> is added at the end of the key area (KA), where key is the primary key of the inserted record and pt is the record physical address[1]. If a record

[1] Like all state of the art methods mentioned in Section 2, we concentrate the study on primary keys. The management of secondary keys is discussed in [21].

is deleted, its identifier (or its address) is inserted at the end of the delete area (DA) but no update is performed in RA nor KA. A record modification is implemented by a deletion (of the old record) followed by an insertion (of the new record value). To search for a record by its key, the lookup operation first scans KA, retrieves the required index entry if it exists, check that pt $\notin$DA and gets the record in RA. Assuming a buffering policy allocating one buffer in RAM per sequential data structure, this updating process never rewrites pages in Flash memory.

The benefits and drawbacks provided by this simple database organization are obvious with respect to the metrics introduced in Section 2. *RAM:* a single RAM buffer of one page is required per sequential structure (RA, KA and DA). The buffer size can even be reduced to a Flash sector in highly constrained environments. *Read/write cost:* a lower bound is reached in terms of reads/writes at insertion time (IR and W) since: (1) the minimum of information is actually written in Flash memory (the records to be inserted and their related index entries and no more), (2) new entries are inserted at the index tail without requiring any extra read to traverse the index. On the other hand, the lookup cost is dramatically high since R = ( $|$KA$|$/2 + $|$DA$|$ + 1) on the average, where $||$ denotes the page cardinality of a structure. *Flash memory usage:* besides DA, a lower bound is reached in terms of Flash usage, again because the information written is minimal and never updated. Hence, the number VP of valid pages containing the index equals the raw size of this index and the number OP of obsolete pages is null. Hence, the garbage collection cost is saved. *Predictability:* since data never moves and is never reclaimed, PBFilter can bypass the FTL address translation layer and garbage collection mechanism. RAM and Flash memory consumption is accurately bounded as discussed above. However, performance predictability is not totally achieved since the uncertainty on R is up to ( $|$KA$|$ - 1).

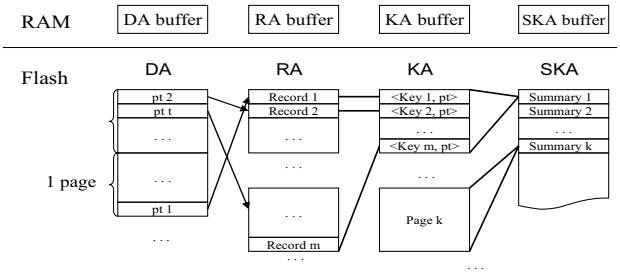



**Figure 1. Database organization**

The objective is now to decrease the lookup cost R to an acceptable value with a minimal degradation of the benefits listed above. Summarization and Partitioning are two principles introduced to reach this goal.

*Summarization* refers to any method which can condense sequentially the information present in KA. Let us consider an algorithm that condenses each KA page into a summary record. Summary records can be sequentially inserted into a new structure called SKA through a new RAM buffer of one page (or sector) size. Then, lookups do a first sequential scan of SKA and a KA page is accessed for every match in SKA in order to retrieve the requested key, if it exists. Summarization introduces an interesting trade-off between the compression factor *c* (c= $|$KA$|$/$|$SKA$|$) and the fuzziness factor *f* (i.e., probability of false positives) of the summary, the former decreasing the I/O required to traverse SKA and the latter decreasing the I/O required to access KA. The net effect of summarization is reducing R to ( $|$KA$|$/2c + f$|$KA$||$/2) on the average, where $||$ $||$ denotes the element cardinality of a structure. The positive impact on R can be very high for favorable values of *c* and *f*. The negative impact on the RAM consumption is limited to a single new buffer in RAM. The negative impact on the write cost and Flash memory usage is linear with $|$SKA$|$ and then depends on *c*. Different algorithms can be considered as candidate "condensers", with the objective to reach the higher *c* with the lower *f*, if only they respect the following property: *summaries must allow membership tests with no false negatives*.

The idea behind *Partitioning* is to vertically split a sequential structure into *p* partitions so that only a subset of partitions has to be scanned at lookup time. Partitioning can apply to KA, meaning that the encoding of keys is organized in such a way that lookups do not need to consider the complete key value to evaluate a predicate. Partitioning can also apply to SKA if the encoding of summaries is such that the membership test can be done without considering the complete summary value. The larger *p*, the higher the partitioning benefit and the better the impact on the read cost and on Predictability. On the other hand, the larger *p*, the higher the RAM consumption (*p* buffers) or the higher the number of writes into the partitions (less than *p* buffers) with the bad consequence of reintroducing page moves and garbage collection. Again, different partitioning strategies can be considered with the following requirement: *to increase the number of partitions with neither significant increase of RAM consumption nor need for garbage collection.*

# 4. PBFILTER INSTANTIATION

## 4.1 Bloom Filter Summaries

The Bloom Filter data structure has been designed for representing a set of elements in a compact way while allowing membership queries with a low rate of false positives and no false negative [5]. Hence, it presents all the characteristics required for a condenser.

A Bloom filter represents a set $A=\{a_1, a_2, \ldots a_n\}$ of *n* elements by a vector *v* of *m* bits, initially all set to 0. The Bloom filter uses *k* independent hash functions, $h_1$, $h_2$, … $h_k$, each producing an integer in the range [1,m]. For each element $a_i \in A$, the bits at positions $h_1(a_i)$, $h_2(a_i)$, ..., $h_k(a_i)$ in *v* are set to 1. Given a query for element $a_j$, all bits at positions $h_1(a_j)$, $h_2(a_j)$, ..., $h_k(a_j)$ are checked. If any of them is 0, then $a_j$ cannot be in *A*. Otherwise we conjecture that $a_j$ is in *A* although there is a certain probability that we are wrong. The parameters *k* and *m* can be tuned to make the probability of false positives extremely low [8].

**Table 1. False positive rate under various m/n and k**

| m/n | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 |
|---|---|---|---|---|---|---|
| **8** | .0306 | .024 | .0217 | .0216 | .0229 | |
| **12** | .0108 | .0065 | .0046 | .0037 | .0033 | .0031 |
| **16** | .005 | .0024 | .0014 | .0009 | .0007 | .0006 |

This probability, called the *false positive rate* and denoted by *f* in the sequel, can be calculated easily assuming the k hash functions are random and independent. After all the elements of *A* are hashed into the Bloom filter, the probability that a specific bit is still 0 is $\left(1-1/m\right)^{kn} \approx e^{-kn/m}$. The probability of a false positive is then $\left(1-\left(1-1/m\right)^{kn}\right)^k \approx \left(1-e^{-kn/m}\right)^k = \left(1-p\right)^k$ for $p=e^{-kn/m}$. The salient feature of Bloom filters is that three performance metrics can be traded off against one another: computation time (linked to the

number k), space occupancy (linked to the number m), and false positive rate *f*. Table 1 illustrates these trade-offs for some values of k and m. This table shows that a small increase of m may allow a dramatic benefit for f if the optimal value of k is selected. We consider that k is not a limiting factor in our context, since methods exist to obtain k hash values by calling only three times the hash function, while giving the same accuracy as by computing k independent hash functions [7].

Bloom filters can be used as a condenser algorithm in PBFilter as follows. For each KA page, a Bloom filter summary is built by applying *k* hash functions to each index key present in that page. This computation is performed when the KA page is full, just before the RAM buffer containing it is flushed to the Flash memory. The computed Bloom filter summary is stored in the RAM buffer allocated to SKA. In turn, the SKA buffer is flushed to the Flash memory when full. At lookup time, the searched key $a_i$ is hashed with the *k* hash functions. Then, SKA is scanned to get the first Bloom filter summary having all bits at positions $h_1(a_i)$, $h_2(a_i)$, ..., $h_k(a_i)$ set to 1. The corresponding page of KA is directly accessed and the probability that it contains the expected index entry ($a_i$,pt) is (1-f). Otherwise, the scan continues in SKA. The last step is to check that pt $\notin$DA before accessing the record in RA.

## 4.2 Dynamic Partitioning

Despite the benefits of summarization, the lookup performance remains linked to the size of SKA (on the average, half of SKA needs to be scanned). The lookup performance can be improved by applying the partitioning principle suggested in section 3. Each Bloom filter is vertically split into *p* partitions (with $p \leq m$), so that the bits in the range [1 .. m/p] belong to the first partition, the bits in the range [((i-1)*m/p + 1) .. (i*m/p)] belong to the i$^{th}$ partition, etc. When the SKA buffer is full, it is flushed into *p* Flash pages, one per partition. By doing so, each partition is physically stored in a separate set of Flash pages. When doing a lookup for key $a_i$, instead of reading all pages of SKA, we need to get only the SKA pages corresponding to the partitions containing the bits at positions $h_1(a_i)$, $h_2(a_i)$, ..., $h_k(a_i)$. The benefit is a cost reduction of the lookup by a factor p/k. The larger *p*, the higher the partitioning benefit for lookups but also the greater the RAM consumption (p more buffers) or the greater the number of writes (because page fragments have to be flushed in the partitions in Flash memory instead of full pages) and then the need for garbage collection (because of multiple writes in the same page of Flash).

We propose below a partitioning mechanism which exhibits the nice property of supporting a dynamic increase of p with no impact on the RAM consumption and no need for a real garbage collection (as discussed at the end of the section, obsolete data is naturally grouped in the same blocks which can be erased as a whole at low cost). This dynamic partitioning mechanism comes at the price of introducing a few reads and extra writes at insertion time. The proposed mechanism relies on: (1) the usage of a fixed amount of Flash memory as a persistent buffer to organize a stepwise increase of p and (2) the fact that a Flash page is divided into *s* sectors (usually *s*=4) which can be written independently. The former point gives the opportunity to reclaim the Flash buffer at each step in its integrality (i.e., without garbage collection). The latter point allows *s* writes into the same Flash page before requiring copying the page elsewhere.

Figure 2 illustrates the proposed partitioning mechanism. The size of the SKA buffer in RAM is set to the size of a Flash page and the buffer is logically split into *s* sectors. The number of initial partitions, denoted next by L1 partitions, is set to *s* and one page of Flash is initially allocated to each L1 partition. The first time the SKA buffer in RAM becomes full (step 1), each sector $s_i$ (with $1 \leq i \leq s$) of this buffer is flushed in the first sector of the page allocated to the i$^{th}$ L1 partition. The second flush of the SKA buffer will fill in the second sector of these same pages and so forth until the first page of each L1 partition becomes full (i.e., after *s* flushes of the SKA buffer). A second Flash page is then allocated to each L1 partition and the same process is repeated until each partition contains *s* pages (i.e., after $s^2$ flushes of the SKA buffer). Each L1 partition contains 1/s part of all Bloom filters (e.g., the i$^{th}$ L1 partition contains the bits in the range [((i-1)*m/s + 1) .. (i*m/s)]).

At this time (step 2), the *s* L1 partitions of *s* pages each are reorganized (read back and rewritten) to form $s^2$ L2 partitions of one page each. Then, each L2 partition contains $1/s^2$ part of all Bloom filters. As illustrated in Figure 2, each L2 partition is formed by projecting the bits of the L1 partition it stemmed from on the requested range, *s* times finer (e.g., the i$^{th}$ L2 partition contains the bits in the range [((i-1)*m/$s^2$ + 1) .. (i*m/$s^2$)]).

After another $s^2$ SKA buffer flushes (step 3), *s* new L1 partitions have been built again and are reorganized with the $s^2$ L2 partitions to form ($s^2$+$s^2$) L3 partitions of one page each and so forth. The limit is *p*=*m* after which there is no benefit to partition further since each bit of bloom filter is in a separate partition. After this limit, the size of partitions grows but the number of partitions remains constant (i.e., equal to m).

In the example presented in Figure 2, where s=4, the number of partitions grows in an approximately linear way (4, 16, 32…)[2]. Assuming for illustration purpose Flash pages of 2KB, bloom filters of size m=2048 bits in SKA and <key,pt> of size 8 bytes in KA, each page of L3 partitions gathers 1/32 part of 256 bloom filters summarizing themselves 65536 keys. Scanning one complete partition in SKA costs reading the corresponding page in L3 plus 1 to s pages in L1.

More precisely, the benefit of partitioning dynamically SKA is as follows. A lookup needs to consider only *k* Li partitions of one page each (assuming the limit p=m has not been reached and Li partitions are the last produced) plus min (*k ,s)* L1 partitions, the size of which vary from 1 to *s* pages. This leads to an average cost of (*k* + min (k, s) * *s*/2). This cost is both low and independent of the file size while p≤m.

The RAM consumption remains unchanged, the size of the SKA buffer being one page (note that extending it to *s* pages would save the first iteration). The impact on IR and W (Read and write cost at insertion time) is an extra cost of about $\sum_i 2^{\lceil \log_2 i \rceil} * s^2$ reads and writes (see the cost model for details). This extra cost may be considered important but is strongly mitigated by the fact that it applies to SKA where each page condenses $B_p$/d records, where $B_p$ is the size of a Flash page in bits ($B_p$ /d is likely to be

[2] In practice, it does not grow exactly linearly because the bloom filter cannot be equally divided into an arbitrary number of partitions. For the same reason, the bloom filter size is always a power of 2, so one bloom filter may summarize more than 1 (less than 2) KA pages. The impact of these implementation details have been taken into account in the cost model in Section 5, and the extra cost has proven low.

greater than 1000). Section 5 will show that this extra cost is actually low compared to existing indexing techniques. Section 5 will also show the low impact of partitioning on the Flash usage for the same reason, that is the high compression ratio obtained by Bloom filters making SKA small with respect to KA.

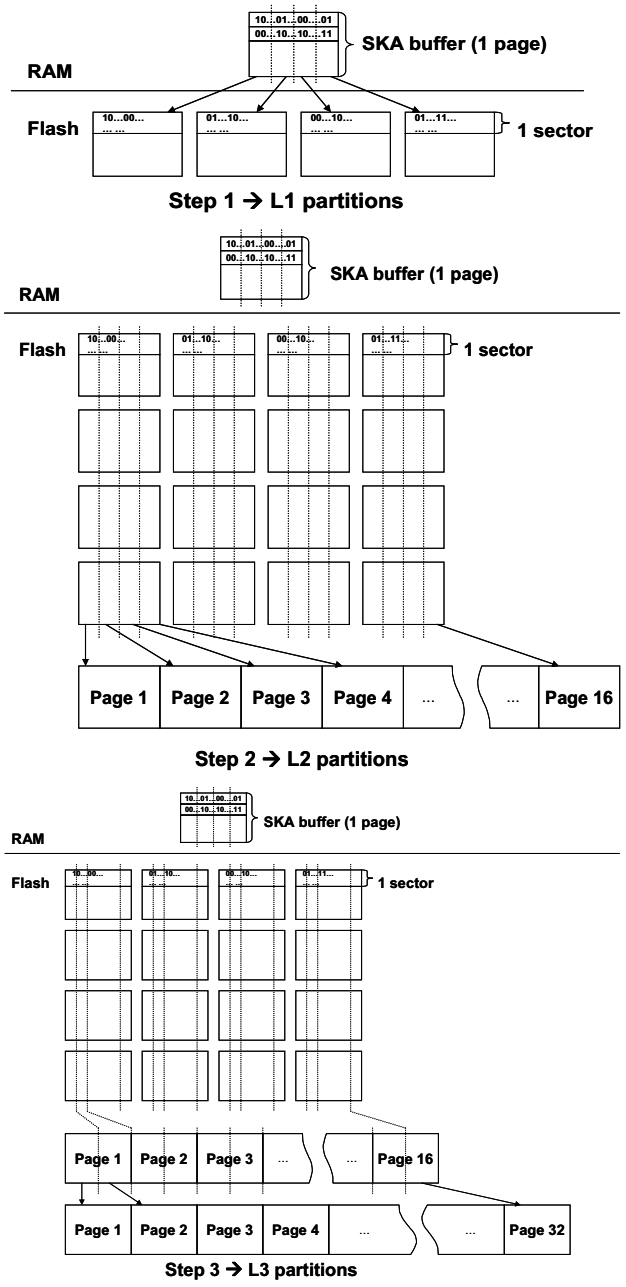



**Figure 2. Dynamic partitioning**

At the end of each step i, and after Li partitions have been built, the Flash buffer hosting L1 partitions and the pages occupied by $L_{i-1}$ partitions can be reclaimed. Reclaiming a set of obsolete pages stored in the same block is far more efficient than collecting garbage crumbs spread over different pages in different blocks. The distinction between garbage reclamation and garbage collection is actually important. Garbage collection means that active pages present in a block elected for erasure must be moved first to another block. In addition, if at least one item is active in a page, the complete page remains active. In methods like BFTL, active index units can be spread over a large number of pages in an uncontrolled manner. This generates a worst situation where many pages remain active while they contain few active index units and these pages must be often moved by the garbage collector. PBFilter never generates such situations. The size of the Flash buffer and of the $L_i$ partitions is a multiple of $s^2$ pages and these pages are always reclaimed together. Blocks are simply split in areas of $s^2$ pages and a block is erased when all its areas are obsolete.

## 4.3 Hash then Partition

As stated above, the benefit of partitioning is a cost reduction of the lookup by a factor $p/k$. The question is whether this factor can still be improved. When doing a lookup for key $a_i$ in the current solution, the probability that positions $h_1(a_i)$, $h_2(a_i)$, ..., $h_k(a_i)$ fall into a number of partitions less than $k$ is low, explaining the rough estimate of the cost reduction by the factor $p/k$. This situation could be improved by adding a hashing step before building the Bloom filters. Each Bloom filter is split into $q$ buckets by a hash function $h_0$ independent of $h_1$, $h_2$, ..., $h_k$,. Each time a new key is inserted in KA, $h_0$ is applied first to determine the right bucket, then $h_1$, $h_2$, ..., $h_k$ are computed to set the corresponding bits in the selected bucket. This process is similar as building $q$ small Bloom filters for each KA page. The experiments we conducted led to the conclusion that $q$ must remain low to avoid any negative impact on the false positive rate. Thus, we select $q=s$ (with $s$ usually equals to 4). The benefit of this initial hashing is guaranteeing that the $k$ bits of interest for a lookup always fall into the same L1 partition, leading to an average cost of ($k$ + s/2) for scanning SKA.

## 4.4 An Illustration of Hashed PBFilter

Now let us illustrate the key insertion and lookup processes of hashed PBFilter through an example (Figure 3). As pointed above, we set q=s=4 and m=2048, while supposing the size of <key, pt> is 8 bytes and the size of a page is 2048 bytes. To simplify the calculation, we use only 3 hash functions to build the bloom filters, denoted by $h_1$(key), $h_2$(key) and $h_3$(key). The hash function used in the pre-hashing step is denoted by $h_0$(key).

When the first key key1 is inserted, the hash bucket number is determined first by using $h_0$, and then $h_1$, $h_2$ and $h_3$ are computed. Suppose that: $h_0$(key1) = 0, $h_1$(key1) = 1, $h_2$(key1) = 201, and $h_3$(key1) = 301. Accordingly, the 1st, 201st and 301st bits in bucket 0 (the first 512 bits) of the first bloom filter bloom1 are set to 1 (Status 1 in Figure 3).

After inserting 2048 keys, the SKA buffer is full with 8 bloom filters (each bloom filter summarizes one KA page which contains 256 <key, pt> entries), so the bloom filters are partitioned and flushed into the L1 partitions: the first 512 bits (bucket 0) of each bloom filter are written into the first sector of page $P_{01}$, the second 512 bits (bucket 1) of each bloom filter are written into the first sector of page $P_{11}$, and so on (Status 2).

After inserting 32768 keys, the L1 partition pages are full, so the bloom filters are repartitioned into smaller pieces forming L2: the first 128 bits of all 128 bloom filters are written into the first L2 partition P1, the second 128 bits of all 128 bloom filters are written into the second L2 partition P2, and so forth (Status 3).

After inserting 65536 keys, the new L1 partitions are full again,

so the bloom filters are repartitioned once more into ever smaller pieces forming L3: the first 64 bits of all 256 bloom filters (128 from the L2 partitions and 128 from the L1 partitions) are written into the first L3 partition P1', the second 64 bits of all 256 bloom filters are written into the second L3 partition P2', and so forth (Status 4).

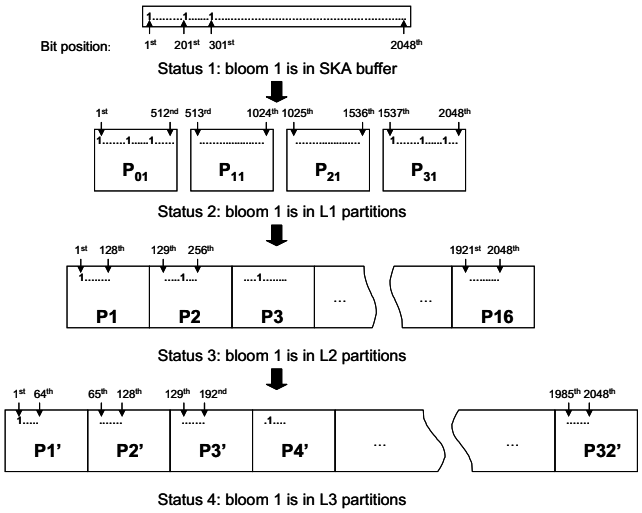



**Figure 3. Storage status changing of a bloom filter**

Now the bloom filters have been partitioned three times and have produced 32 L3 partitions each containing 64 bits of each bloom filter. Note that each of the L3 partitions still belongs to a single hash bucket set by $h_0$: the first 8 pages belong to bucket 0, the second 8 pages belong to bucket 1, and so on.

At this time, the process for looking up key1 is as follows. First, compute the hash functions to locate the required bit positions: $h_0(key1) = 0$, $h_1(key1) = 1$, $h_2(key1) = 201$, and $h_3(key1) = 301$, which means that, the 1st, 201st, and 301st bit positions of bucket 0 should be checked. In the L3 partitions, the three bit positions are stored in P1', P4' and P5' respectively, so only these pages have to be loaded into RAM. In this case, key1 will be found by only checking these pages. In other cases, if the searched key is not found in the L3 partitions, the current L1 partitions must be checked also: instead of scanning all the L1 partitions, only the pages in the corresponding bucket need to be checked (at most s pages), for example, if $h_0(key) = 1$, only $P_{11}$, $P_{12}$, $P_{13}$ and $P_{14}$ are scanned if they are not empty.

## 4.5 Deletes and Updates

PBFilter has been preliminary designed to tackle applications where insertions are more frequent and critical than deletes or updates. This characteristic is common in the embedded context. For instance, deletes and updates are proscribed in medical folders and many other administrative folders for legal reasons. Random deletes and updates are also meaningless in several applications dealing with historical personal data, audit data or sensed data. Note that cleaning history to save local space differs from deleting/updating randomly elements. While the latter impose to deal with a large DA area, the former can be easily supported. Indeed, cleaning history generates bulk and sequential deletes of old data. A simple low watermark mechanism can isolate the data related in RA, KA and SKA to be reclaimed together.

Let us now consider a large number of random deletes and updates enlarging DA and thereby decreasing the lookup performance. The solution to tackle this situation is to index DA itself using the same strategy, that is building bloom filters on the content of DA pages and partitioning them. The lookup cost being non linear with the file size, there is a great benefit to keep a single DA area for the complete database rather than one per file. This will bound the extra consumption of RAM to s more buffers for the whole architecture. The extra cost in Flash memory is again strongly limited by the high compression ratio of bloom filters. As section 5 will show, the lookup cost is kept low, though roughly multiplied by a factor 2 with high update/delete rate.

# 5. PERFORMANCE EVALUATION

The first objective of this section is to study how traditional B+Tree, batch methods and PBFilter perform in the embedded context. To allow a fair comparison between the approaches and isolate the FTL cost indirectly paid by batch methods and B+Tree, we introduce a precise analytical cost model. The results are more easily interpretable than real measurements performed on an opaque firmware. These results show that, while B+Tree and batch methods can slightly outperform PBFilter in some situations, PBFilter is the sole method to meet all requirements of an embedded context. Then, this section discusses how PBFilter can be tuned in a co-design perspective. Finally, preliminary performance measurements conducted on a specific hardware platform are given for illustrative purpose.

## 5.1 Analytical Performance Comparison

### *5.1.1 Indexing Methods under Test*

As stated above, the objective is not to perform an exhaustive comparison of all Flash-based indexing methods, considering that only PBFilter has been specifically designed to cope with embedded constraints. The comparison will then concentrate on opposite approaches (traditional, batch, Summarization & Partitioning), rather than focusing on variations. Regular B+Tree running on top of FTL, denoted by BTree hereafter, is considered as a good representative of traditional disk-based indexing methods running on Flash memory with no adaptation. BFTL [18] is selected as a good, and probably best known, representative of batch methods. To better understand the impact of (not) bounding the log size in batch methods, we consider two variations of BFTL: BFTL1 with no compaction of the node translation table and BFTL2 with the periodic compaction of the node translation table suggested in [18]. The Bloom filter instantiation of PBFilter, denoted by PBF hereafter, is so far the unique representative of Summarization & Partitioning methods.

The performance metrics used to compare these methods are those introduced in Section 2.3, namely: RAM (RAM consumption in KB), R (average number of page reads to lookup a key), IR (total number of page reads to insert N records), W (total number of page writes to insert N records), VP (total number of valid Flash pages) and OP (total number of obsolete Flash pages).

### *5.1.2 Parameters and Formulas*

The parameters and constants used in the analytical model are listed in Table 2 and Table 3, respectively.

Table 4 contains basic formulas used in the cost model and the cost model itself is presented in Table 5. To make the formulas as precise as possible, we use Yao's Formula [20] when necessary.

- ***Yao's Formula**: Given n records grouped into m blocks (1<m≤n), each contains n/m records. If k records (k≤n-n/m) are randomly selected, the expected number of blocks hit is:*

$$Yao(n,m,k) = m \times \left[1 - \prod_{i=1}^{k} \frac{nd - i + 1}{n - i + 1}\right], where, d = 1 - 1/m$$

**Table 2. Parameters for the analytical model**

| Param | Signification |
|---|---|
| N | Total number of inserted records |
| Sk | Size of the primary key (in bytes) |
| B | Number of buffer pages in RAM |
| C | Maximum size of a node translation table list in BFTL2 |
| d | Value of m/n in Bloom filter (see Table 1 for examples) |
| k | Number of hash functions used by Bloom filter |

**Table 3. Constants for the analytical model**

| Constants | Signification |
|---|---|
| Sr=4 (bytes) | Size of a physical pointer |
| fb=0.69 | Average fill factor of B+Tree [19] |
| β =2 | Expansion factor of flash storage caused by the buffering policy in BFTL [18] |
| Sp=2048(bytes) | Size of a Flash page |

### *5.1.3 Performance Comparison*

We first compare the four methods under test on each metric with the following parameter setting: N=1 million records, Sk=12, C=5 for BFTL2 (a medium value wrt [18]), and B=7, d=16, k=7 for PBF (which correspond also to medium values). The results are shown in Figures 4(a) to 4(e).

BTree exhibits an excellent lookup performance and consumes little RAM but the price to pay is an extremely high write cost and consequently a very high number of obsolete pages produced (OP). Hence, either the Flash memory usage will be very poor or the garbage collection cost very high. Considering that writes are more time and energy consuming than reads, BTree adapt poorly Flash storage whatever the environment (embedded or not) [3].

BFTL has been primarily designed to decrease the write cost incurred by BTree and Figures 4(c) and 4(e) show the benefit. BFTL1 exhibits the highest benefit in terms of writes and Flash memory usage. However, it incurs an unacceptable lookup cost and RAM consumption given that the node translation lists are not bounded. The IR cost is also very high since each insertion incurs a traversal of the tree. By bounding the size of the node translation lists, BFTL2 exhibits a much better behaviour for metrics R, IR and RAM (though RAM remains high wrt embedded constraints) at the expense of a higher number of writes (to refresh the index nodes) and a higher Flash memory consumption (BFTL mixing valid and obsolete data in the same Flash pages). To better capture the influence of the log size in batch methods, we vary parameter C in Figure 4(f), keeping the preceding values for the other parameters, and study the influence on metrics W, VP and OP. As expected, W and OP (which equals to W) decrease as C increases since the tree reorganizations become less frequent (VP stays equal to 0), up to reach the same values as BFTL1 (equivalent to an infinite C). Conversely, R and RAM grows linearly with C (e.g., R=105 and RAM=2728 when C=30, as shown by formula in Table 5). Trading R and RAM for W and OP is common to all batch methods but there is no trade-off which exhibits acceptable values for RAM, W and OP altogether to meet embedded constraints (Low_RAM, Low_Energy, Low_Storage). Even FlashDB [16] which dynamically takes the best of BTree and BFTL according to the query workload cannot solve the equation.

Though slightly less efficient for lookups than BTree and even BFTL2 when the update/delete rate is high (figure 4(a) shows that metric R for PBF ranges from 10 without update up to 22 with 100% updates)[4], PBF is proved to be the sole indexing method to meet all embedded constraints at once. In this setting, PBF exhibits excellent behaviour in terms of IR, W, VP and OP while the RAM consumption is kept very low. Note that if the RAM constraint is extremely high, the granularity of the buffer could be one sector, as explained in Section 3 and 4.2, leading to a total RAM consumption of 3.5KB[5].

The point is to see whether the same conclusion can be drawn in other settings, and primarily for larger files where sequential methods like PBF are likely to face new difficulties. Figures 4(g) to 4(i) analyse the scalability of BFTL and PBF on R, W and RAM varying N from 1 million up to 7 million records, keeping the initial values for the other parameters (Figures 4(g) and 4(i) use a logarithmic scale for readability). BTree is not further considered considering its dramatically bad behaviour in terms of W and OP.

BFTL2 scales better than PBF in terms of R and even outperforms PBF for N greater than 2.5 million records (though R performance of PBF remains acceptable). However, BFTL2 scales very badly in terms of W. BFTL1 scales much better in terms of W but exhibits unacceptable performance for R and RAM. Unfortunately, PBF scales also badly in terms of W. Beyond this comparison which shows that efficient Flash-based method for indexing very large files still need to be invented, let us see if the scalability of PBF can be improved to cover the requirements of most embedded applications. Actually, the cost of repartitioning becomes dominant for large N and repartitioning occurs at every Flash buffer overflow. A solution for large files is then to increase the size of the Flash buffer hosting the L1 partitions under construction. The comparison between PBF1 and PBF2 on Figure 4(i) shows the benefit of increasing the Flash buffer from 16 pages for PBF1 (that is 4 L1 partitions of 4 pages each) to 64 pages for PBF2 (16 L1 partitions of 4 pages each). Such increase does not impact metric R since the number of reads in L1 partitions does not depend on the number of partitions but of their size (which we keep constant). The RAM impact sums up to 12 more buffers for SKA, but this number can be reduced to only 3 pages by organizing the buffers by sectors. Hence, PBF can accommodate gracefully rather large embedded files (a few millions tuples) assuming the RAM constraint is slightly relaxed (a co-design choice).

[3] The same conclusion can be drawn for other traditional indexing techniques applied to Flash with no adaptation. E.g., for hashing, either the number of buckets is kept very small so that a RAM buffer can be allocated to each and R is very bad (because of the bucket size) or the number of buckets is very high and RAM is very high too.

[4] Note that the R cost for BFTL and BTree neglects the FTL address translation cost which may be high (usually a factor 2 to 3).

[5] For the sake of simplicity, the formulas of the cost model consider the granularity of buffers to be one page.

**Table4. Basic formulas of the analytical model**

| Vars | Annotations | Expressions |
|---|---|---|
| Common formulas | | |
| M | Number of index units (IUs) in each page [Note1] | $\lfloor Sp/(Sk+5*Sr) \rfloor$ for BFTL1&2, $\lfloor Sp/(Sk+Sr) \rfloor$ for others |
| Formulas specific to Tree-based methods (Btree, BFTL1, BFTL2) | | |
| ht | Height of B+Tree | $\lceil \log_{fb*M+1} N \rceil$ |
| Nn | Total number of B+Tree nodes after N insertions | $\sum_{i=1}^{ht} \lceil N/((fb*M)(fb*M+1)^{i-1}) \rceil$ |
| Ns | Number of splits after N insertions | Nn-ht |
| L | Average number of buffer chunks that the IUs from the same B+Tree node are distributed to [Note2] | Yao(N, N/(fb*M*B), fb*M) for BTree,<br>Yao(N, β*N/(M*B), fb*M) for BFTL1&2, |
| α | Number of index units of a logical node stored in a same physical page [Note3] | fb*M/L |
| Nc | Number of compactions for each node in BFTL2 | $\lfloor (L-1)/(C-1) \rfloor$ |
| Formulas specific to PBF | | |
| $N_{KA}$ | Total number of pages in KA | $\lceil N/M \rceil$ |
| Sb | Size of a bloom filter (bits) | $2^{\lceil \log_2(M*d) \rceil}$ |
| Mb | Number of bloom filters in a page | $\lfloor Sp*8/Sb \rfloor$ |
| M1 | Number of <key, pinter> pairs contained by one bloom filter | $\lfloor Sb/d \rfloor$ |
| Nr | Total number of partition reorganizations [Note4] | $\lfloor \lfloor N_{KA}/Mb \rfloor /(L1*s) \rfloor$ |
| Pf | Number of last final partitions | $L1*s*2^{\lceil \log_2(Nr\%(Sb/L1/s)) \rceil}$, if Nr>0, else Pf =0 |
| $N_{FB}$ | Number of pages occupied by the final valid blooms | $\lfloor N/(Sp*8*M1) \rfloor * Sb + Pf + \lfloor (\lfloor N_{KA}/Mb \rfloor \bmod (L1*s))/s \rfloor * s$ |
| $N_E$ | Total number of pages which can be erased | $\lfloor N/(Sp*8*M1) \rfloor * \left\lfloor \sum_{i=2}^{Sb/(L1*s)} (2^{\lceil \log_2(i-1) \rceil} * L1*s) \right\rfloor + \left\lfloor \sum_{i=2}^{Nr\%(Sb/L1/s)} (2^{\lceil \log_2(i-1) \rceil} * L1*s) \right\rfloor + Nr*L1*s$ |

[Note1] In BFTL, there are five pointers in each Index Unit (data_ptr, parent_node, identifier, left_ptr, right_ptr), explaining factor 5.
[Note2] Yao's formula is used here to compute how many buffer chunks (1 buffer chunk containing B pages) that fb*M records are distributed to, which is the average length of the lists in node translation table for BFTL1.
[Note3] The IUs from the same logical node are stored in different physical pages, so we divide the total number of IUs (fb*M) by the total number of physical pages to get the average number of IUs stored in the same physical page.
[Note4] L1 denotes the number of pages in each initial L1 partition and L1*s is the size of the Flash buffer used to manage them.

**Table 5. Final formulas of the analytical model**

| Metrics\Methods | | BTree | BFTL1 | BFTL2 | PBF |
|---|---|---|---|---|---|
| **R** | [Note1] | ht | (ht-1)*L+L/2 | (ht-1)*C+C/2 | R1+R2+$\lceil f*N_{KA}*\lceil M/M \rceil /2 \rceil$+ R3 |
| **W** | [Note2] | N/α+2Ns | β *N/M + β *Ns/2 | W1 | $N_{KA} + N_{FB} + N_E$ |
| **IR** | [Note3] | IR1 | IR2 | IR3 | $N_E$ |
| **RAM** | [Note4] | B*Sp/1024 | (Nn*L*Sr+B*Sp) /1024 | (Nn*C* Sr+B*Sp) /1024 | B*Sp/1024 |
| **VP** | [Note5] | Nn | W | W | $N_{KA} + N_{FB}$ |
| **OP** | [Note5] | W-Nn | 0 | 0 | $N_E$ |

Where, IR1 = Ns*(fb*M/2) + $\sum_{i=1}^{ht-2} i*((fb*M)((fb*M+1)^i-(fb*M+1)^{i-1})+1)+(ht-1)*(N-(fb*M)(fb*M+1)^{ht-2})$

IR2 = Ns*(fb*M/2) + $\sum_{i=1}^{ht-2} i*L*((fb*M)((fb*M+1)^i-(fb*M+1)^{i-1})+1)+(ht-1)*L*(N-(fb*M)(fb*M+1)^{ht-2})$

IR3 = Ns*(fb*M/2) + $\sum_{i=1}^{ht-2} i*C*((fb*M)((fb*M+1)^i-(fb*M+1)^{i-1})+1)+(ht-1)*C*(N-(fb*M)(fb*M+1)^{ht-2})$

W1= β *N/M + β *Ns/2 + β *Nn* $\sum_{i=0}^{Nc-1}(\alpha C+i*(\alpha C-1))$ /M

R1 = $\lfloor (N_{FB}-Pf)/N_{FB} \rfloor * \lceil ((N_{FB}/L1) \bmod s)/2 \rceil$ , R2= $\lceil Yao(Sb/s, Pf/s, k) \rceil$ ,R3= $\lceil N/(Sp*8*M2)/2 \rceil * Yao(Sb/s, Sb/s, k)$

[Note1] For BTree, we did not consider the additional I/Os of going through the FTL indirection table. For BFTL1&2, loading a node requires traversing, in the node translation table, the whole list of IUs belonging to this node and accessing each in Flash. In PBF, the read cost comprises: the lookup in the final and initial partitions and the cost to access (KA), including the overhead caused by false positives.
[Note2] For BTree, the write cost integrates the copy of the whole page for every key insertion (2 times more for splits). BFTL methods also need data copy when doing splits and the write cost of BFTL2 integrates the cost of periodic reorganizations. The write cost for PBF is self-explanatory.
[Note3] For Tree-based methods, the IR cost integrates the cost to traverse the tree up to the target leaf and the cost to read the nodes to be split. For PBF, it integrates the cost to read the partitions to be merged at each iteration.
[Note4] RAM comprises the size of the data structures maintained in RAM plus the size of the buffers required to read/write the data in Flash.
[Note5] VP+OP is the total number of pages occupied by both valid and stale index units. In BFTL1&2, OP=0 simply because stale data are mixed with valid data. By contrast, stale data remain grouped in BTree and PBF. In BTree, this good property comes at a high cost in terms of OP.

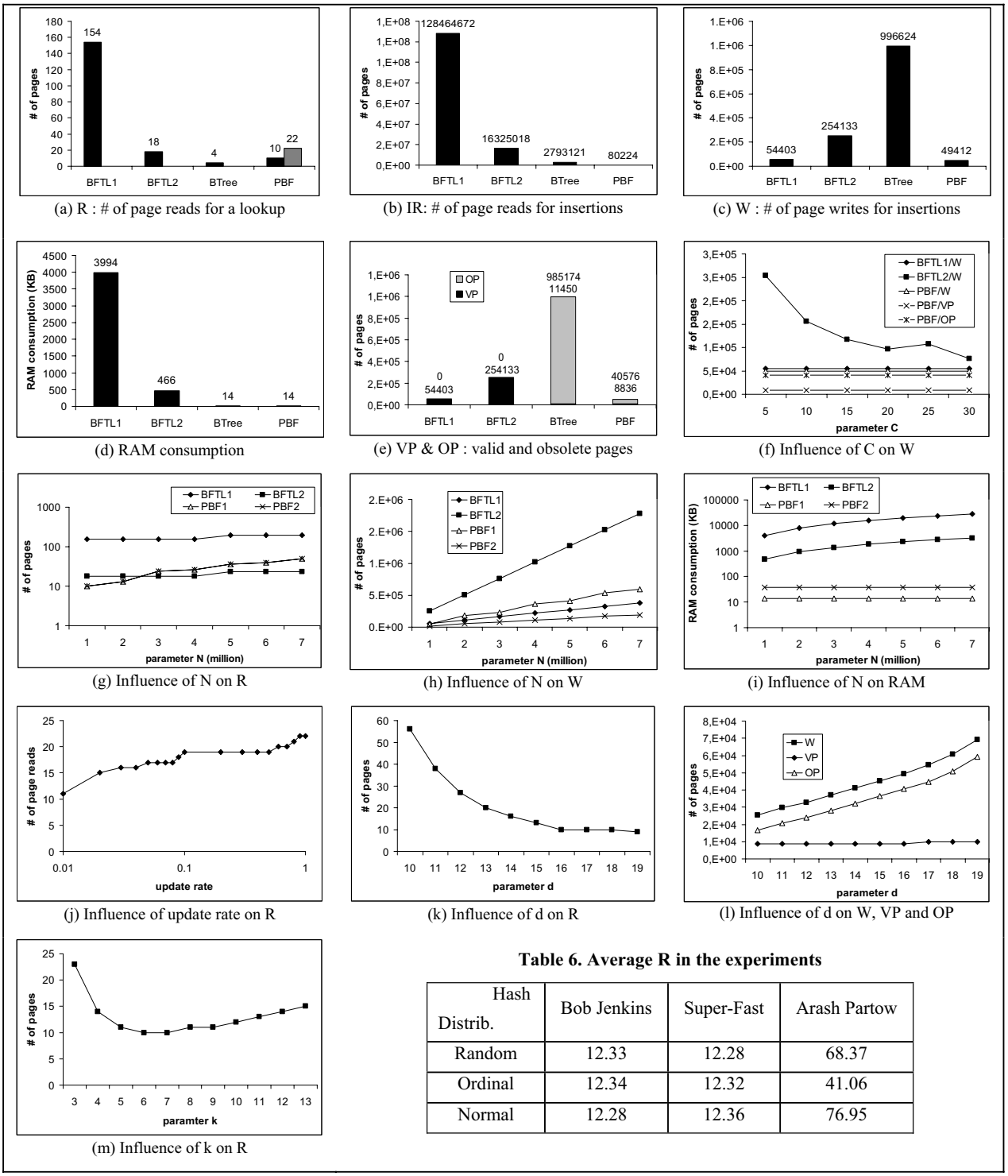



**Table 6. Average R in the experiments**

| Hash / Distrib. | Bob Jenkins | Super-Fast | Arash Partow |
|---|---|---|---|
| Random | 12.33 | 12.28 | 68.37 |
| Ordinal | 12.34 | 12.32 | 41.06 |
| Normal | 12.28 | 12.36 | 76.95 |

**Figure 4. Evaluation results**

### *5.1.4 About Frequent Deletions*

As shown is Figure 4(a), large number of random deletions or updates degrades the lookup performance of PBFilter because of the search in DA. Figure 4(j) shows more precisely the impact of random updates/deletions on metric R when there are 1 million valid tuples. It grows with the update rate (number of updates/number of valid tuples) slowly thanks to DA indexing (e.g., for an update rate of 100%, R = 22). This confirms the benefit to build a single DA area for the complete database if RAM buffers needs to be saved.

## 5.2 PBFilter Adaptability and Predictability

Tuning parameters d and k used to build the Bloom filters in PBF determines the quality of the summarization (false positive rate), the size of the summary and then the partitioning cost with a direct consequence on metrics R, W, VP and OP. This makes PBF adaptable to various situations and brings high opportunities in terms of co-design, assuming the consequences of tuning actions can be easily predicted and quantified.

Figure 4(k) shows the influence of *d* on R with the other parameters set to the previous values: N=1million, Sk=12, B=7, k=7. As expected, the bigger *d*, the smaller the false positive rate and then the better R. At the same time, larger Bloom filters increase the frequency of repartitioning and then increase W and OP in the proportion shown in Figure 4(l). The impact on VP is however very limited because of the small size of SKA compared to KA (e.g., for d=16 and k=7, |SKA|/|KA| = 0.1). Figure 4(m) shows the influence of *k* on R with d=16. The bigger k, the better R, up to a given threshold. This threshold is explained by the Bloom filter principle itself (see formula in Section 4.1 showing that there is an optimal value for k beyond which the false positive rate increases again) and by the fact that a bigger k means scanning more partitions in SKA, a benefit which must be compensated by lower accesses in KA.

As a conclusion, *d* introduces a very precise trade-off between R and W, VP, OP, allowing adapting the balance between these metrics to the targeted application/platform tandem. The choice of *k* under a given *d* should minimize i*k+f*|KA|/2, where i is the number of pages in each final partition.

To illustrate this tuning capability, let us come back to the management of large files. Section 5.1.3 presented a solution to increase PBF scalability in terms of W. The scalability in terms of R can be also a concern for some applications. The decrease of R performance for large files is due to the increase of the number of pages in each final partition and of the average accesses to KA which is f*|KA|/2. The growth of the size of each final partition can be compensated by a reduction of k and a smaller f can still be obtained by increasing d. For instance, the values d=24 and k=4 produce even better lookup performance for N=5 million records (R=9) than the one obtained with d=16 and k=7 for N=1 million records (R=10). The price to pay in terms of Flash memory usage can be precisely estimated thanks to our cost model.

## 5.3 Experimental Results on Real Hardware

### *5.3.1 Platform Description*

PBFilter has been implemented and integrated in the storage manager of an embedded DBMS dedicated to the management of secure portable folders [1]. The prototype runs on a secure USB Flash platform provided by Gemalto, our industrial partner. This platform is equipped with a smartcard-like secure microcontroller connected by a bus to a large (Gigabyte-sized soon) NAND Flash memory (today the 128MB Samsung K9F1G08X0A module), as shown in Figure 5.

The microcontroller itself is powered by a 32 bit RISC CPU (clocked at 50 MHz) and holds 64KB of RAM (half of it preempted by the operating system) and 1MB of NOR Flash memory (hosting the on-board applications' code and used as write persistent buffers for the external NAND Flash).

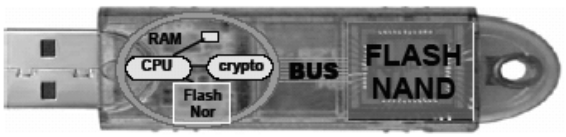



**Figure 5. Secure USB Flash device**

There are three nested API levels to access the NAND Flash module: FIL (Flash Interface Layer) providing only basic controls such as ECC, VFL (Virtual Flash Layer) managing the bad blocks and FTL (Flash Translation Layer) implementing the address translation mechanism, the garbage collector and the wear-leveling policies. We measured the cost of reading/writing one sector/page through each API level using sequential (seq.) and random (rnd.) access patterns. The numbers are listed in Table 7 and integrate the cost to upload/download the sector/page to the Flash module register and the transfer cost from/to the RAM of the microcontroller (masking part of the difference in the hardware cost). FIL and VFL behave similarly for sequential and random access patterns while the variation is significant with FTL. Random writes exhibit dramatic low performance with FTL (a behavior we actually observed in many Flash devices).

**Table 7. I/O Performance through different API levels**

| API Levels | R(µs) sector/page | W(µs) sector/page |
|---|---|---|
| **FIL(seq. & rnd.)** | 100/334 | 237/410 |
| **VFL(seq. & rnd.)** | 109/367 | 276/447 |
| **FTL(seq.)** | 122/422 | 300/470 |
| **FTL(rnd)** | 380/680 | ≈ 12000 |

### *5.3.2 Experimental Results*

We ran our prototype under all the parameter settings used in 5.1 and 5.2. We measured the I/Os and compares the results with those produced by the cost model.

Unsurprisingly, the tests produced exactly the same numbers as those computed by the cost model for all metrics but R. Indeed, the sequential organization and the fixed size of all data structures make the insertion process and the number of repartition steps fully predictable for a given parameter setting, avoiding any uncertainty for IR, W, VP and OP metrics (In the prototype, transaction atomicity is guaranteed thanks to internal NOR Flash buffers and do not interfere with the NAND Flash management).

The discrepancy related to the R metric deserves a deeper discussion. The cost model computes the false positive rate using the formula given in 4.1, assuming the k hash functions are totally independent, a condition difficult to meet in practice. Much work [7, 12] has been done to build efficient and accurate bloom filter hash functions. In our experiment, we compared Bob Jenkins' lookup2, Paul Hsieh's SuperFastHash, and Arash Partow hash over datasets of different distributions (random, ordinal and normal) produced by Jim Gray's DBGen generator. The results show that the degradation of the false positive rate is quite acceptable for the former two hash functions but not for the latter. Table 6 shows the R metric measured for each hash function and data distribution under the setting: N=1 million, Sk=12, d=16, k=7 (the cost model gives R=10 for this setting). About the efficiency of hash functions, Bob Jenkins and SuperFastHash are quite fast (6n+35 and 5n+17 cycles respectively, where n is the key size in bytes), and k independent hash values can be obtained by calling only three times the hash function [7].

We have done preliminary performance measurements in terms of response time for insertions and lookups on top of different API

levels. Today, we are not granted permission by our industrial partner to publish absolute performance numbers, other than those given in Table 7, due to a pending patent. However, the preliminary observations show that (1) the average insertion cost of PBFilter is low in every situations (even on top of FTL) due to its sequential write feature, (2) the lookup cost is very satisfactory on top of FIL and VFL with an increase of nearly 70% on top of FTL and (3) the CPU cost remains low (less than 15% of the total) despite the low frequency of the microcontroller. Further experiments are required to fully capture the behaviour of PBFilter on this hardware platform considering different NAND Flash APIs and variant datasets. We expect that real numbers would be made public soon.

# 6. CONCLUSION

NAND Flash has become the most popular stable storage medium for embedded systems and efficient indexing methods are highly required to tackle the fast increase of on-board storage capacity. Designing these methods is complex due to a combination of NAND Flash and embedded system constraints. To the best of our knowledge, PBFilter is the first indexing method addressing specifically this combination of constraints.

The paper introduces a comprehensive set of metrics to capture the requirements of the targeted context. Then, it shows that batch methods are inadequate to answer these requirements and proposes a very different way to index Flash-resident data. PBFilter, organizes the index structure in a pure sequential way and speeds up lookups thanks to Summarization and Partitioning. A Bloom filter based instantiation of PBFilter has been implemented and a comprehensive performance study shows its effectiveness.

PBFilter is today integrated in the storage manager of an embedded DBMS dedicated to the management of secure portable folders. Thanks to its tuning capabilities, PBFilter seems adaptable to various Flash-based environments and application requirements. Typically, PBFilter seems well adapted to any RAM constrained environment, embedded or not. Our future work is to complete performance measurements on real hardware, to propose an accurate management of secondary keys and to investigate new summarization and partitioning strategies to ever enlarge PBFilter application domain.

# 7. ACKNOWLEDGMENTS

The authors wish to thank Luc Bouganim, Dennis Shasha and Björn Þór Jónsson for fruitful discussions on this paper and Jean-Jacques Vandewalle and Laurent Castillo from Gemalto for their technical support. This research is partially supported by the French National Agency for Research (ANR) under RNTL grant PlugDB and by the Natural Science Foundation of China under grants 60833005, 60573091.

# 8. REFERENCES

[1] Anciaux, N., Benzine, M., Bouganim, L., Jacquemin, K., Pucheral, P., and Yin, S. Restoring the Patient Control over her Medical History. *21th IEEE Int. Symposium on Computer-Based Medical Systems (CBMS)*, 2008.

[2] Anciaux, N., Bouganim, L., Pucheral, P., Valduriez, P. DiSC: Benchmarking Secure Chip DBMS. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), vol. 20, n°10, 2008*.

[3] Birrel, A., Isard, M., Thacker, C., and Wobber, T. A Design for High-Performance Flash Disks. *Operating Systems Review* 41(2), 2007.

[4] Bityutskiy, A-B., JFFS3 Design Issues. *Tech. report*, Nov. 2005.

[5] Bloom, B. Space/time tradeoffs in hash coding with allowable errors. *Communications of the ACM, 13(7)*, 1970.

[6] Dekart SRL.: Dekart Smart Container, 2007. http://www.dekart.com/products/integrated/smart container

[7] Dillinger, P. C., and Manolios, P. Fast and Accurate Bitstate Verification for SPIN. *11th Int. Spin Workshop on Model Checking Software,* LNCS 2989, 2004.

[8] Gonnet, G. and Baeza-Yates, R. *Handbook of Algorithms and Data Structures,* Addison-Wesley, Boston, MA, USA, 1991.

[9] Hamid L. New directions for removable USB mass storage, Press release, 2006. http://www.itwales.com/997893.htm

[10] Intel Corporation, Understanding the Flash Translation Layer (FTL) specification. 1998.

[11] Kim, G., Baek, S., Lee, H., Lee, H., and Joe, M. LGeDBMS: A Small DBMS for Embedded System with Flash Memory. *Int. Conf. on Very Large Data Bases (VLDB),* 2006.

[12] Kirsch, A., and Mitzenmacher, M. Less Hashing, Same Performance: Building a Better Bloom Filter. *Algorithms – ESA 2006, 14th European Symposium,* LNCS 4168, 2006.

[13] Lee, S-W., and Moon, B. Design of Flash-Based DBMS: An In-Page Logging Approach. *Int. Conf. on Management of Data (SIGMOD)*, 2007.

[14] Mani, A., Rajashekhar, M. B., and Levis, P. TINX - A Tiny Index Design for Flash Memory on Wireless Sensor Devices. *ACM Conf. on Embedded Networked Sensor Systems (SenSys)* 2006, Poster Session.

[15] Mitzenmacher, M. Compressed Bloom Filters. *Proceedings of ACM PODC*, 2001.

[16] Nath, S., and Kansal, A. FlashDB: Dynamic Self-tuning Database for NAND Flash. *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2007.

[17] Rosenblum, M., and Ousterhout, J. K. The Design and Implementation of a Log-Structured File System. *ACM Transactions on Computer Systems (TOCS)* 10(1), 1992.

[18] Wu, C., Chang, L., and Kuo, T. An Efficient B-Tree Layer for Flash-Memory Storage Systems. Int. Conf. on *Real-Time and Embedded Computing Systems and Applications (RTCSA)*, 2003.

[19] Yao, A. On Random 2-3 Trees. *Acta Informatica*, 9 (1978).

[20] Yao, S. Approximating the Number of Accesses in Database Organizations. *Communication of the ACM 20(4),*1977.

[21] Yin, S., Pucheral, P., Meng X. PBFilter: Indexing Flash-Resident Data through Partitioned Summaries. Tech. Rep. RR-6548. INRIA. 2008.

[22] Zeinalipour-Yazti, D., Lin, S. V., Kalogeraki, Gunopulos, D., and Najjar, W. MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices. *USENIX Conf. on File and Storage Technologies (FAST)*, 2005.

In the proceedings of the 17th ACM SIGSPATIAL International Conference on Advances
in Geographic Information Systems (GIS2009), page256-265, November 4-6, 2009, Seattle, Washington

# Distortion-based Anonymity for Continuous Queries in Location-Based Mobile Services

Xiao Pan [1] Xiaofeng Meng [1] Jianliang Xu [2]
[1]School of Information
Renmin University of China
{smallpx,xfmeng}@ruc.edu.cn
[2]Dept of Computer Science
Hong Kong Baptist Unversity
xujl@comp.hkbu.edu.hk

## ABSTRACT

Privacy preservation has recently received considerable attention for location-based mobile services. Various location cloaking approaches have been proposed to protect the location privacy of mobile users. However, existing cloaking approaches are ill-suited for continuous queries. In view of the privacy disclosure and poor QoS (Quality of Service) under continuous query anonymization, in this paper, we propose a $\delta_p$-privacy model and a $\delta_q$-distortion model to balance the tradeoff between user privacy and QoS. Furthermore, two incremental utility-based cloaking algorithms — *bottom-up cloaking* and *hybrid cloaking*, are proposed to anonymize continuous queries. Experimental results validate the efficiency and effectiveness of the proposed algorithms.

## Categories and Subject Descriptors

H.2.m [**DATABASE MANAGEMENT**]: Miscellaneous-performance measures]

## General Terms

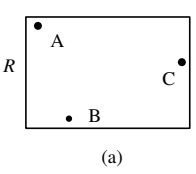
Algorithms, Performance, Information Privacy

## Keywords

Privacy Protection, Continuous Queries, Location-Based Services

## 1. INTRODUCTION

With advances in wireless communication and mobile positioning technologies, location-based services (LBSs) have been gaining increasingly popularity in recent years. Research efforts have been put into investigating how to preserve the privacy of mobile users, while still ensuring high quality of LBSs. In general, there are two types of privacy issues: location privacy [6] (a sensitive location is protected from being linked to a specific user) and query privacy [4] (a query is protected from being linked to a specific user). For example, suppose Alice issues the following continuous query to the service provider (SP) (e.g., GoogleMap) via her mobile phone: "where is the nearest dermatosis hospital for next 30 minutes?". Concerning the location privacy, Alice wants to hide her exact location during her movement (e.g., being in a clinic or pub); concerning the query privacy, she wants to hide the fact that the above query about dermatosis hospital was issued by her.




| User | User location | Cloaking region | Query content |
|---|---|---|---|
| A | $l_A$ | | Hospital |
| B | $l_B$ | $R$ | Hotel |
| C | $l_C$ | | Gas station |

(b)

**Figure 1: Location privacy and query privacy**

To protect the location privacy, Gruteser and Grunwald [6] proposed spatio-temporal cloaking based on a location $k$-anonymity model, that is, the cloaked location is made indistinguishable from the location information of at least $k$-1 other users. To achieve location $k$-anonymity, each user location is extended to a cloaking region such that each region covers at least $k$ users. Figure 1(a) illustrates an example of location 3-anonymity ($k$=3), where the locations of A, B and C are extended to region $R$(i.e., users A, B and C form a *cloaking set*), such that the adversary cannot figure out their genuine locations in $R$. Under some circumstances, the adversary knows the users' genuine locations [4]; thus, the location contained in a query would become a quasi-identifier (QI) [12] to link the query to a specific user. Fortunately, the location $k$-anonymity model is also applicable to tackle this query privacy issue. Consider the example shown in Figure 1(b), by simply extending the locations contained in the query to the same region $R$ , the exact query locations can be successfully hidden and hence the query privacy is preserved.

Most of the existing cloaking algorithms focus on anonymizing *snapshot* queries [11, 5, 7]. As the cloaking sets for the same user are different at different timestamps [4], directly applying these algorithms to *continuous* queries is not sufficient to protect the query privacy. Figure 2 depicts an example where the query privacy is disclosed under continuous queries. As shown, six users A∼F issue six different continuous queries $Q_1 \sim Q_6$, respectively, and A is successfully cloaked as {A,B,D}, {A,B,F} and {A,E,C} at the timestamps $t_i$, $t_{i+1}$ and $t_{i+2}$, respectively. Each of these cloaking sets is consistent to the location 3-anonymity. However, as their intersection contains A only, the adversary can easily infer that A issued query $Q_1$ and, hence, A's query privacy is disclosed.


*ACM GIS* '09, November 4-6,2009. Seattle, WA, USA
(c) 2009 ACM ISBN 978-1-60558-649-6/09/11 ...$10.00.

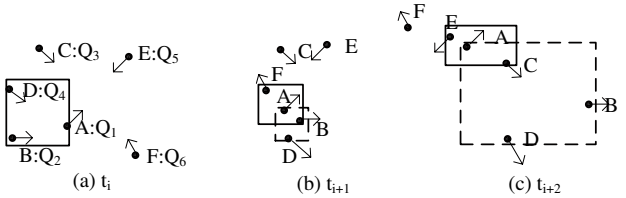



**Figure 2: A continuous query example**

As can be observed from the above example, the privacy disclosure is due to the use of different cloaking sets for the same user at different timestamps. To conquer this problem, queries issued from the same cloaking region should stick together at all timestamps [4]. In detail, under such a scheme, the cloaking set of A in Figure 2 should always be kept as {A, B, D} during $t_i$ through $t_{i+2}$ (the cloaking regions are represented as dashed rectangles in Figures 2(b) and 2(c)). Although this scheme successfully protects the query privacy, it leads to new problems: 1) User location privacy might be disclosed. As shown in Figure 2(b), the minimum bounding rectangle (MBR) of {A, B, D} shrinks to a smaller region at $t_{i+1}$ which might violate the location privacy requirement of A. In the worst case, it may shrink to a point, thereby exposing the genuine user location. 2) The quality of service becomes poor. As shown in Figure 2(c), the size of the cloaking region {A, B, D} is significantly large at $t_{i+2}$, which would make the subsequent query processing much more costly. In an extreme case, the users might scatter over the entire space over time, forcing the cloaking region to cover the whole area.

The reason behind the aforementioned new problems is that, the algorithm exploits the proximity of current user locations only, but ignores their future locations. As we known, a user's future location depends on the velocity of her movement and the duration of the continuous query. In an ideal case, all users within the same cloaking set move with the same velocity such that the size of the cloaking region remains the same at all timestamps. Unfortunately, this is unlikely to happen in practice, and the location proximity tends to change once the involved locations update. Specifically, on one hand, queries whose locations are close at the current timestamp may become far away from each other at a future timestamp; on the other hand, queries who are now far away from each other may meet at some future timestamp. For a continuous query whose location is dynamically changing, it is hard to find an optimal cloaking region for all timestamps. The main challenge is how to achieve a good QoS while still preserving query privacy during the query period with frequent location updates.

In this paper, we consider protection of both location privacy and query privacy for both continuous queries. To address this issue, we propose a $\delta_p$-privacy model and a $\delta_q$-distortion model to balance the tradeoff between user privacy and QoS. The perimeter of a cloaking region is adopted to evaluate the distortion of the location information. As pointed out in [10], moving objects with similar patterns would move in a cluster eventually. Motivated by this observation, we propose to map the location distortion to a similarity distance of the queries, based on which queries are clustered such that the distortion of location information in each cluster is minimized. These clusters are incrementally maintained as queries move in and out.

The contributions we make in this paper can be summarized as follows:

- We propose a $\delta_p$-privacy model and a $\delta_q$-distortion model to balance the tradeoff between user privacy and QoS under continuous queries.
- We propose to map the location distortion to a temporal similarity distance of the queries. Furthermore, we propose two incremental utility-based cloaking algorithms.
- A series of experiments is conducted to evaluate the performance of our proposed algorithms. The experimental results validate the efficiency and effectiveness of our proposed algorithms.

The rest of the paper is organized as follows. We review the related work in Section 2. The problem under investigation is formally defined in Section 3. Several utility-based cloaking algorithms are proposed in Section 4. Algorithms for distortion and privacy verification are proposed in Section 5. Section 6 presents the performance evaluation results of our proposed algorithms. Finally, the paper is concluded in Section 7.

## 2. RELATED WORK

Location privacy and query privacy are two types of privacy issues concerned in location-based mobile services. Location $k$-anonymity is the most popular location privacy metric. It was proposed by Gruteser and Grunwald [6], and was later refined in [11, 1]. In terms of the techniques used for protecting location privacy, existing approaches can be classified into cloaking [6], dummy [8], and encryption [5]. However, all of the above work focuses on privacy protection for *snapshot* queries.

Most of the prior research does not distinguish location privacy and query privacy.The first work to distinguish them and to explore privacy protection for continuous queries is presented in [4]. Nonetheless, it has two drawbacks. First, only the query locations at the issuing time are employed to generate cloaking sets, which may lead to location privacy disclosures and poor QoS, as discussed in the previous section. Second, as the valid period of each query is ignored, continuous queries may be cloaked with snapshot queries. If any snapshot query moves out of the service area, all continuous queries within the same cloaking set may no longer meet the location $k$-anonymity privacy requirement. Our work also employs location $k$-anonymity and memorizes users, but it differs from [4] in the following aspects. First, a temporal location distortion model is employed to find the cloaking set. Second, the queries with similar expiration times are clustered together, which guarantees that every continuous query will always satisfy the privacy requirement during its valid period.

Another work addressing location anonymity for continuous queries is presented in [15]. It employs *entropy* to measure the anonymity level of a cloaking region by assuming that the probabilities of users in a cloaking region are not equal. However, as entropy does not consider whether the user locations are really different or not, location privacy might be disclosed when $k$ different users are at the same location. In [13], a mobility-aware cloaking algorithm is proposed to defend trace analysis attacks. However, the privacy metric employed in [13] is location granularity, rather than location $k$-anonymity considered in this paper.

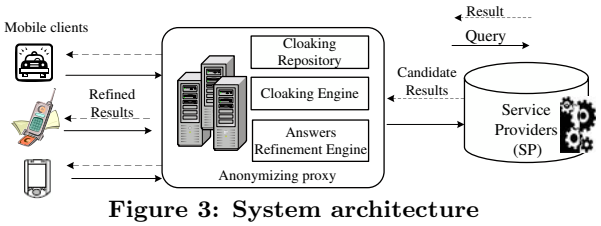



**Figure 3: System architecture**

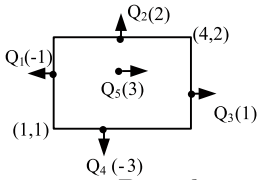



**Figure 4: Boundary velocities**

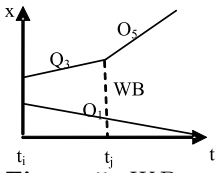



**Figure 5: $WB$**

Although most research on privacy protection in LBSs does not discuss about the location data utility, some of its measurements have been proposed in data publishing, such as *generalization height*, *discernibility*, *information loss*, *classification metric* and *information-gain-privacy-loss ratio* [9]. Data utility in data publishing mainly focuses on how the distribution of the original data is preserved for the purpose of data mining, whereas in LBSs, we mainly focus on how the generalized location approximates to the original location. Therefore, in this paper, we employ information loss, namely location distortion, as the utility measure. We remark that our location distortion is different from that of [14] in the following aspects. First, as QI attributes in data publishing are independent, each attribute can be associated with a weight to reflect its importance. Nevertheless, the information in LBSs, including location $(x, y)$ and velocity $v$, is dependent to each other w.r.t. time $t$. Second, the information distortion in [14] is static as long as the anonymizing table is given, while the location distortion in our paper is a temporal function, which changes as time evolves.

# 3. PRELIMINARIES

## 3.1 System Architecture

Like most existing work [6, 11], we employ a centralized system, which consists of mobile users, a trusted anonymizing proxy, and an un-trusted SP, as shown in Figure 3. Each mobile user sends location-based queries to the anonymizing proxy. There are two types of queries: *new query* and *active query*. New query, as the name implies, is a query newly issued by a user. Active query is a continuous query which was issued at some previous time but not yet expired. For example, a user issues a continuous query $Q$ at $t_i$, and its valid period is $\Delta t$. Then, at $t_i$, $Q$ is a new query, while for any $t \in (t_i, t_i + \Delta t]$, it is regarded as an active query.

The anonymizing proxy consists of *cloaking engine*, *cloaking repository* and *answer refinement engine*. Upon receiving a new query, *cloaking engine* replaces the user *id* with a pseudonym $id'$. Meanwhile, it invokes the location cloaking algorithm to generate a cloaking region in accordance with the user's privacy requirement. This cloaking set is saved in the cloaking repository in the form of $(CID, Qset, R_{L,t}, R_{v,t})$[1]. Upon receiving an active query, *cloaking engine* searches for the original cloaking set, which was generated at the issuing time, in the cloaking repository and then computes the new cloaking region $R_{L,t}$. Later on, the anonymizing proxy forwards the cloaked query to the SP. By maintaining a cloaking repository, the anonymizing proxy can incrementally compute the cloaking set (i.e., by updating the original cloaking region) for the active queries and thereby achieving a higher efficiency.

Finally, candidate results generated by SP are first refined by *answer refinement engine*, and then relayed to the mobile user. In this paper, we focus on the location cloaking algorithm, which considers the location data utility as well as the user-specified privacy requirements.

To facilitate our study, we further make the following assumptions. 1) *Every mobile user is trusted* — this is a common assumption in the conventional location privacy preserving techniques [6, 11, 1]. 2) *The movement velocity of a user remains unchanged during the query period.* Under this assumption, the movement function of every query is linear.

[1]The meanings of these parameters will be explained later in Definition 2.

## 3.2 Preliminaries

*Definition 1.* (Location-based query) Query $Q$ is represented as $Q = (l, \bar{v}, t, T_{exp}, con)$, where $(l, \bar{v}, t)$ implies $Q$ is at location $l$=($x$, $y$) with velocity $\bar{v} = (v_x, v_y)$ at the timestamp $t$, $T_{exp}$ is the timestamp when the query expires, and $con$ is the content of this query.

*Definition 2.* (Cloaking set) Each cloaking set $CS$ is formalized as:

$$CS = (CID, Qset, R_{L,t}, R_{v,t}),$$

where $CID$ is the identifier of this cloaking set, $Qset$ is the set of queries contained in $CS$, $R_{L,t} = (L_{x-,t}, L_{y-,t}, L_{x+,t}, L_{y+,t})$ is the location MBR for the queries in $Qset$ at $t$, and $R_{v,t}$= $(v_{xmin,t}, v_{ymin,t}, v_{xmax,t}, v_{ymax,t})$ is the BVR (Boundary Velocity Rectangle). $v_{xmin,t}$=min($v_{x+,t}$, $v_{x-,t}$), $v_{xmax,t}$=max($v_{x+,t}$, $v_{x-,t}$), $v_{ymin,t}$=min($v_{y+,t}$, $v_{y-,t}$), $v_{ymax,t}$=max($v_{y+,t}$, $v_{y-,t}$), where $v_{x-,t}$ ($v_{x+,t}$) is the boundary velocity of the query at $L_{x-,t}$ ($L_{x+,t}$) on the x dimension, and $v_{y-,t}$ ($v_{y+,t}$) is the boundary velocity of the query at $L_{y-,t}$ ($L_{y+,t}$) on the y dimension at time $t$.

Note that $v_{xmax,t}$ ($v_{ymax,t}$) and $v_{xmin,t}$ ($v_{ymin,t}$) may not be the maximum and minimum velocity in $Qset$ on the x (y) dimension. Hence, the queries on the boundary of a cloaking set change with users movements. Thus, $R_{v,t}$ and $R_{L,t}$ are both piece-wise functions w.r.t. $t$, i.e.:

$$\begin{aligned}(L_{x-,t}, L_{y-,t}) &= (L_{x-,t_{i-1}}, L_{y-,t_{i-1}}) + (v_{x-,t}, v_{y-,t}) \times [t - t_{i-1}] \\ (L_{x+,t}, L_{y+,t}) &= (L_{x+,t_{i-1}}, L_{y+,t_{i-1}}) + (v_{x+,t}, v_{y+,t}) \times [t - t_{i-1}]\end{aligned} \quad (1)$$

where $t \in [t_{i-1}, t_i]$. Taking Figure 4 as an example, queries $Q_1 \sim Q_5$ constitute a cloaking set $CS$ at time $t_i$. The number in parenthesis is the query's velocity and the arrow indicates the movement direction. $CS.R_{L,t_i}$=(1,1,4,2) and $CS.R_{v,t_i} = (-1, -3, 1, 2)$. Here the maximum speed is 3 on the x dimension (for $Q_5$), but it is not a boundary speed at time $t_i$. If $Q_5$ overtakes $Q_3$ at time $t_j$, $v_{xmax}$ will change to 3 at $t_j$.

*Definition 3.* (Width/Height of boundary) For a cloaking set $CS$ with MBR $R_{L,t}$, its width at time $t$, denoted as $WB_t$, is

$$WB_t = L_{x+,t} - L_{x-,t} \quad (2)$$

Similarly, its height at time $t$, denoted as $HB_t$, is

$$HB_t = L_{y+,t} - L_{y-,t} \quad (3)$$

$WB_t/HB_t$ is also a piece-wise linear function. For the example in Figure 4, the changing trend of $WB$ is shown in Figure 5. The changing trend of $HB$ is similar; so we omit it here.

Recall that privacy is protected by reducing the resolution of the location information. Obviously, the more does the privacy preserve, the less is the utility of the location data. In this paper, we use distortion to measure the utility of anonymized location. In other words, distortion reflects the location information loss, i.e., how much is sacrificed for privacy preserving. The smaller is the distortion value, the higher is the data utility.

*Definition 4.* (Distortion of a query) Let query $Q \in CS$, whose MBR (BVR) at time $t$ is denoted as $R_{L,t}$ ($R_{v,t}$). Let $A_{width}$ ($A_{height}$) be the width (height) of the whole space. The distortion for a query $Q$ at time $t$ is defined as

$$Distortion_{R_{v,t}}(Q, R_{L,t}) = \frac{(L_{x+,t} - L_{x-,t}) + (L_{y+,t} - L_{y-,t})}{A_{height} + A_{width}}.$$

Thus, the distortion of $Q$ during its valid period can be represented as

$$\int_{T_s}^{T_{exp}} Distortion_{R_{v,t}}(Q, R_{L,t})dt \quad (4)$$

where $T_s$ is the timestamp when $Q$ is cloaked successfully.

For the sake of convenience, let $P_A = A_{height} + A_{width}$, $P_{L,t} = (L_{x+,t} - L_{x-,t}) + (L_{y+,t} - L_{y-,t})$, $P_{v,t} = (v_{x+,t} - v_{x-,t}) + (v_{y+,t} - v_{y-,t})$. Let $TSet$ denote the set of timestamps $\{t_1, t_2, \cdots, t_n\}$ ($t_1 = T_s, t_n = T_{exp}$) when the boundary queries change. Then, Equation (4) can be rewritten as

$$\begin{aligned}
&\int_{T_s}^{T_{exp}} Distortion_{R_{v,t}}(Q, R_{L,t})dt \\
&= \frac{1}{P_A}\{\int_{t_1}^{t_2} [P_{L,t_1} + P_{v,t_1}(t - t_1)]dt + \\
&\cdots + \int_{t_{i-1}}^{t_i} [P_{L,t_{i-1}} + P_{v,t_{i-1}}(t - t_{i-1})]dt + \\
&\cdots + \int_{t_{n-1}}^{t_n} [P_{L,t_{n-1}} + P_{v,t_{n-1}}(t - t_{n-1})]dt\}
\end{aligned}$$

*Definition 5.* (Distortion of a cloaking set) Let $CS$ be a cloaking set with MBR $R_{L,t}$ and BVR $R_{v,t}$ at time $t$. The distortion of $CS$ at time $t$ is defined as

$$Distortion_{R_{v,t}}(CS, R_{L,t}) = \sum_{Q_i \in CS} Distortion_{R_{v,t}}(Q_i, R_{L,t})$$

Thus, the distortion of $CS$ during its valid period is defined as

$$\int_{T_s}^{maxT} Distortion_{R_{v,t}}(CS, R_{L,t})dt \quad (5)$$

where $T_s$ is the timestamp when $CS$ is generated and $maxT = max_{Q_i \in CS}(Q_i.T_{exp})$.

For any two queries, if their states (i.e., initial locations and velocities) are similar, their future locations tend to be near to each other. In extreme cases, if two queries are on the same initial location with the same velocity, they will have the same location during their common valid periods. This implies that if queries with similar states are cloaked together, their distortions during the valid period are likely to be small. This observation inspires us to map the distortion with the similarity distance of queries, as defined as follows:

*Definition 6.* (Temporal similarity distance between two queries) Let $Q_1$ and $Q_2$ be two queries, and they constitute a cloaking set $CS_{12}$ with MBR (BVR) $R_{L12,t}$ ($R_{v12,t}$) at time $t$.

The temporal similarity distance between $Q_1$ and $Q_2$ is defined as

$$SimDis(Q_1, Q_2) = \int_{T_s}^{maxT} Distortion_{R_{v12,t}}(CS_{12}, R_{L12,t})dt$$

where $maxT$=max($Q_1.T_{exp}, Q_2.T_{exp}$).

The similarity distance possesses the following properties:

- SimDis($Q_1$,$Q_1$)=0
- SimDis($Q_1, Q_2$)=SimDis($Q_2, Q_1$)
- SimDis($Q_1, Q_2$)$\leq$ SimDis($Q_1, Q_3$)+SimDis($Q_3, Q_2$)

The proof is obvious. Due to space limitations, we omit it here.

*Definition 7.* (Temporal similarity distance between two query sets) Let $U_1$ and $U_2$ denote two non-interleaved query sets (i.e., $U_1 \cap U_2 = \phi$), and $U = U_1 \bigcup U_2$. $R_{L,t}$ ($R_{v,t}$) denotes the MBR (VBR) of $U$ at time $t$.

The similarity distance between $U_1$ and $U_2$ is defined as

$$\begin{aligned}
SimDis(U_1, U_2) = &\int_{T_s}^{maxT} Distortion_{R_{v,t}}(U_1, R_{L,t})dt + \\
&\int_{T_s}^{maxT} Distortion_{R_{v,t}}(U_2, R_{L,t})dt
\end{aligned}$$

where $maxT = max_{Q \in U}(Q.T_{exp})$. It is easy to know that the similarity distance between two queries can be regarded as the special case for two query sets where $|U_1| = |U_2|$=1.

## 3.3 Privacy Model

Recall that the queries within the same cloaking set are required to stick together before they expire, which makes it hard to strike a good balance between the user privacy and QoS for continuous queries. In this section, two models, namely, $\delta_p$-*privacy* and $\delta_q$-*distortion*, are proposed to formalize user privacy and QoS requirements.

The location and velocity of a query are projected to the x and y dimensions. Now let us first discuss a one-dimension case. Assume that a candidate cloaking set contains three queries $\{Q_1, Q_2, Q_3\}$, whose velocities on the x dimension are shown in Figure 6(a). From the figure, we can see that $WB$ decreases in early stage and shrinks to a point at $T_w$; after that, $WB$ increases again. Similar observations can be drawn on the y dimension. In the worst case, two boundary segments on the x dimension and y dimension would shrink at same time, as shown in Figure 6(b). Consequently, the cloaking region would shrink to a point, which leads to the exposure of the genuine location.

Location disclosures are prohibited, regardless of how many dimensions they are disclosed on. If neither $WB$ nor $HB$ has the opportunity to shrink to a point, apparently, the privacy would be preserved. A privacy model is designed to formalize how much $WB$ and $HB$ are allowed to shrink.

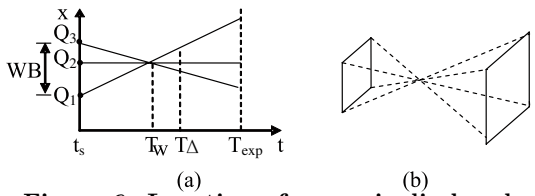



**Figure 6: Location of query is disclosed**

*Definition 8.* ($\delta_p$-privacy model) Let $WB_t$ ($HB_t$) be the width (height) of the boundary segment on the x (y) dimension at time $t$, and $\delta_p$ be the privacy threshold specified by the users. If for $\forall t \in [T_s, maxT]$, $\min(WB_t, HB_t) \geq \delta_p$, $\delta_p$-privacy is satisfied.

While Definition 8 guarantees the protection of user privacy, the following Definition 9 ensures the quality of services. The distortion of a query set $CS$ should not grow larger than the user's requirement $\delta_q$, which is the user's tolerable worst service quality. Note that forcing the distortion to be smaller than $\delta_q$ at $T_s$ cannot ensure that it always meets the requirement $\delta_q$ during the entire period $T_s$ through $maxT$.

*Definition 9.* ($\delta_q$-distortion model) Assume that the user's tolerable worst service quality is $\delta_q$, the set of user queries is $CS$ with MBR $R_{L,t}$ and VBR $R_{v,t}$. If for any $t \in [T_s, maxT]$ and any $Q \in CS$, $Distortion_{R_{v,t}}(Q, R_{L,t}) \leq \delta_q$, then $\delta_q$-distortion is satisfied.

In summary, the requirements for a successful cloaking set $CS$ include:

- $|CS| \geq K$;
- Let $minT = min_{Q \in CS} Q.T_{exp}$, $maxT = max_{Q \in CS} Q.T_{exp}$, $maxT - minT \leq \delta_T$;
- $CS$ satisfies $\delta_p$-privacy and $\delta_q$-distortion during $[T_s, maxT]$.

# 4. UTILITY-BASED ALGORITHMS

## 4.1 Greedy Cloaking Algorithm

The main idea of the greedy cloaking algorithm (GCA) is as follows. For every newly arrived query $r$, we first compute its temporal similarity distance with those existing queries which have not yet been anonymized successfully. Then, the one having the minimal similarity distance with $r$ is put into the cloaking set. The above steps repeat until no more queries can be added into the cloaking set. We detail it in Algorithm 1.

Specifically, when a new query $r$ arrives, it is first inserted into the candidate cloaking set $U$ (see step 1∼2). Then each query $r_m$ in the set of existing queries which are not yet anonymized (denoted as $RSet$) is retrieved. If the difference between $r_m.T_{exp}$ and $r.T_{exp}$ is bigger than the value of $\delta_T$, then $r_m$ cannot be clustered with $r$ and therefore it is filtered out (see step 5). Otherwise, the boundary queries during its valid period are calculated and each boundary query is stored in the **b**oundary **t**ime **q**ueue (BTQ) $bq$, which would be detailedly elaborated in Section 5.1. After all boundary queries are captured, the $\delta_q$-distortion requirement (the detailed procedure is given in Section 5.2) is verified. If true, the request $r_{min}$ with the minimal temporal similarity distance is inserted into $U$. The above steps repeat until no more queries can be put into $U$ or $|U| \geq K$. Finally, if $U$ covers more than $K$ users, the $\delta_p$-privacy is also verified (we will present its detailed procedure in Section 5.3). If both $\delta_q$-distortion and $\delta_p$-privacy are satisfied, $U$ is returned as the cloaking set.

**Algorithm 1** : Greedy cloaking algorithm(GCA)

```
1: a candidate cloaking set U=null;
2: put r into U;
3: while true do
4:   for each query r_m in RSet do
5:     if |r.T_exp − r_m.T_exp|>δ_T then
6:       get the next query in RSet;
7:     else
8:       BoundaryObjectsComputing(r_m, bq, U)
9:       if DistortionDetection(r_m, bq, U)=true then
10:        dis=SimDis(r_m, U);
11:        if (mindis>dis) then
12:          mindis=dis;
13:          r_min=r_m;
14:  insert r_min into U;
15:  RSet = RSet - {r_min};
16:  if |U| does not change or |U| ≥ K then
17:    break;
18: if (|U| ≥ K) then
19:  Check δ_p-privacy and return cloaking set;
```

## 4.2 Bottom-up Cloaking Algorithm

The drawback of GCA is that for every newly arrived query, it needs to search the cloaking set from scratch, which incurs expensive computational cost. Actually, intermediate results computed in the previous iterations can be exploited. The basic mechanism is to cluster those queries whose distortions are always less than $\delta_q$ together during their valid periods, and then to incrementally maintain these clusters. Obviously, the cloaking sets are the subsets of these clusters. Before presenting our new cloaking algorithm, we give the definition of continuous cluster:

*Definition 10.* (Continuous cluster) A query set $C$ is a continuous cluster during $[t_1, t_2]$ if (1) $C$ satisfies $\delta_q$-distortion; (2) $maxT_{exp} - minT_{exp} \leq \delta_T$, where $maxT_{exp} = max_{Q \in C}(Q.T_{exp})$ and $minT_{exp} = min_{Q \in C}(Q.T_{exp})$

Based on the continuous clusters, the basic idea of *bottom-up cloaking* algorithm (BCA) is as follows. When a new query $r$ arrives, $r$ itself forms a cluster $\{r\}$, which naturally satisfies $\delta_q$-distortion. Then, among the existing continuous clusters, the one with the minimal temporal similarity distance with $r$, denoted as $c_r$, is selected to merged with $\{r\}$. If $\{c_r, r\}$ contains not less than $K$ queries, it is verified to see if it meets the $\delta_p$-privacy requirement. If fails, this cluster is kept in service space for merging future queries. Algorithm 2 shows the pseudo-code of *bottom-up cloaking* upon the arrival of a new query $r$.

In Algorithm 2, $r$ is the newly arrived query, and $CR$ is the set of existing continuous clusters in service space. For each cluster $c$ in $CR$, it associates with a BTQ $bq_c$, which maintains its boundary queries during the valid period. When $r$ arrives, each cluster $c$ in $CR$ is scanned and the following steps are conducted. First, new boundary queries of $c$ with $r$ inserted are computed (see step 5); Second, $\delta_q$-distortion is checked (see step 6); Third, the temporal similarity distance between $c$ and $r$ is calculated (see step 8). After that, the

**Algorithm 2** : Bottom-up cloaking algorithm(BCA)

```
1: for each cluster c ∈ CR do
2:    maxT=max(c.maxT_exp, r.T_exp);
3:    minT=min(c.minT_exp, r.T_exp);
4:    if maxT − minT ≤ δ_T then
5:       BoundaryObjectsComputing(r, bq_c, c);
6:       if DistortionDetection(r, bq_c, c)=false then
7:          continue;
8:       dist=SimDis(r, c);
9:       if dc_min>dist then
10:         dc_min=dist;
11:         c_min = c;
12:      else
13:         if dc=dist and |c_min| < |c| then
14:            dc_min=dist;
15:            c_min=c;
16: if c_min not exists then
17:    put {r} into CR;
18: else
19:    c_min = c_min ∪ {r};    /*do the merging*/
20:    if |c_min| ≥ K then
21:       Check δ_p-privacy and return the cloaking set c_min;
```

cluster $c_{min}$ which has the minimal temporal similarity distance with $r$ is sought. If such $c_{min}$ exists, $c_{min}$ is updated by merging it with $\{r\}$. Otherwise, $r$ itself forms a cluster and is added into $CR$. Note that if there exist two clusters with the same minimal similarity distance with $r$, the one with more queries is preferable to be chosen for $\{r\}$ to be merged with (see step 13∼15) so that more queries can be cloaked.

Consider the example shown in Figure 7, where $C_1$, $C_2$, $C_3$ and $C_4$ are continuous clusters, whose distortions are always smaller than $\delta_q$ before they expire. When a new query $r$ arrives, according to BCA, $C_4$ is selected to be merged with $\{r\}$ as it has the minimal temporal similarity distance with $r$. Then, $C_4$ would have four queries after the merging is conducted. If $K \leq 4$, $C_4$ is sent to check the $\delta_p$-privacy requirement, otherwise, it will stay in the service space to merge with other arriving queries.

## 4.3 Hybrid Cloaking Algorithm

Many possible cluster merges are ignored in *bottom-up cloaking* algorithm (BCA), which might decrease the success rate of cloaking. Continue with the example in Figure 7, as discussed previously, $C_4$ cannot be returned as a cloaking set if $K$=5. However, by merging some queries of other cluster, e.g., $C_1$, it is possible that $C_4$ would successfully become the cloaking set. In fact, $C_4 \cup \{A\}$ satisfies both $\delta_q$-distortion and $\delta_p$-privacy requirements, and thus would be returned as a cloaking set. However, such opportunities are omitted in BCA. On the other hand, as cluster merges are time-consuming, especially when the locations of queries are frequently updating, conducting merges in the granularity of queries surely deteriorates the performance of the cloaking algorithms.

In order to resolve this problem, we propose *hybrid cloaking* algorithm, which aims to combine the advantages of BCA and GCA together. Specifically, BCA is used to search the proper cluster for each newly arrived query, while GCA is for cluster refinement. To further improve the searching efficiency, a TPR-tree is adopted to index existing clusters,

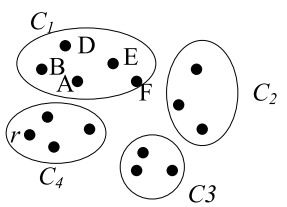


**Figure 7: BCA**

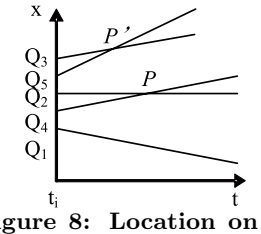


**Figure 8: Location on x-dimension**

such that some clusters can be filtered out and thereby accelerating the searching process. Before presenting the specific algorithm, for ease of exposition, we introduce *nearest neighbor cluster* whose definition is as follows:

*Definition 11.* (Nearest neighbor cluster, NNC) A cluster $C_n$ is the NNC of $C$ iff for any cluster $C_i (C_i \neq C$ and $i \neq n)$, perimeter(MBR($C_i, C$))> perimeter (MBR($C_n, C$)).

With the help of TPR-tree, we can easily get the set of nearest neighbor clusters (denoted as $CS_{nn}$) for each newly arrived query $r$ during its valid period. The following theorem shows that, if there exists a cluster $C$ in $CS_{nn}$ and $C$'s distortion with $r$ violates $\delta_q$-distortion, then the distortions between other clusters and $r$ must also violate $\delta_q$-distortion.

THEOREM 1. *Let $CS_{nn}$ be query $r$'s CNN during its valid period $[T_s, T_{exp}]$. If $\exists\ C_{i,t_i,t_{i+1}} \in CS_{nn}$,*

$$distortion(C_{i,t_i,t_{i+1}}, r, t) > \delta_q$$

*where $t \in [t_i, t_{i+1}], T_s < t_i < t_{i+1} < T_{exp}$, then for any cluster $C' (C' \neq C_{i,t_i,t_{i+1}})$,*

$$distortion(C', r, t) > \delta_q.$$

PROOF. For ease of presentation, without loss of generality, we assume every cluster has the same valid period as $r$. $CS_{nn}$ is in the form of $\{(C_{1,t_1,t_2}, t_1, t_2), \ldots, (C_{i,t_i,t_{i+1}}, t_i, t_{i+1}), \ldots, (C_{n,t_{n-1},t_n}, t_{n-1}, t_n)\}$, where $t_1 = T_s$, $t_n = T_{exp}$, and $(C_i, t_i, t_{i+1})$ represents that $C_{i,t_i,t_{i+1}}$ is $r$'s NN during $[t_i, t_{i+1}]$. For any cluster $C'$ which is not NN at timestamp $t$ ($t \in [t_i, t_{i+1}]$), according to Definition 11, perimeter($C', r$) > perimeter ($C_{i,t_i,t_{i+1}}, r$) at $t$. Apparently, if distortion ($C_{i,t_i,t_{i+1}}, r, t$) > $\delta_q$, we have distortion ($C', r, t$)> $\delta_q$. □

According to Theorem 1, for a newly arrived request $r$, if its nearest neighbor cluster at any timestamp of its valid period violates the $\delta_q$-distortion requirement, other clusters are filtered out from checking and thus some computational cost can be saved. However, two new problems arise: 1) how to efficiently find CNN of $r$ during its valid period; 2) how to find the cloaking set based on $CS_{nn}$.

As the queries within a cluster are dynamically changing in terms of their locations and velocities, under such circumstance, it is a complicated and time-consuming to identify the $CS_{nn}$ for a query. However, since the purpose of $CS_{nn}$ is just for filtering, instead of finding out the exact nearest neighbor for a query, we turn to the approximate computing. Before that, we first define the centroid of a cluster.

*Definition 12.* (Centroid of a cluster) The centroid $O_{cn}$ of a cluster $C$ is represented as $(x, y, v_x, v_y)$, where (1)$x = \frac{\sum_{Q \in C} Q.x}{|C|}$ and $y = \frac{\sum_{Q \in C} Q.y}{|C|}$; (2) $v_x = \frac{\sum_{Q \in C} Q.v_x}{|C|}$ and $v_y = \frac{\sum_{Q \in C} Q.v_y}{|C|}$

**Algorithm 3** : Hybrid cloaking algorithm(HCA)

```
1: find the CNN cluster CS_nn for r on TPR-tree;
2: invoke BCA on CS_nn to find cluster c_min;
3: if c_min not found then
4:    insert r into TPR-tree;
5: else
6:    c_min = c_min ∪ {r};    /*do the merging*/
7:    if |c_min| < K then
8:       for each query o in CS_nn − c_min do
9:          maxT=max(c_min.maxT_exp, o.T_exp);
10:         minT=min(c_min.minT_exp,o.T_exp);
11:         if |maxT − minT| < δ_T then
12:            if DistortionDetection(o, bq, c_min)=false then
13:               insert o into c_min;
14:               delete o from the cluster c it is in;
15:               insert c into queue uq;
16:               if |c_min| ≥ K then
17:                  break;
18:   for each cluster c in uq do
19:      update centroid of cluster c ;
20:      update it in TPR-tree;
21:   if |c_min| ≥ K then
22:      Check δ_p-privacy and return the cloaking set c_min;
23:   else
24:      insert centroid of c_min into TPR-tree;
```

Now, each cluster can be simply represented by its centroid, and a TPR-tree can be built on the centroids of clusters. Also, the NNC for a query can be quickly discovered by searching over this TPR-tree. In other words, the original problem is successfully transformed to a traditional CNN problem on moving objects, which has been well studied in literature. We detail the proposed *hybrid cloaking* in Algorithm 3. We employ best-first traversal using min metric [2] to compute $CS_{nn}$ for $r$ (step 1). Based on $CS_{nn}$, BCA is adopted to find the cluster $c_{min}$ with minimum temporal similarity distance first (step 2). If such $c_{min}$ does not exist, $r$ itself forms a cluster and its centroid is inserted into TPR-tree (step 4). Otherwise, the merging of $\{r\}$ and $c_{min}$ is conducted. Finally, if $c_{min}$ has not less than $K$ queries, it is directly returned as a candidate cloaking set for privacy verification (step 22). Otherwise, we invoke GCA to do the cluster refinement (step 7∼17) — for each query $o$ in $CS_{nn}$ but not in $c_{min}$, if $c_{min} \cup \{o\}$ satisfies $\delta_q$-distortion, then $o$ is moved into $c_{min}$ and their centroids in TPR-tree are updated correspondingly. Such process repeats until $c_{min}$ contains $K$ queries.

Continue with the example shown in Figure 7. Suppose that the set of clusters $\{C_1, C_2, C_3, C_4\}$ is the CNN (denoted as $CS_{nn}$) found on the TPR-tree when $r$ arrives, and $K$ is equal to 5. After invoking BCA, $C_4$ is selected to merge with $\{r\}$. As $C_4$ only contains four queries after merging, cluster refinement is conducted. Specifically, each query $o$ in $\{C_1, C_2, C_3\}$ is checked to see if it can be inserted into $c_{min}$. Hence, in this example, A is found and is moved from $C_1$ to $C_4$. Consequently, the centroid of $C_1$ is updated and $C_4$ is removed from the TPR-tree. Finally, $C_4 \cup \{A\}$ is returned as a valid cloaking set.

# 5. DISTORTION AND PRIVACY VERIFICATIONS

The proposed cloaking algorithms, namely, GCA, BCA and HCA, involve three main steps: boundary query computing, $\delta_q$-distortion and $\delta_p$-privacy verifications. We will elaborate them in Section 5.1, Section 5.2 and Section 5.3, respectively.

## 5.1 Boundary Query Computing

As discussed previously, each cluster is associated with a queue BTQ[2], which maintains the boundary queries at different timestamps. Each item in BTQ is in the form of <*time*, *query*>, where *time* is the timestamp when this *query* becomes boundary. Inside BTQ, the boundary queries are kept sorted in ascending order of the timestamp. As queries are changing along with the movement of its issuers, it is costly to track the boundary queries online.

Figure 8 shows five queries projected on the x-dimension. For the timestamps $t_i \sim t_j$, each query at the timestamp $t$ $(t < t_j)$ can be located by

$$x = x_{t_i} + v_x * (t - t_i) \tag{6}$$

Hence, the crossed points of lines in Figure 8 can be easily computed. Note that we only need to compute those crossed points which can contribute to the width of the boundary's segments. In Figure 8, the crossed point $P$ can be ignored. Although we take the case on the x-dimension as an example, the case on the y-dimension can be handled similarly. For every cluster $C$, let $VS+/VS-$ be the velocity sets of boundary queries on positive/negative x-dimension during its valid period. The main idea of boundary query computing is as follows: when a query $r$ is inserted into $C$, if $\forall v_+ \in VS+$, $r.v_x < v_+$, and $\forall v_- \in VS-$, $r.v_x > v_-$, $r$ is impossible to be the boundary on the x-dimension. Otherwise, the timestamp when $r$ becomes the boundary is computed by using equation (6). In addition, those crossed points are inserted into BTQ. For the example in Figure 8, if $Q_3$ is the query to be inserted, as $Q_5$'s velocity is larger than $Q_3$'s, it would be picked up to compute the crossed point $P'$. In this way, all boundary queries of a cluster can be calculated. Due to space limitations, we omit the detailed algorithm.

## 5.2 $\delta_q$-Distortion Verification

By maintaining BTQ, it is easy to get any boundary query at any time for a cluster. Therefore, during two consecutive timestamps $[t_i, t_{i+1}]$ in BTQ, as each boundary movement is a linear function with timestamp $t$, $P_{L,t}$ and $P_{v,t}$ can be computed. To satisfy $\delta_q$-distortion, for any timestamp $t \in [t_i, t_{i+1}]$, the following inequation should be held:

$$\frac{1}{P_A}[P_{L,t_i} + P_{v,t_i}(t - t_i)] < \delta_q. \tag{7}$$

By setting the left side of the inequation (7) to be $\delta_q$, we can compute the upper bound of $t$, denoted as $t^+$. If $t^+$ locates in $[t_i, t_{i+1}]$, it is easy to know that $\delta_q$-distortion is violated. Otherwise, $\delta_q$-distortion is satisfied. The algorithm is quite straightforward and thus is omitted here.

## 5.3 $\delta_p$-Privacy Verification

Like $\delta_q$-distortion, $\delta_p$-privacy can be verified during each time interval between two consecutive timestamps in BTQ by using equations (2) and (3). However, it is not necessary

[2]In the implementation, there are four queues, namely, $BTQ_{x-}, BTQ_{x+}, BTQ_{y-}, BTQ_{y+}$, which maintain the boundary queries on every direction of the cluster MBR. For ease of presentation, we unify them as a queue BTQ.

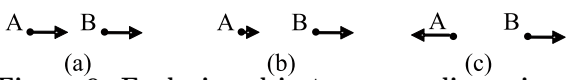


**Figure 9: Exclusive objects on one dimension**

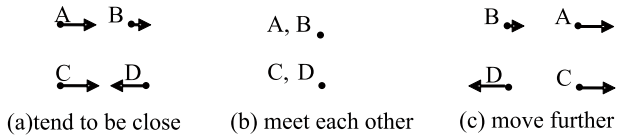


**Figure 10: Non-exclusive objects on one dimension**

for $\delta_p$-privacy to be checked during every time interval of the valid period.

Let's define *exclusive* first:

*Definition 13.* (Exclusive) Let *dist* be the distance between two queries $r_1$ and $r_2$. For any two timestamps $t_i$, $t_j$, $t_i < t_j$, if $\text{dist}(r_1, r_2, t_i) \leq \text{dist}(r_1, r_2, t_j)$, we say these two queries are exclusive to each other.

Intuitively, the distance between two exclusive queries would increase as the time elapses. Moreover, two observations can be drawn. First, if two boundary queries on each dimension are exclusive and $WB/HB$ is larger than $\delta_p$ at $t_i$, it will not be less than $\delta_p$ after $t_i$. Second, for any two boundary queries, even if they are not exclusive to each other at the current timestamp, they will become exclusive at some future time. Figure 9 shows the cases when two queries are exclusive. (a) Two queries have the same velocity, thus the distance between them remains constant. (b) The velocities of two queries A and B have the same direction, but B, which is in front of A on the moving direction, has a larger velocity. Apparently, (a) is a special case of (b). (c) Two queries move on the opposite direction. Figure 10 demonstrates the cases when two queries are not exclusive at the early stage but become exclusive to each other after a certain timestamp. As shown, the queries A and B (C and D) are not exclusive in (a). However, as time goes by, after A and B converge in (b), they become exclusive to each other (e.g., in (c)).

The main idea of $\delta_p$-privacy verification is as follows. If the boundary queries of a cluster on both dimensions are exclusive, the actions are subject to different cases: when both $WB$ and $HB$ are larger than $\delta_p$, the cluster can be returned as a cloaking set directly; when $WB$ or $HB$ is less than $\delta_p$, we can delay to the time to publish the cloaking set, until both $|WB|$ and $|HB|$ are larger than $\delta_p$. If the boundary queries on any dimension are not exclusive, their information are kept on tracking in the BTQ until they are exclusive or all boundary queries in BTQ have been checked. The former terminal condition implies this cluster is a successful cloaking set, while the latter one indicates this cluster should remain in the service space and wait for anonymization. Due to space limitations, we omit the specific algorithm here.

Figure 6(a) shows an example when the cloaking set needs to be delayed for publishing. Assume $Q_1 \sim Q_3$ constitute a cluster at $t_s(K{=}3)$. As shown, $WB$ shrinks to zero at $T_w$, and it increases to $\delta_p$ at $T_\Delta$. Therefore, this cluster would wait until $T_\Delta$ is to be published as a cloaking set.

# 6. PERFORMANCE EVALUATION

In this section, the effectiveness and efficiency of our proposed algorithms, including GCA, BCA, and HCA, are experimentally evaluated under various system settings. Although the privacy technique proposed in [4] is the most representative approach for continuous queries, as discussed in Section 2, it suffers from disclosing the location privacy in some cases. Moreover, none of the existing cloaking algorithms consider the temporal utility of the cloaking region during the anonymization. Hence, we do not include any existing cloaking algorithm for comparison. The evaluation metrics include the cloaking success rate, the cloaking cost, the cloaking time, and the processing time for successful queries.

**Table 1: Default system settings**

| Parameters | Default values |
|---|---|
| Number of queries | 10,000 |
| Valid period | Randomly chosen from [0,1440] |
| Privacy level $K$ | 10 |
| $\delta_p$ | 1% of $\min(A_{width}, A_{height})$ |
| $\delta_q$ | 10% of the space |
| $\delta_T$ | $60s$ |
| Perimeter of road map | $\sim$6,000$km$ |

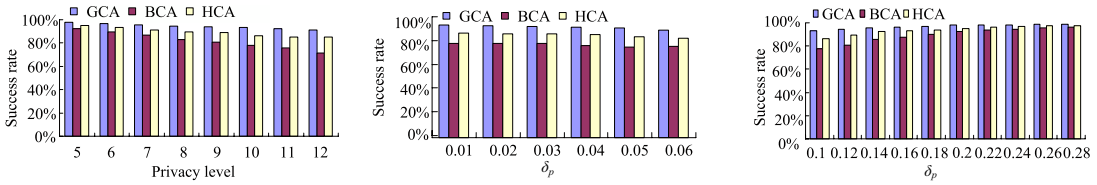
We use the well-known Thomas Brinkhoff Network-based Generator [3] to generate the moving objects in the system. The input of the generator is the road map of Oldenburg County (with perimeter around 6,000$km$). Our algorithms are implemented in Java and evaluated on a desktop running Windows XP SP2 with an Intel 2.0GHz CPU and 2GB main memory. A total of 10,000 moving objects are generated at the beginning of the simulation. Each object issues a continuous query, whose valid period is a random number in the range of [0, 1440]. Meanwhile, we assume a new query is not issued until the last query is successfully cloaked or expired. By default, for each query, the privacy level $K$ is set to 10, $\delta_p$ is set to 1% of the $\min(A_{width}, A_{height})$, $\delta_q$ is set to 10% of the system service space, and $\delta_T$ is set to $60s$. We summarize the default parameter settings in Table 1.

## 6.1 Cloaking Success Rate

In this section, the average cloaking success rates of GCA, BCA and HCA are evaluated under various settings of $K$, $\delta_p$ and $\delta_q$.

Increasing $K$ implies the privacy requirement becomes more constrained, which indicates more queries should be covered by a cloaking set. From Figure 11(a), we can observe that the success rate decreases when $K$ increases. Among the proposed algorithms, GCA is the best and its success rate decreases slightly with $K$ increasing. By contrast, BCA is the worst. That is because every query in GCA finds its cloaking set greedily in the entire *query set*, while the cloaking set is found among *cluster sets* in BCA. As explained in Section 4.3, BCA omits some possible cluster merges, and hence a number of cloaking sets cannot be successfully created. For HCA, it successfully resolves the problem of BCA by introducing a post-step to refine the cluster. However, as it insists to do the cluster merges on the granularity of clusters in the pre-step, some possible merges are still missing, which harms its cloaking success rate. Therefore, the success rate of HCA is between that of BCA and that of GCA.

Increasing $\delta_p$ also implies a higher privacy requirement. Figure 11(b) shows that the cloaking success rate slightly decreases, with $\delta_p$ increasing. This indicates that $\delta_p$ has little impact on the success rate. On the other side, increasing $\delta_q$ implies the requirement for QoS is relaxed. Figure 11(c) shows that the cloaking success rate increases when $\delta_q$ increases. The success rate of BCA increases obviously, as




(a) Different privacy levels (b) Different $\delta_p$ (c) Different $\delta_q$

**Figure 11: Cloaking success rate**

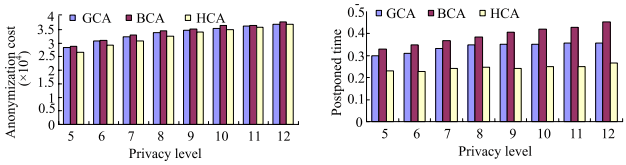
more clusters can be cloaked together. Compared with BCA, both the success rates of GCA and HCA increase slowly, indicating that $\delta_q$ has smaller impact on their success rates.

## 6.2 Cloaking Cost

This section evaluates the average cloaking cost under different settings of privacy level $K$, $\delta_p$ and $\delta_q$. The cloaking cost concerned in this paper can be divided into two parts: the anonymization cost and the postponed time.

The anonymization cost is the average perimeter of the cloaking region, which indicates the distortion of location information. As shown in Figure 12(a), for all cloaking algorithms, the average anonymization cost increases when the privacy level $K$ increases. This is as expected because each cloaking set is required to embrace more queries so as to meet a higher privacy requirement. GCA has a lower anonymization cost than BCA, because the cloaking region is built by adding queries one by one in GCA, while in BCA, the basic incremental unit is a cluster rather than a query. HCA has the lowest anonymization cost among three algorithms. This can be explained as follows. Different from GCA, HCA finds the cloaking set for each newly arrived query from its CNN clusters and thus it can guarantee that those residing queries of the cloaking set are close to this new query. As a result, HCA would have a lower anonymization cost than GCA. In addition, though both HCA and BCA seek for the cloaking set from CNN clusters, as there is a post-step to refine the clusters in HCA, BCA has a relatively higher anoymization cost than HCA.

Postponed time evaluates the cost of postponing $T_{exp}$ when forming cloaking sets. As the expiration time of a cloaking set is the largest $T_{exp}$ among all residing queries, queries whose $T_{exp}$ are prolonged are regarded as dummies after they expire. Postponed time is defined as the ratio of the prolonged time over the valid period for each successful query. As shown in Figure 12(b), the postponed time increases as $K$ increases. The rationale behind is that, as more queries are covered by a cloaking set, the expiration time of the cloaking sets becomes larger and therefore the queries can be prolonged for a longer period. As can be observed from Figure 12(b), HCA increases very slightly, and it has the smallest postponed time among the proposed algorithms. This is due to the benefits of the cluster refinement, which enables each cluster to have more queries and thus accelerate the procedure of forming the cloaking sets. In addition, its processing time is relatively longer (see Figure 15(b)), therefore, every query would have more chances to be clustered with those queries which hold a similar expiration time. As can be observed from the figure, GCA has a smaller postponed time, when comparing with BCA. The reason is twofolded. First, same as HCA, GCA has a longer processing time, which provides more chances for those queries with similar expired time to be anonymized together. Second, for BCA, as each query becomes fixed in a cluster once it is inserted, queries within a cluster cannot be merged with the queries of other clusters, even if they have the same expired time.

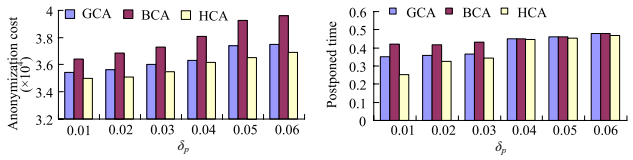



(a) Anonymization cost (b) Postponed time

**Figure 12: Different privacy levels $K$**

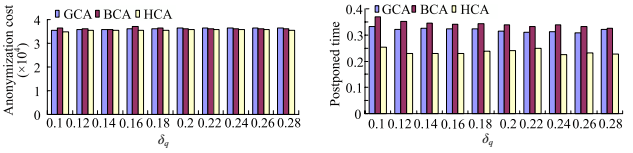
Along with the increment of $\delta_p$, the lower bound of the width/height of the cloaking boundary grows larger, which makes the newly arrived query to find farther queries for cloaking. As a result, the anonymization cost increases with $\delta_p$ increasing, just as shown in Figure 13(a). From Figure 13(b), we observe that each successful query has to be postponed for a longer period when $\delta_p$ grows larger. When $\delta_p$ is smaller (i.e., 0.01∼0.03), HCA is the best, and BCA is the worst. When $\delta_p$ grows beyond 0.03, as each query needs to be clustered with much farther queries for cloaking, their postponed time increases to a half of their valid periods. Meanwhile, their difference on the postponed time is overshadowed.

As shown in Figure 14(a) and Figure 14(b), $\delta_q$ has little effect on both of the anonymization cost and the postponed time. They are stable when $\delta_q$ increases.




(a) Anonymization cost (b) Postponed time

**Figure 13: Different $\delta_p$**




(a) Anonymization cost (b) Postponed time

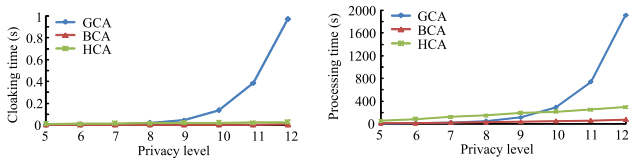
**Figure 14: Different $\delta_q$**

## 6.3 Cloaking Time and Processing Time

The average cloaking time and the processing time are evaluated in this section. The cloaking time of a query is the elapsed time between the moment when the query is received and the moment when it is successfully cloaked. It includes the computational time for maintaining the data structure (e.g., TPR-tree in HCA), boundary query computing, $\delta_q$-distortion verification and $\delta_p$-privacy verification.

From Figure 15(a), we can see that cloaking time increases as $K$ increases. This can be explained: the bigger is the value of the privacy level $K$, the more is the queries of each cloaking set. As a result, a longer cloaking time is needed. Among the proposed algorithms, BCA has the shortest cloaking time. This is because BCA finds the cloaking sets from the existing continuous clusters directly, and incurs little overhead for data structure maintenance. We can also observe that, GCA performs better than HCA when $K$ is small (e.g., $< 9$). The reason is that, a query can easily find its cloaking set under a small value of $K$, thus few queries are maintained in the service space to wait for anonymization which weakens the advantage of TPR-tree. Meanwhile, HCA has the overhead to maintain TPR-tree and clusters. Nonetheless, when $K$ grows larger (e.g., $\geq 9$), the cloaking time of GCA increases exponentially, while the advantage of TPR-tree in HCA becomes obvious. Consequently, HCA outperforms GCA after the value of $K$ reaches 9. Note that such performance gap becomes bigger with the increment of $K$.

The processing time includes the cloaking time and the time waiting for cloaking. As shown in Figure 15(b), the waiting time dominates the overall processing time, and the average processing time increases when $K$ increases. Based on the results shown in the figure, we can have the following observations: (1) BCA has the shortest processing time; (2) when $K$ is small (e.g., $< 6$), GCA requires less processing time than HCA. However, when $K$ grows larger (e.g., $\geq 6$), the processing time of GCA increases exponentially, and the performance improvement of HCA over GCA becomes much larger. The rationales behind these observations are similar as described in the last paragraph.




(a) Cloaking time (b) Processing time

**Figure 15: Different privacy level $K$**

# 7. CONCLUSIONS

In this paper, we investigated utility-based cloaking algorithms which protect both location privacy and query privacy for continuous queries. We observed that most of the existing location cloaking algorithms cannot effectively prevent from privacy disclosure or poor QoS for continuous queries. To address this problem, we proposed a greedy cloaking algorithm (GCA) and two incremental utility-based cloaking algorithms, called *bottom-up cloaking* (BCA) and *hybrid cloaking* (HCA). A series of experiments has been conducted to evaluate these algorithms under various system settings. Experimental results show that, GCA has the highest success rate, but suffers from a long cloaking time especially when the privacy level is high; BCA has the best efficiency, but its anonymization cost, cloaking success ratio and postponed time are relatively worse; HCA achieves the best overall performance in terms of various performance metrics.

# 8. ACKNOWLEDGMENTS

This research was partially supported by the grants from China 863 High-Tech Program(No:2007AA01Z155, 2009AA011904);the Natural Science Foundation of China under grant number 60833005. Jianliang Xu's work was supported in part by the Research Grants Council, Hong Kong SAR, China (Project Nos. HKBU211206 and HKBU211307).

We would like to thank Xing Hao of Renmin University of China for her valuable help in the experiments.

# 9. REFERENCES

[1] B. Bamba and L. Liu. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of International Conference on World Wide Web (WWW)*, 2008.

[2] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal*, 15(3):229–249, September 2006.

[3] T. Brinkhoff. Thomas brinkhoff network-based generator of moving objects, 2008.

[4] C. Chow and M. F. Mokbel. Enabling privacy continuous queries for revealed user locations. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, 2007.

[5] G. Ghinita, P. Kalnis, A. Khoshgozaran, and et al. Private queries in location based services: anonymizers are not necessary. In *Proceedings of SIGMOD*, 2008.

[6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[7] H. Hu and J. Xu. Non-exposure location anonymity. In *Proceedings of ICDE*, 2009.

[8] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *Proceedings of ICPS*, 2005.

[9] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proceedings of SIGMOD*, 2006.

[10] Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proceedings of the tenth ACM international conference on Knowledge discovery and data mining(SIGKDD)*, pages 617–622, 2004.

[11] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of VLDB*, 2006.

[12] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[13] J. Xu, X. Tang, H. Hu, and J. Du. Privacy-conscious location-based queries in mobile environments. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2009, to appear.

[14] J. Xu, W. Wang, J. Pei, and et al. Utility-based anonymization using local recoding. In *Proceedings of KDD*, 2006.

[15] T. Xu and Y. Cai. Location anonymity in continuous location based services. In *Proceedings of GIS*, 2007.

The Third SIGMOD PhD Workshop on Innovative Database Research
(IDAR2009), June 28, 2009, Providence, USA

# Complex Event Detection in Pervasive Computing

Chunjie Zhou, Xiaofeng Meng
School of Information, Renmin University of China, Beijing, China
{lucyzcj, xfmeng} @ ruc.edu.cn

## ABSTRACT

In pervasive computing environments, wide deployment of sensor devices has generated an unprecedented volume of atomic events. However, most applications such as healthcare, surveillance and facility management, as well as environmental monitoring require such events to be filtered and correlated for complex event detection. Therefore how to extract interesting, useful and complex events from low-level atomic events is becoming more and more important in daily life. Due to the increasing importance of complex event detection, this paper proposes a framework of Complex Event Detection and Operation (CEDO) in pervasive computing. It gives an event model and extends current detection by incorporating temporal and spatial settings of events and different levels of granularity for event representation. We first show research issues, related works, and main research problems in this area. Then our current research works and the preliminary results are introduced. Finally, the research plan of my PhD project is presented for discussion.

## Keywords

pervasive computing, atomic events, complex events, complex event detection

## 1. INTRODUCTION

In pervasive computing environments, sensors are deployed in everything from IT networks to enterprise software systems and physical world devices (through RFID readers, bar code scanners, manufacturing equipment sensors, and others). As these systems continue to proliferate, they generate events at a growing rate. Usually there are thousands of data records in a normal sensor device, which make it difficult for operators to find exceptional events by checking every record. While operators should find the relative records timely when analyze the exceptional events afterwards. However, the traditional detection methods are short of intelligent analysis and the data records are unable to be indexed efficiently. People must search artificially according to the rough time interval, so the data analysis waste a lot of time and energy.

In order to solve these problems, the efficient method is to do intelligent analysis on events atomically, and extract the centralized and interesting events timely. So it can give an alarm in time and index data efficiently based on the stored event information. For a concrete example, in a retail store, a occurrences and non-occurrences of events, and impose temporal constraints over these events. When there is a scenario where an item was picked up from a shelf and then taken out of the store without being checked out, the system could give an alarm atomically. Today, significant improvements in operational business decisions await those organizations to capture and process these events into meaningful business insight.

At present many applications need to extract complex events (often user-specified) from these flows of low-level atomic events. Such applications include supply chain management, financial services [1], business activity monitoring, elder care [2], and various pervasive computing applications. In it, the applications of indoor environment include: 1) checking "Whether the patient has already been taken care of" which contains a series of checks "Did the patient take his medicine?" "Did he have his lunch?" "Was his symptom normal or not?" and so on. 2) The security system might decide whether to take some precautionary action by comparing the complex events at the same time of different days. The applications of outdoor environment include: in the airport, station, or district, it can be used to detect and follow the people, the vehicles, or other suspicious objects. It also can be used to judge whether there are exceptional actions of people or vehicles in the restricted area.

All the above applications require such events to be filtered and correlated for complex event detection and transformed to new events that reach a semantic level appropriate for end applications [3]. These requirements need to perform real time translation of data describing a physical world into information useful to end applications. So this paper proposes a framework of Complex Event Detection and Operation (CEDO), which provides a rich, declarative environment for the development of event processing applications that may process and act on thousands of events per second. CEDO will be integrated into standard middleware architectures and be embedded in any standard enterprise application. It can be deployed as a stand-alone offering on third party application severs or as an integrated service engine. The main contribution of the framework is to address modeling, representation, and detection of events, where the focus is to detect about events rather than about the changes in objects' states.

The remainder of the paper is organized as follows. Section 2 introduces the related works. The event model and our framework of CEDO are shown in section 3. Section 4 presents the research plan of my PhD project.

## 2. RELATED WORKS

In pervasive computing environments, events distribute in nodes scattered, some of which are mobile nodes. If we use a central node to detect atomic events and form complex event expressions, this node will become the bottleneck of event detection. Therefore, in order to detect events effectively, we should choose the appropriate detection method according to


*Proceedings of the Third SIGMOD PhD Workshop on Innovative Database Research (IDAR 2009), June 28, 2009, Providence, RI, USA*

characteristics of pervasive computing environments and system requirements. The current existing complex event detection methods include: (1) based on the event tree [5] complex event detection; (2) based on the diagram detection method [6]; (3) based on automata [7] complex event detection; (4) based on Petri nets [4] complex event detection; (5) pipeline operation [8] detection methods. The above complex event detection methods all have their own advantages and weak points: GEM [5] considers the delay between events occurrence and detection, and handles events disorder by assigning the biggest tolerant delay. But it assumes there is a perfect global synchronous clock, which is unsuitable for no-centralized management and distributed systems of clock drift and loose coupling. Due to the lack of consideration of unpredictable delay, it cannot make breaking and mobile detection in mobile database efficiently. Snoop [6] only provides the simple time model, in which every event is regarded as a certain time point. Atomic events are based on definitions, while complex events are based on semantic. This method is suitable for centralized system or LAN. ODE [7] uses Finite Automata to express events, which can express real-world events intuitively, establish automata and detect complex events. But pure automata can neither detect parameter-events nor express event-disorder, so it cannot meet requirements of distributed systems. In addition, the above complex event detection methods don't consider uncertainty at all, which is the essential characteristic of pervasive computing environments.

As mentioned above, the existing complex event detection methods all cannot satisfy the requirements of pervasive computing environments. Therefore, based on the characteristics of pervasive computing environments, we summarize the current research works mainly from the three characteristics of complex event detection. Current researches on complex event detection generally include the following aspects: from the angle of event-type, describing the representation of complex events; from the angle of time, describing all kinds of sequential representations; from the precision degree of data, analyzing and handling the probabilistic data. Current research works emphasize particularly on different aspects.

According to the above three characteristics of complex event detection, the existing research works can be classified and summarized as in figure 1. In it, three axes correspond to three characteristics: time (time point and time interval), data (precise and uncertain), and events (atomic and complex). Three axes divide the space into eight quadrants (as shown in figure 1), and each quadrant corresponds to different attribute values of the three characteristics.

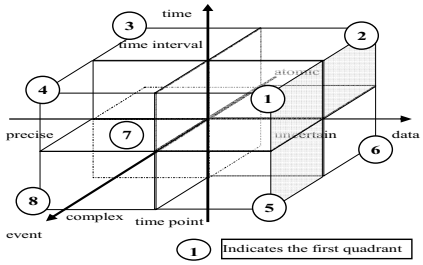



**Figure 1 the summary of current research works**

As in figure 1, the cube region in quadrant 7 stands for researches about precise atomic events at time point. At present, there are many research works about it [9]. The cube regions in quadrant 3, 4, 6 and 8 indicate separately the precise atomic events in time interval, the precise complex events in time interval, the uncertain atomic events at time point, and the precise complex events at time point. There are some research works about them [10, 8, 11, 12, 13, 14 and 20]. The cube regions in quadrant 1, 2, 5 are mainly about uncertain data, including the uncertain atomic events at time point, the uncertain complex events in time interval, and the uncertain complex events at time point. As far as we know, there is merely related research works about them. As shown in table 1, in quadrants 3, 4, 6 and 8, [9] is about time point, atomic events and precise data; [10, 8] is about time point, complex events and uncertain data; [11, 12 and 19] is about time interval, atomic events and precise data; [13] is about time interval, complex events and precise data; [14] is about uncertain data.

**Table 1 the comparison of current research works**

| Research works | Time interval | Complex events | Uncertain data |
|---|---|---|---|
| [9] | No | No | No |
| [10,8] | No | Yes | No |
| [11, 12, 19] | Yes | No | No |
| [13] | Yes | Yes | No |
| [14] | No | No | Yes |

# 3. FRAMEWORK

In this section, we present the framework of CEDO, and illustrate how this framework can be used to support application requirements. Before that, we give some preliminaries and an event model, which are the basis of the following.

## 3.1 Preliminaries

Events are defined as something that users are interested in. Events are happening all around us all the time. Detection of a person in a room, the firing of a CPU timer, and a Denial of Service (DoS) attack in a network are example events from various application domains. All events signify certain activities; however their complexities can be significantly different. For instance, the firing of a timer is instantaneous and simple to detect, whereas the detection of a DoS attack is an involved process that requires computation over many simpler events. So events can be divided into two types: atomic events and complex events [10]. In the following we will give definitions of them.

**Definition 1 (atomic event):** An atomic event is defined as a thing that happens instantaneously at a specific time point. It can be expressed as $E_{atomic_i} = Action < o_i, p_i, t_i >$. In it, $o_i$ stands for certain object; $p_i$ indicates some place, which is the current location of object $o_i$; $t_i$ expresses a certain time point; *Action* means the activity of object $o_i$ at time $t_i$ in place $p_i$. An atomic event $E_{atomic_i}$ corresponds to something in the physical world. For example, *Coffee ('Mary', 'Room 301', 10:00am)* is an atomic event, which means "Mary is getting coffee in Room 301 at 10:00am".

**Definition 2 (complex event):** A complex event often happens in a continuous time interval, which is assigned by users (called case 1) or abstract directly from atomic events (called case 2). It can be expressed as $E_{complex_i} =< Q_i, E_i, T_i >$. In it $Q_i$ stands for a certain query, which is only useful in case 1 and be used to indicate the query condition, while in case 2, its value is null; $E_i$ expresses the set of atomic events, which is $E_i = \{1 \le i \le n \mid E_{atomic_i}\}$. The atomic events in the set are connected and some relational operators exist among them (such as positive correlation, negative correlation, parallelism, serial, and so on); $T_i$ means a time interval. Querying or abstracting on a series of correlated atomic events in a time interval $T_i$ is the process of getting complex events. For example, "Mary is getting

coffee" can be extracted from a series of atomic events "Mary is in her office", "Mary is in coffee room", "Mary is in her office", and so on. Complex events are generated by composing atomic or other complex events using a set of event detection operators.

## 3.2 Event Model

In this paper, we consider events in the context of spatio-temporal databases. As introduced in section 3.1, our model includes atomic events and complex events. Here we use a model like data cube which is a three- (or higher) dimensional array of values. The three dimensions are separately object, time and place. As in figure 2, a snapshot of the model taken at time *i* contains all objects' current positions. For simplicity, we call each such snapshot a *world W* and it can be expressed as $W=\{1 \le i \le n \mid < o_i, p_i >\}$ (the meaning of $o_i$ and $p_i$ can be found in section 3.1). A *stream* shows the same place where different people are in at different time. A *flow* of objects is a set of places in which the object is at distinct timestamps. An *event database* consists of several flows of objects in the time interval T. We call each such event database a complex event, and denote it as a sequence of sets of tuples: $E_{complex} = (E_1, \cdots, E_n, T_i)$ where $E_i = \{1 \le i \le n \mid E_{atomic_i}\}$ (see section 3.1). The start time, the end time, and the duration of complex events can be showed in the time dimension. In addition, the choice of granularity for time dimension is very important. When the granularity of time is too small (e.g., milliseconds), an event query such as "What did he do during the first three days of May, 2008?" lead to a massive amount of uncertain (3×24×60×60×1000) possibilities.

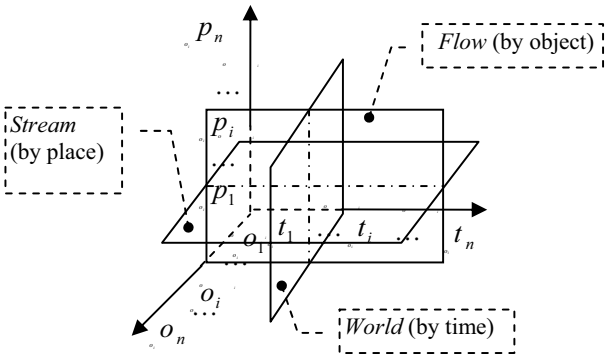



**Figure 2 the event model**

## 3.3 Complex Event Detection and Operation

Our framework has two sub-processes: complex event detection and complex event operation, which are the inverse procedures (as shown in figure 3). In the following, we will introduce every block of our framework.

**The terminal layer**

This is the source of raw data, including mobile devices, the smart phone, PDA, PC, and so on. Every created data has two elements: the data itself and a timestamp [15].

**The application layer**

The applications include healthcare, security monitoring, people tracker, and so on. These applications need high-level complex events oriented to clients.

**Smart device bus**

According to the two sub-processes, the smart device bus has two functions. In the process of complex events detection, the smart device bus is in charge of feeding raw data to CEDO; while in the process of complex events operation, it translates the message into the commands that can be recognized by a physical reader.

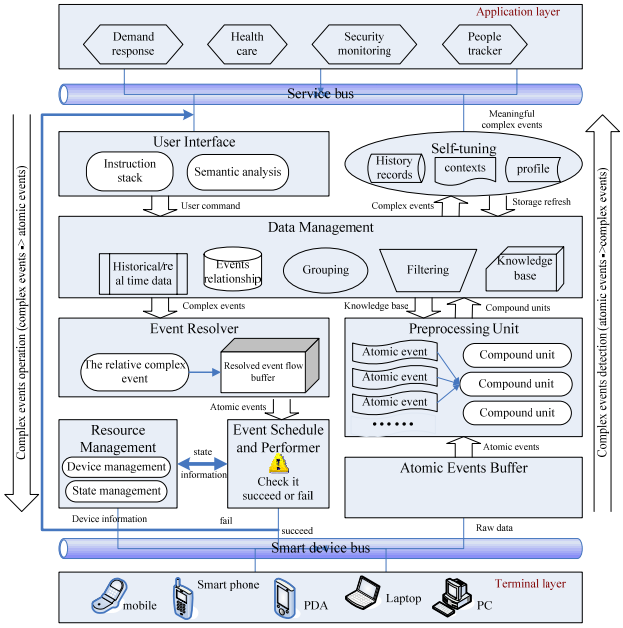



**Figure 3 the framework of CEDO**

### *3.3.1 Complex event detection*

The goal of complex event detection is to enable information contained in the events flowing through all of the layers of the framework to be discovered, understood in terms of its impact on high-level management goals and business processes, and acted upon in real time (as shown in the right side of figure 3). Here we consider as an example a database that handles typical behaviors of occupants in a smart home. Table 2 shows raw data collected from physical devices.

**Table 2 raw data stream (RDS)**

| RID | Obj | Time | Place | Probability |
|---|---|---|---|---|
| 1 | O1 | 6:40:00am | Kitchen | 1.0 |
| 2 | O1 | 6:40:01am | Bedroom | 1.0 |
| 3 | O1 | 6:50:00am | Kitchen | 1.0 |
| 4 | O1 | 7:00:20am | Kitchen | 1.0 |
| 5 | O1 | 7:10:00am | Dining-hall | 1.0 |
| 6 | O1 | 7:20:35am | Dining-hall | 1.0 |

Ps: RID (Record ID), Obj (Object).

**Atomic events buffer**

Many types of applications generate data streams as opposed to data sets. Managing and processing data for these types of applications involves building buffer storage and forming atomic events with a strong temporal focus. Atomic events outputted from the buffer have context, that is, timing (when it happened, both in absolute terms and relative to other events), sequence, and linking relationships to other events.

In the above example, the raw data are inputted to "atomic events buffer" through "smart device bus". After a certain time interval, some atomic events are outputted from "atomic event buffer", which are shown in Table 3.

**Table 3 atomic events (AE)**

| RID | Obj | Time | Place | Probability |
|---|---|---|---|---|
| 1 | O1 | 6:40:00am | Kitchen | 1.0 |

| | | | | |
|---|---|---|---|---|
| 2 | O1 | 6:40:01am | Bedroom | 1.0 |
| 3 | O1 | 6:40:02am~ 7:09:59am | Kitchen | < 1.0 |
| 4 | O1 | 7:10:00am~ 7:29:59am | Dining-hall | < 1.0 |

**Preprocessing unit**

CEDO looks at events in the context of other events rather than in isolation. So the preprocessing unit classifies large volume of atomic events into compound units, according to the incorporating temporal and spatial settings of the incoming events. In order to get meaningful compound units, a pattern matching capability is typically included in the preprocessing unit.

The above atomic events are inputted into "preprocessing unit" intermittently, and be classified into compound unit. This process often needs some extra information which is stored in "database management". In this example, the probability of record 2 is reduced based on the rules of spatial information (shown in table 6). O1 may be cooking or washing in the kitchen (based on the knowledge base in table 5), so we integrate the two atomic events into one compound unit. Because O1 went to the dining-room after that, we guess O1 was more likely cooking in the kitchen and the probability of record 1 is set 0.7 (as shown in table 4).

**Table 4 compound units (CU)**

| CU | RID | Obj | Time duration | Place | Action | Probability |
|---|---|---|---|---|---|---|
| 1 | 1 | O1 | 6:40:00am~ 7:09:59am | Kitchen | cooking | 0.75 |
| | 2 | O1 | 7:10:00am~ 7:29:59am | Dining hall | eating | |
| 2 | 3 | O1 | 6:40:00am~ 7:09:59am | Kitchen | washing | 0.24 |
| | 4 | O1 | 7:10:00am~ 7:29:59am | Dining hall | eating | |

**Data management**

The data management infrastructure of CEDO supports the notion of streams of structured data records together with stored relations. Many modern applications require long-running queries over continuous unbounded sets of data [18]. So there are two kinds of event records stored in the database. One is called "real-time event record", which is the processing of events as they arrive; the other is called "historical event record", which is the use of a sophisticated and optimized storage mechanism. CEDO is really an integration of historical event record coupled with real-time event record. A knowledge base also should be stored in the data management, which includes the extra information, such as the spatial location information, and the possible actions in certain place. Relations identify the relationships between incoming atomic events in CEDO. In CEDO the instantaneous relation is used to denote a relation in the traditional bag-of-tuples sense, and relation to denote a time-varying bag of tuples.

As in the above example, both data records and relationships are stored in the "database management" as shown in table 6. In addition, there should be a knowledge base used to store extra information (shown in table 5).

**Table 5 the knowledge base in database management**

| Place | Time interval | Action | Probability |
|---|---|---|---|
| Kitchen | 6:00am-7:30am or 11:00am-12:30am or 18:00pm-19:30pm | cooking | 0.75 |
| | | washing | 0.25 |
| Kitchen | 7:30am-8:00am or 12:30am-13:00pm or 19:30pm-20:30pm | cooking | 0.75 |
| | | washing | 0.25 |

**Table 6 rules in database management**

| Place1 | Place2 | Distance | Cost time |
|---|---|---|---|
| Kitchen | Bedroom | 20 meters | 15 seconds |

**Table 7 complex events chain (CEC)**

| CE Identity/Name | Sequence | Atomic Event | Probability |
|---|---|---|---|
| Dinner | 1 | Cooking | 0.75 |
| | 2 | Eating | |
| | 3 | Washing | |

**Self-tuning**

Filtering the complex events outputted from the data management according to the people' profile, the context, the historical records and so on. So the complex events reported to end users must be meaningful. The final results of self-tuning are also stored in the database.

In the above example, through "self-tuning", complex events in table 7 will become more and more precise, which are useful to end applications. The complex events chain can be shown intuitively in figure 4.

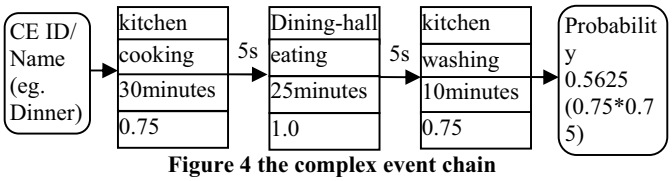



**Figure 4 the complex event chain**

### *3.3.2 Complex event operation*

The goal of complex event operation is to resolve complex events into a set of corresponding atomic events, which can be recognized and performed by physical devices through semantic analysis (as shown in the left side of figure 3).

**User interface**

CEDO provides a graphical user interface (GUI). It differs from traditional graphical user interface technologies in that they are designed to display and manipulate time-based information typically found in event processing systems.

**Event resolver**

Resolve the complex event into a set of relative atomic events, which are stored in a buffer temporarily. This is the inverse process of complex event detection.

**Resource management**

It includes device management and state management. Device management schedules physical devices and decides which device should be used. State management defines state situations of physical devices. Each physical device has four statuses: undefined, unrequest, request and active. "Undefined" means this kind of physical device does not exist; "unrequest" indicates there is such physical device but no event requests it; "request" implies there is at least one event that wants to use this physical device; "active" signifies this device is being used now.

**Event schedule and performer**

Decide which one in the resolved atomic events buffer should be performed first and check whether the performance conditions of the relative event are satisfied. If all relative physical devices are available, then the event performance succeeds and sends instructions to corresponding devices. Otherwise, it fails and returns the failing reason to the user.

## 4. RESEARCH WORKS

In my PhD project, I plan to focus on a few key problems in

CEDO. To make our work based on a strong foundation, we would like to fully implement CEDO and build the experimental platform of complex event detection in pervasive computing based on our framework. This section will introduce our main work on event probability, event disorder, and event relationship in CEDO.

**1) Event probability**

Uncertainty is one of the most important challenges of complex event detection (such as RFID data). However, there are many reasons for producing probabilistic data. For example: 1) conflicting readings, e.g. Alice is read by two adjacent antennas, what is her true location? [17]; 2) missed readings, e.g. readers commonly detect only about 60%-70% of tags in their vicinity [17]; 3) granularity mismatch, an application queries about offices, but the system only provides information about sensors.

Complex event detection on probabilistic data can be divided into two types: local uncertainty detection and global uncertainty detection. When event detection only concern with the uncertainty of the tuple / object itself, and are independent from other objects/ tuples, we call them local uncertainty detection. Let's think the example of getting coffee in section 3.1, when and where does Mary want to get coffee, and the time duration of getting coffee are all uncertain. As shown in table 8 (a), the time when Mary gets coffee may be at 10:15 am or at 9:55am; the place where she gets coffee may be in No.1 cafe or in No.2 café; the time duration may be 15 minutes or 17 minutes. But these factors are only based on Mary's own willing, and independent of others', so it is called local uncertainty.

On the other hand, when the event detection must consider the uncertainty of combinations of objects / tuples, we call such detection global event detection. We still take getting coffee as an example. Suppose that Mary likes to get coffee together with Joe, then the time, place and duration of getting coffee are not only based on Mary's own willing, but be decided by many uncertain factors, such as Joe's willing and the actions of others. As shown in table 8(b), in July, due to the influence of Joe, the time that Mary got coffee is earlier and the duration are shorter than in March. But their coffee time and duration in different dates are still uncertain. Generally, when whether an object / tuple satisfies a detection condition depends on other objects or tuples not involved in the same generation rule, global uncertainty has to be considered. Semantically, we have to examine the possible worlds one by one and count the probability that a combination of objects / tuples is an answer.

**Table 8(a) the event local uncertainty**

| Date | Name | Coffee time | Café ID | Time duration |
|---|---|---|---|---|
| March 1st | Mary | 10:15am | 1 | 15 minutes |
| March 2nd | Mary | 9:55am | 2 | 17 minutes |
| March 1st | Joe | 8:55am | 1 | 10 minutes |
| March 2nd | Joe | 9:05am | 1 | 8 minutes |

**Table 8(b) the event global uncertainty**

| Date | Name | Coffee time | Café ID | Time duration |
|---|---|---|---|---|
| July 1st | Mary | 9:35am | 1 | 12 minutes |
| July 2nd | Mary | 9:25am | 1 | 10 minutes |
| July 1st | Joe | 9:35am | 1 | 12 minutes |
| July 2nd | Joe | 9:25am | 1 | 10 minutes |

Most of the current researches on complex event detection suppose events are precise, however, they are imprecise in many real applications. Probability is the essential problem in pervasive computing, even in complex event detection. For example, in the application scenario of "smart home", the sensor data, the behavior pattern and customers of the occupants are all probabilistic. So how to extract meaningful and precise information according to these imprecise data is a challenge problem. Probability has become a hot research problem in resent years, but there are still many problems should be further researched in probabilistic complex event detection, such as the probabilistic sensor data, the local uncertainty detection and global uncertainty detection. With the appearance of large volume of probabilistic events, the probabilistic complex event detection will become more and more important and demand prompt solution.

**2) Event disorder**

The tuples in an event flow may or may not be in order by some desired attribute of those tuples. When such an ordering exists, some operations become easier and can be performed without the need for arbitrary storage; however, when this ordering is violated, this is called "event disorder." Poset processing consists of performing operations on a set of tuples that may not be related by a total ordering. Any partially ordered set of tuples can be processed in arbitrary ways within an event flow processing system by storing those tuples and retrieving as needed to match desired patterns. Most of current researches suppose events are ordering, that is to say, they don't consider the concurrent and overlapping events. However, in many real applications this assumption is unacceptable. Take the healthcare in section 1 for example; the atomic events (such as toothbrushing and taking temperature) may happen currently in the process of detecting complex events (eg. healthcare). In addition, atomic events and complex events in this process are possible disorder because of different habits of different people. Future research on complex event detection must take disorder events into account.

**3) Event relationship**

Current researches on complex event detection usually suppose events are isolate, but actually they have a thousand and one links. So in complex event detection, we must consider the relationships of the same object in different times, the interaction of different objects, and the factors of identity, position, and so on. Here we use an example in [14] to explain. The location of Joe at T=7 and T=8 are separately shown in Figure 5(a) and 5(b). [14] use sampled distributions produced by the particle filter to express the location probability. Each particle represents a guess about Joe's location and the locations are uncertain. Figure 5(c) shows the location of Joe and Sue are connected, that is to say, we can guess approximately the location of Sue according to Joe's location. In figure 5(c) the probabilities of Joe in H1 and in O2 are both 0.4 at T=7. If we know Sue is the secretary of Joe and they are almost together. We can guess Joe was more likely to be in O2 at T=7 based on the probability that Sue was in O2 at T=7 is 0.6. Figure 5(d) shows the location of Joe at T=7 and at T=8 are also connected. If Joe was in O2 at T=7, he was more likely to be still in O2 at T=8. However, current researches don't consider these connective factors.

In addition, most current research works only consider converting atomic events to complex events, few studies convert complex events to more complex ones. The input of the latter is the output results of the former, so the former research is an important step of more complex event detection. However, with wide applications in real world, more complex event detectionwill become increasingly important. Take the health-care in section 1 as an example, checking "Whether the patient has already been taken care of" contains a series of checks "Did the patient take his medicine?" "Did he have his lunch?" "Was his symptom normal

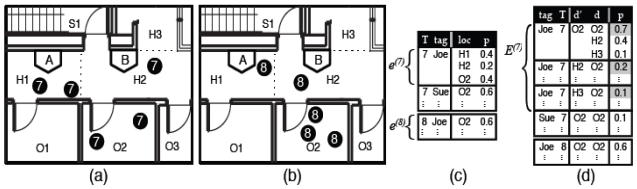



**Figure 5 the relationships of events**

or not?" and so on. In this example, we can regard the whole process of health-care as a more complex event. The actions involved in it can be atomic events, or complex events. For example, checking "Did the patient take his medicine?" is a complex event, because it includes the following atomic events: "pick up a cup of water ", "take up the medicine bottle", and "take water". While checking "whether the reading of blood pressure and body temperature is normal or not" are atomic events. We use figure 6 to express the ranked events intuitively.

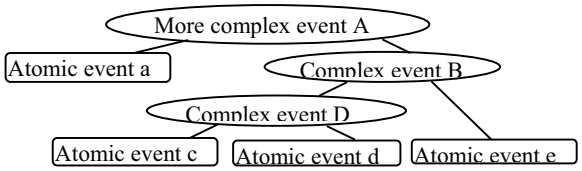



**Figure 6 the ranked complex events**

The specific solution approaches for the various challenges listed are our future research works, and maybe we will consider tree-based complex events.

# 5. CONCLUSION

This paper presents the sketch of my research plan on PhD project. Firstly, we summarize the current research status in this area. Then our framework of complex event detection and operation in pervasive computing is introduced. It gives an event model and extends current detection by incorporating temporal and spatial settings of events and different levels of granularity for event representation. Based on this framework, the unprecedented volume of atomic events can be filtered and correlated to get interesting, useful and complex events. In conclusion, the main works of my PhD project are to build a complex event detection system and to address several key issues in this area.

# 6. ACKNOWLEDGMENTS

This research was partially supported by the grants from the Natural Science Foundation of China (No: 60833005, 60573091); China 863 High-Tech Program (No: 2007AA01Z155); PhD Foundation Program of Education Ministry (No: 200800020002).

# 7. REFERENCES

[1] Demers, A. J., Gehrke, J., and Hong, M. 2006. Towards expressive publish/subscribe systems. In EDBT, pages 627-644.

[2] Liao, L., Patterson, D. J., Fox, D., and Kautz, H. A. 2007. Learning and inferring transportation routines. Artif. Intell, 171(5-6):311-331.

[3] Khoussainova, N., Balazinska, M., Suciu, D. 2008. Probabilistic Event Extraction from RFID Data. Proceedings of IEEE 24th International Conference on Data Engineering (ICDE 2008). 1480-1482

[4] Gatziu, S., Dittrich, K. R. 1993. Events in an active object-oriented database system. In Proceeding of the 1st Int'l Conference on Rules in Database Systems. 23-39

[5] Samani, M., Sloman, M., Gen, M. 1997. A generalized event monitoring language for distributed systems. IEEE / IOP/BCS Distributed Systems Engineering. 96-108

[6] Chakravarthy, S., Rasad, V. K., and Anwar, E. 1993. Anatomy of a composite event detector. Technical Report UF-CIS-TR-93-039[R]. Gainesville: University of Florida

[7] Gehani, N. H., Jagadish, H. V., Shmueli, O. 1992. Event specification in an active object oriented database. Proceedings .of the ACM SIGMOD International Conference on Management of Data. ACM Press. 81-90

[8] Wu, E., Diao, Y., Rizvi, S. 2006. High-performance complex event processing over streams. In SIGMOD06: Proceedings of the 2006 ACM SIGMOD. 407-418

[9] Agrawal, J., Diao, Y., Gyllstrom, D., and Immerman, N. 2008. Efficient Pattern Matching over Event Streams. International Conference on Management of Data Proceedings of the 2008 ACM SIGMOD international conference on Management of data. Vancouver, Canada. 147-160

[10] Akdere, M., Cetintemel, U., and Tatbul, N. 2008. Plan-based Complex Event Detection across Distributed Sources. Proceedings of the VLDB Endowment.Vol.1,Issue 1: 66-77

[11] Papapetrou, P., Kollios, G., Sclaroff, S., and Gunopulos, D. 2005. Discovering Frequent Arrangements of Temporal Intervals. ICDM Proceedings of the Fifth IEEE International Conference on Data Mining IEEE. 354 - 361

[12] Wu, S. Y., and Chen, Y. L. 2007. Mining Nonambiguous Temporal Patterns for Interval-based events. IEEE Transactions on Knowledge and Data Engineering. Volume 19, Issue 6: 742-758

[13] Patel, D., Hsu, W., and Lee, M. L. 2008. Mining Relationships Among Interval-based Events for Classification. Proceedings of International Conference on Management of Data Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 393-404

[14] Re, C., Letchner, J., Balazinksa, M., and Suciu, D. 2008. Event Queries on Correlated Probabilistic Streams. Proceedings of International Conference on Management of Data Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 715-728

[15] Complex Event Processing in the Real World, an Oracle White Paper, September 2007

[16] Romero, M., and Rodriguez, M. A. 2007. A Graph-Oriented Model and Query Language for Events. SeCoGIS 2007-International Workshop on Semantic and Conceptual Issues in Geographic Information Systems. 358-367

[17] Jeffery, S. 2006. Adaptive cleaning for RFID data streams. In Proc. of the 32nd VLDB Conference.

[18] Chen, Q., Li, Z. H., Liu, H. L. 2008. Optimizing Complex Event Processing over RFID Data Streams. Proceedings of IEEE 24th International Conference on Data Engineering (ICDE 2008). 1442-1444

[19] Kam, P. S., Ada, W. F. 2000. Discovering Temporal Patterns for Interval-based events. Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery.

Geoinformatica
DOI 10.1007/s10707-008-0052-5

# Update-efficient indexing of moving objects in road networks

**Jidong Chen · Xiaofeng Meng**

Received: 22 December 2006 / Revised: 1 April 2008 /
Accepted: 30 April 2008


**Abstract** Recent advances in wireless sensor networks and positioning technologies have boosted new applications that manage moving objects. In such applications, a dynamic index is often built to expedite evaluation of spatial queries. However, the development of efficient indexes is a challenge due to frequent object movement. In this paper, we propose a new update-efficient index method for moving objects in road networks. We introduce a dynamic data structure, called *adaptive unit*, to group neighboring objects with similar movement patterns. To reduce updates, an adaptive unit captures the movement bounds of the objects based on a prediction method, which considers road-network constraints and the stochastic traffic behavior. A spatial index (e.g., R-tree) for the road network is then built over the adaptive unit structures. Simulation experiments, carried on two different datasets, show that an adaptive-unit based index is efficient for both updating and querying performances.

**Keywords** Spatial-temporal databases · Moving objects · Index structure · Road networks

## 1 Introduction

Recent advances in wireless sensor networks and positioning technologies have enabled a variety of new applications such as traffic management, fleet management,

The corresponding author currently works at EMC Research China, Beijing, China.

J. Chen (✉) · X. Meng
School of Information, Renmin University of China, Beijing, China
e-mail: chen_jidong@emc.com

X. Meng
Key Laboratory of Data Engineering and Knowledge Engineering, MOE, Beijing, China
e-mail: xfmeng@ruc.edu.cn

and location-based services that manage continuously changing positions of moving objects [25], [27]. In such applications, a dynamic index is often built to expedite evaluation of spatial queries. However, existing dynamic index structures (e.g. B-tree and R-tree) suffer from poor performance due to the large overhead of keeping the index updated with the frequently changing position data. The development of efficient indexes to improve the update performance is an important challenge.

Current work on reducing the index updates of moving objects mainly contains three kinds of approaches. First, most efforts [9], [18], [19], [36] focus on the update optimization of the existing multi-dimensional index structures especially the adaptation and extension of the R-tree [12]. To avoid the multiple paths search operation in the R-tree during the top-down update, some recent works propose the bottom-up approach [18], [19] and memo-based [36] structure to reduce the updates of the R-tree. Another method [9] exploits the change-tolerant property of the index structure to reduce the number of updates that cross the minimized boundary rectangle (MBR) boundaries of the R-tree.

However, the indexes based on MBRs exhibit high concurrency overheads during node splitting, and each individual update is still costly. Therefore, some index methods based on a low-dimensional index structure (e.g. $B^+$-tree) are proposed [14], [37], which construct the second category of index methods. They combine the dimension reduction and linearization technique with a single $B^+$-tree to efficiently update the index structure.

The third kind of approaches use a prediction method with a time-parameterized function to reduce the index updates [27], [29], [32]. They describe a moving object's location by a linear function and the index is updated only when the parameters of the function change, for example, when the moving object changes its speed or direction. The MBRs of the index vary with the time as a function of the enclosed objects. However, it is hard for the linear prediction to reflect the movement in many real applications and therefore leads to a low prediction accuracy and frequent updates.

Though these index structures solve the problem of index updates to some extent, they are designed to index objects performing free movement in a two-dimensional space. We focus on the index update problem in real life environments, where the objects move within constrained networks, such as vehicles on roads. In such a setting, the spatial property of objects' movement is captured by the network and the static information can be separated from the dynamic information. Therefore, the spatial location of moving objects can be indexed by means of the road-network index structure. For example, moving objects can be accessed by each road segment which is indexed by the R-tree. Since the road network seldom changes and objects just move from one part to the other part of the network, the R-tree in this case remains fixed. Existing index work that handles network-constrained moving objects [3], [11], [25] is based on this feature. These works separate spatial and temporal components of the moving objects' trajectories and index the spatial aspect by the network with an R-tree. However, they are mostly concerned with the historical movements and therefore they do not consider the problem of index updates.

In this paper, we address the problem of efficient indexing of moving objects in road networks to support heavy loads of updates. We exploit the constraints of the network and the stochastic behavior of the real traffic to achieve both high updating

and querying efficiency. We introduce a dynamic data structure, called *adaptive unit* (AU for short) to group neighboring objects with similar movement patterns in the network. A spatial index (e.g., R-tree) for the road network is then built over the AUs to form the index scheme for moving objects in road networks. The index scheme optimizes the update performance for the following reasons. (1) An AU functions as a one-dimensional MBR in the TPR-tree [29], while it minimizes expanding and overlaps by considering more movement features. (2) The AU captures the movement bounds of the objects based on a prediction method, which considers the road-network constraints and stochastic traffic behavior. (3) Since the movement of objects is reduced to occur in one spatial dimension and attached to the network, the update of the index scheme is only restricted to the update of the AUs. Since the AU is approximated by its center object for efficiency, the query will possibly has false negative result. For improving it, we refine the prediction accuracy by simulating two trajectories based on different assumptions on the traffic conditions and revising the trajectory bounds when prediction accuracy decreases over time. We have carried out extensive experiments based on two datasets. The results show that an adaptive-unit based index not only improves the efficiency of each individual update but also reduces the number of index updates and is efficient for both updating and querying performance.

The main contributions of this paper are:

- The introduction of the graph of cellular automata (GCA) model and the simulation-based prediction (SP) model which capture traffic features and reduce the index updates.
- The introduction of AUs that optimize for frequent index updates and support the predictive query on moving objects in road networks.
- An experimental evaluation and validation of the efficient update as well as query performance of the proposed index structure.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 introduces our data model and trajectory prediction method. Section 4 describes the structure and algorithms of AUs for efficient updates and query processing. Section 5 contains algorithm analysis and experimental evaluation. We conclude and propose the future work in Section 6.

## 2 Related work

Many efforts have been made on reducing the need for index updates of moving objects. In summary, they can be classified into three categories.

First, most work focuses on updating optimization of existing multi-dimensional index structures especially in the adaptation and extension of the R-tree [12]. The top-down update of an R-tree is costly since it needs several paths for searching the right data item considering the MBR overlaps. In order to reduce the overhead, Kwon et al. [18] develop the Lazy Update R-tree, which is updated only when an object moves out of the corresponding MBR. With adding a secondary index on the R-tree, it can perform the update operation in a bottom-up way. Recently, by exploiting the change-tolerant property of the index structure, Cheng et al. [9] present the CTR-tree to maximize the opportunity for applying lazy updates

and reduce the number of updates that cross MBR boundaries. Lee et al. [19] extends the main idea of [18] and generalizes the bottom-up update approach. However, they are not suitable to the case where consecutive changes of objects are large. Xiong and Aref [36] present the RUM-tree that processes R-tree updates in a memo-based approach, which eliminates the need to delete the old data item during an index update. Therefore, its update performance is stable with respect to the changes between consecutive updates. In our index structure, however, the R-tree remains fixed since it indexes the road network and only the AUs are updated.

The second type of methods are based on the dimension reduction technique [25] and a low-dimensional index [14], [37] (e.g. $B^+$-tree). The $B^x$-tree [14], [37] combines the linearization technique with a single $B^+$-tree to efficiently update the index structure. It uses space filling curves and a pre-defined time interval to partition the representation of the locations of moving objects. This makes the $B^+$-tree capable of indexing the two-dimensional spatial locations of moving objects. Therefore, the cost of individual updating of index is reduced. However, the two-dimensional locations of objects are linearized by a space-filling curve and the time is also partitioned by a pre-defined time interval. Therefore, the $B^x$-tree imposes discrete representation and may not keep the precise values of location and time during the partitioning. For our setting, the two-dimensional spatial locations of moving objects can be reduced to the 1.5 dimensions [16] by the road network where objects move.

The techniques in the third category use a prediction method represented as the time-parameterized function to reduce the index updates [27], [29], [32]. They store the parameters of the function, e.g. the velocity and the starting position of an object, instead of the real positions. In this way, they update the index structure only when the parameters change (for example, the speed or the direction of a moving object changes). The Time-Parameterized R-tree (TPR-tree) [29] and its variants (e.g. TPR*-tree) [27], [32] are examples of this type of index structures. They all use a linear prediction model, which relates objects' positions as a linear function of the time. Actually, these methods also can support predictive queries that are usually processed by the dual transformation technique in some index methods [2], [23]. However, the linear prediction is hard to reflect the movement in many real application especially in traffic networks where vehicles change their velocities frequently. The frequent changes of the object's velocity will incur repeated updates of the index structure. Moreover, other prediction models with non-linear prediction proposed by Aggarwal et al. [1] using quadratic predictive functions and by Tao et al. [34] based on recursive motion functions for objects with unknown motion patterns improve the precision in predicting the location of each object, but they ignore the correlation of adjacent objects and may not reflect accurately some complex and stochastic traffic movement scenarios. Our techniques also fall into this category and apply an accurate prediction method when updating index structure by considering more transportation features. Our prediction method also can be applied to update policy of objects [8], a different research issue, which focuses on how to minimize the number of location updates sampled by sensors or GPS periodically from moving objects to the server database. However, for comparisons with the TPR-tree in our experiments, we use the same update policy and location representation of objects, but different prediction method in the index structure.

Compared to the TPR-tree, our technique will reduce the indexing update costs when the same amount of updated locations of one object are stored in a database in the server.

There is some work on data models, indexing and query processing for moving objects in road networks, which is also related to our work. Data models for network constrained moving objects have been a focus of recent study [10], [28], [35] because they form a foundation for data storage and query processing. Indexing techniques for objects moving in road networks also become a focus of study [3], [11], [17], [25]. Pfoser et al. [25] propose to convert a 3-dimensional problem into two sub-problems of lower dimensions through certain transformation of the networks and trajectories. Another approach, known as the fixed network R (FNR)-tree [11], separates spatial and temporal components of the trajectories and indexes time intervals that each moving object spends on a given network link. The MON-tree approach [3] further improves the performance of the FNR-tree by representing each edge by multiple line segments (i.e. polylines) instead of just one line segment. However, they all focus on the historical movement and cannot support frequent index updates. There are also other work [22], [24], [31], all based on 3-dimensional variations of R-trees [12] and R*-trees [4], to index the historical trajectory in Euclidean spaces. Their goal is to minimize storage and query cost, which does not consider the index update problem. Similar to our index structure, the IMORS method [17] focuses on reducing the number of index updates on a road network with the same idea of separating dynamic and static parts of an index structure. However, moving objects are indexed by a static small road sector blocks and may move to different sectors very soon, therefore their coordinates and bi-directional pointers to the road sector are likely to be updated frequently when their locations especially velocity have been changed. While the update performance can be improved by enlarging the length of a road sector, it may result in a degradation of the query processing performance. Instead, considering more traffic features, our AU index, a dynamic structure, groups objects having similar moving patterns and can dynamically adapt itself to cover the movement of the objects it contains by a more accurate prediction method. Therefore, it reduces more index updates both by road-network features and by a new prediction method. In addition, the AU index can also support the efficient predictive queries on road networks, which is not implemented in the IMORS method.

Query processing algorithms in spatial network databases have been developed using network distance [6], [13], [15], [26], [30]. Most of them only work for static data objects and do not monitor queries over moving objects in road networks. Mouratidis et al. [20] study the continuous monitoring of nearest neighbors in highly dynamic scenarios, where queries and data objects move frequently in the network. Similar to our work, they target frequent data updates to support the NN queries on moving objects in road networks. However, they store the network, objects and queries in three memory-resident data structures: a spatial index on the network edges, an edge table maintaining network and moving objects, and a query table with expansion tree for each query. To incrementally monitor the NN queries, only updates from objects falling in the expansion tree can alter the NN set of query. The expansion tree is based on a query point and used to facilitate handling of query movements while our AU structure is used to index the moving objects and predict their movement to reduce the index updates.

## 3 Data model and trajectory prediction

We use the GCA model we proposed in [7] to model the network and moving objects. A road network is modeled as a GCA, where the nodes of the graph represent road intersections and the edges represent road segments with no intersections. Each edge consists of a cellular automaton (CA), which is represented, in a discrete mode, as a finite sequence of cells. GCA is based on the work of [21] and adapts the CA model for free traffic to represent the objects movement in road networks. Figure 1 shows an example of a road network and its GCA model. Each node has a label which represents an intersection of the road network. The wide lines represent edges and each edge is treated as one CA connecting many cells.

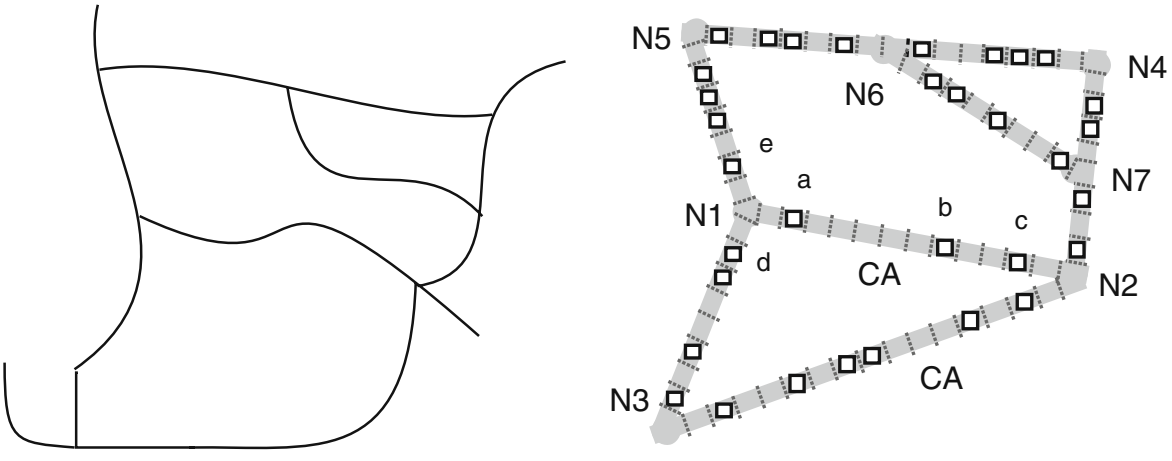
We first recall the definition of CA in this context.

**Definition 1** A CA consists of a finite oriented sequence of cells. In a configuration, each cell is either empty or contains a symbol. During a transition, symbols can move forward to subsequent cells, symbols can leave the CA and new symbols can enter the CA.

An example of CA corresponding to edge $(N_1, N_2)$ in Fig. 1b with a transition between two configurations is given in Fig. 2. We now formally define a GCA.

**Definition 2** The structure of a GCA is a directed weighted graph $G = (V, E, l)$ where $V$ is a set of vertices, $E$ is a set of edges and $l : E \to \mathbb{N}$ is a function which associates to each edge the number of cells of the corresponding CA.

We assume a countably infinite alphabet $\Omega : \{a, b, c, \cdots\}$, denoting moving object's names. Cells are denoted by the edge name and their indices in the edge. Let $C$ be the set of cells of a GCA. A configuration or an instance of a GCA, is a mapping from the cells of the GCA to constants in $\Omega$ together with a given velocity. Intuitively, the velocity is the number of cells an object can traverse during a time unit.




(a) A Road Network (b) An instance of GCA

**Fig. 1** An example of a road network and its GCA model

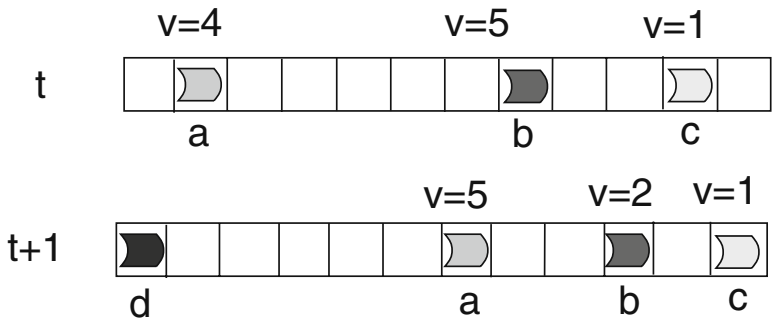



**Fig. 2** Transition of the GCA

**Definition 3** An instance $I$ of a GCA is defined by two functions of the following types:

$\mu : C \rightarrow \Omega \bigcup \{\varepsilon\}$ (1-1 mapping)

$v : \Omega \rightarrow \mathbb{N}$.

A moving object is represented as a symbol attached to a cell in the GCA and it can move several cells ahead at each time unit. A moving object lies on exactly one cell of an edge and its location can be obtained by computing the number of cells relative to the starting node. For instance, object $a$ lies on the edge $(N_1, N_2)$ and there are two cells away from $N_1$ along the edge. Therefore, its position can be expressed by $(N_1, N_2, 2)$.

The motion of an object is represented as some (time, location) information. Representing such information of a moving object as a trajectory is a typical approach [35]. In the GCA model, the trajectory of a moving object can be divided into two types: the in-edge trajectory for the object's movement in one edge (CA) and the global trajectory for the object that may move cross several edges (CAs) during its movement. The in-edge trajectory of an object is a polyline in a two-dimensional space (one-dimensional relative distance, plus time), which can be defined as follows:

**Definition 4** The in-edge trajectory of a moving object in a CA of length $L$ is a piecewise function $f : T \rightarrow \mathbb{N}$, represented as a sequence of points $(t_1, l_1), (t_2, l_2), \ldots, (t_n, l_n)(t_1 < t_2 < \ldots < t_n, l_1 < l_2 < \ldots < l_n \leq L)$, where $l_i$ is the relative distance to the starting node at the time of $t_i$.

When an object moves across multiple edges, its global trajectory is defined as functions mapping the time to the corresponding edge and the relative distance to the starting node.

**Definition 5** The global trajectory of a moving object in different CAs is a piecewise function $f : T \rightarrow (E, \mathbb{N})$, represented as a sequence of points $(t_1, e_1, l_1), \ldots, (t_i, e_j, l_k), \ldots, (t_z, e_m, l_n)(t_1 < t_2 < \ldots < t_z)$.

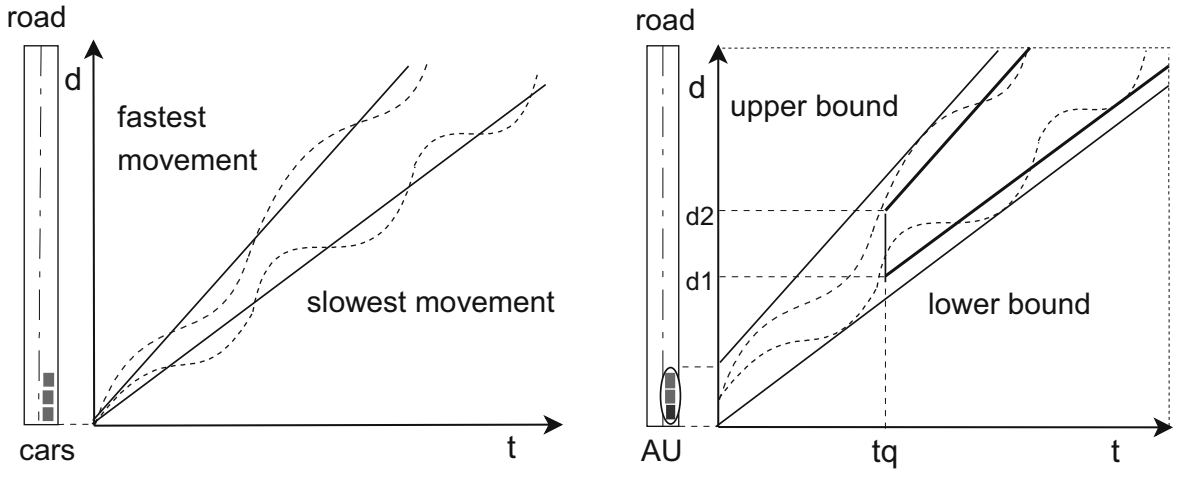
Let $i$ be an object moving along an edge. Let $v(i)$ be its velocity, $x(i)$ its position, $gap(i)$ the number of empty cells ahead (forward gap), and $P_d(i)$ a randomized slowdown rate which specifies the probability it slows down. We assume that $V_{max}$ is the maximum velocity of moving objects. The position and velocity of each object may change at each transition as shown in Definition 6.

**Definition 6** At each transition of the GCA, each object changes velocity and position in a CA of length $L$ according to the rules below:

1. if $v(i) < V_{max}$ and $v(i) < gap(i)$ then $v(i) \leftarrow v(i) + 1$
2. if $v(i) > gap(i)$ then $v(i) \leftarrow gap(i)$
3. if $v(i) > 0$ and $rand() < P_d(i)$ then $v(i) \leftarrow v(i) - 1$
4. if $(x(i) + v(i)) \leq L$ then $x(i) \leftarrow x(i) + v(i)$

The first rule represents linear acceleration until the object reaches the maximum speed $V_{max}$. The second rule ensures that if there is another object in front of the current object, it will slow down in order to avoid collision. In the third rule, $P_d(i)$ models an erratic movement behavior. Finally, the new position of object $i$ is given by the fourth rule as sum of the previous position and the new velocity if the object is in the CA. Figure 2 shows the simulated movement of objects on a CA of the GCA in two consecutive timestamps. We can see that at time $t$, the speed of the object $a$ is smaller than the gap (i.e. the number of cells between the object $a$ and $b$). On the other hand, the object $b$ will reduce its speed to the size of the gap. According to the fourth rule, the objects move to the corresponding positions based on their speeds at time $t + 1$.

We use GCAs not only to model road networks, but also simulate the movements of moving objects by the transitions of the GCA. Based on the GCA, the *SP* method to anticipate future trajectories of moving objects is proposed. The SP method treats the objects' simulated results as their predicted positions to obtain its future in-edge trajectory. To refine the accuracy, based on different assumptions on the traffic conditions we simulate two future trajectories in discrete points for each object on its edge. Then, by linear regression and translating, the trajectory bounds that contain all possible future positions of a moving object on that edge can be obtained. When the object moves to another edge in the GCA, another simulation and regression will be executed to predict new future trajectory bounds. The SP method is shown in Fig. 3.




(a) Simulated trajectories (b) Trajectory bounds

**Fig. 3** The simulation-based prediction

Most existing work uses the CA model for traffic flow simulation in which the parameter $P_d(i)$ is treated as a random variable to reflect the stochastic, dynamic nature of a traffic system. However, we extend this model for predicting future trajectories of objects by setting $P_d(i)$ to values that model different traffic conditions. For example, laminar traffic can be simulated with $P_d(i)$ set to 0 or a small value, and the congestion can be simulated with a larger $P_d(i)$. By giving $P_d(i)$ two values, we can derive two future trajectories, which describe, respectively, the fastest and slowest movements of objects. In other words, the object future locations are most probably bounded by these two trajectories. The value of $P_d(i)$ can be obtained by the experiences or by sampling from the given dataset. Our experiments show one of methods to choose the value of $P_d(i)$.

Through the SP model, we obtain two bounds of objects' future trajectory. In the sequel, we apply this technique in our index to a set of moving objects that have similar movements and are treated as one object.

## 4 The adaptive unit

### 4.1 Structure and storage

Conceptually, an AU is similar to a one-dimensional MBR in the TPR-tree, which expands with time according to the predicted movement of the objects it contains. However, in the TPR-tree, it is possible that an MBR may contain objects moving in opposite directions, or objects moving at different speeds. As a result, the MBR may expand rapidly, which may create large overlaps with other MBRs. The AU avoids this problem by grouping objects having similar moving patterns. Specifically, for objects in the same GCA edge, we use a distance threshold and a speed threshold to cluster the adjacent objects with the same direction and similar speed. The thresholds are set according to the average length of road segments, the average maximum speed on the segment and also the adaptation in the experimental data sets. In comparison, the AU has no obvious enlargement because objects in the AU move in a cluster.

We now formally introduce the AU. An AU is a 7-tuple:

$$AU = (\texttt{auID}, \texttt{objSet}, \texttt{upperBound}, \texttt{lowerBound}, \texttt{edgeID}, \texttt{enterTime}, \texttt{exitTime})$$

where `auID` is the identifier of the AU, `objSet` is a list that stores objects belonging to the AU, `upperBound` and `lowerBound` are upper and lower bounds of predicted future trajectory of the AU. The trajectory bounds are derived from the trajectory bounds of the objects in the AU. We assume the functions of trajectory bounds as follows:

$$\texttt{upperBound}: \quad D(t) = \alpha_u + \beta_u \cdot t$$
$$\texttt{lowerBound}: \quad D(t) = \alpha_l + \beta_l \cdot t$$

`edgeID` denotes the GCA edge that the AU belongs to, `enterTime` and `exitTime` record the time when the AU enters and leaves the edge.

In the GCA, multiple AUs are associated with a GCA edge. Since AUs in the same edge are likely to be accessed together during query processing, we store AUs by clustering on their `edgeID`. That is, the AUs in the same edge are stored in the same disk pages. To access AUs more efficiently, we create a compact summary structure called the *direct access table* for each edge, which is treated as a secondary index of AUs can be accessed by a in-memory buffer. A direct access table stores the summary information of each AU on an edge (i.e. number of objects, trajectory bounds) and pointers to AU disk pages. Each AU corresponds to an entry in the direct access table, which has the following structure `(auID, upperBound, lowerBound, auPtr, objNum)`, where `auPtr` points to a list of AUs in disk storage and `objNum` is the number of objects included in the AU. Similar to AUs, the entries of the same direct access table and of the different direct access table but in the adjacent edge are grouped together so that we can get them into the buffer more efficiently. For the simple network with small amount of AUs, we can keep all direct access tables in the main memory since it only keeps the summary information of AUs.

### 4.2 The index scheme

We build a spatial index (e.g., R-tree) for the GCA (road network) over the AUs to form the index scheme for the network-constrained moving objects. The AU index scheme is a two-level index structure. At the top level, it consists of a 2D R-tree that indexes the spatial information of the road network. On the bottom level, its leaves contain the edges representing multiple road segments (i.e. polylines) included in the corresponding MBR of the R-tree and point to the lists of AUs. Each of entry in a leaf node consists of a road segment, i.e., a line segment in the polyline. The top level R-tree remains fixed during the lifetime of the index scheme (unless there are changes in the network). The index scheme is developed with the R-tree in this paper, but any existing spatial index can also be used without change.

Figure 4 shows the structure of the AU index scheme, which also includes a direct access table. The R-tree, the direct access table and AUs are stored in the disk. However, the direct access table stores the summary information of some AUs on the edge and is similar to a secondary index of AUs. In the index scheme, each leaf node of the R-tree can be associated with its direct access table by its `edgeID` and the direct access table can connect to corresponding AUs by `auPtr` in its entries. Therefore, we only need to update the direct access table when AUs change, which enhances the performance of the index scheme.

### 4.3 Optimizing for updates

When the updated locations are stored in a database in the server, the index structure of moving objects may be updated frequently with the update of locations. Our task is to reduce the cost of such indexing updates by a one-dimensional dynamic AU structure and an accurate prediction method.

An important feature of the AU is that it groups objects having similar moving patterns. The AU is capable of dynamically adapting itself to cover the movement of the objects it contains. By tightly bounding enclosed moving objects for some time

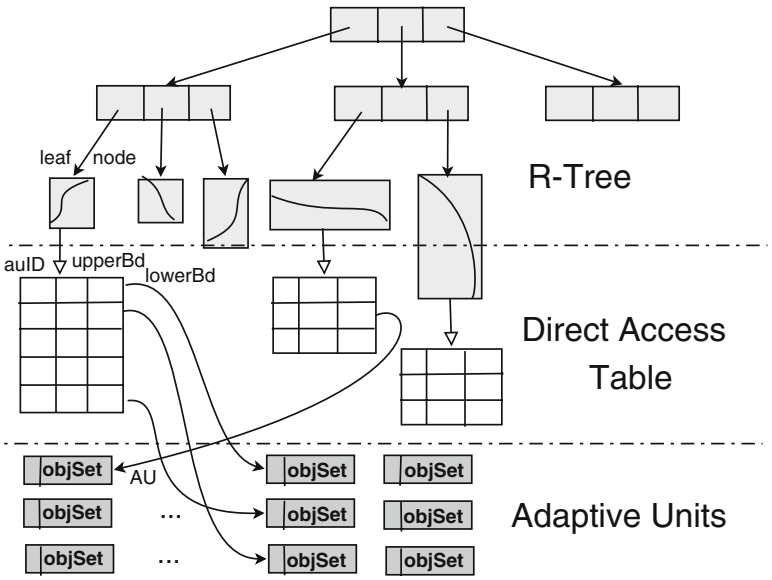



**Fig. 4** Structure of the AU index scheme

in the future, the AU alleviates the update problem of MBR rapid expanding and overlaps in the TPR-tree like methods.

For reducing updates further, the AU captures the movement bounds of the objects based on the SP method, which considers the road-network constraints and stochastic traffic behavior. Since objects in an AU have similar movements, we then predict the movement of the AU by the SP method, as if it were a single moving object. The specific locations of the individual objects inside AUs can be similar and obtained by trajectory bounds of the AU. Through the SP method, we obtain two predicted future trajectory bounds of objects. When an object's position exceeds the AU, the index needs to be updated to delete the object from the old AU and insert the object to another AU. The accurate prediction of an AU's movement and expanding with objects' movement makes it possible that the updated location of each object seldom affects the changing of the AU structure (e.g. deleting and inserting objects, creating and dropping AUs). Therefore, the SP method helps to reduce the index updating costs.

The future trajectory bounds are predicted at each GCA node when an AU is created. The trajectory bounds will not be changed along the edge that the AU moves on until the objects in the AU move to another edge in the GCA. It is evident that the range of predicted bounds of an AU will become wider with the time, which leads to lower accuracy of future trajectory prediction. However, if we issue another prediction when the predicted bounds are not accurate any more, the costs of simulation and regression are high. Considering that the movement of objects along one GCA edge is stable, we can assume the same trends of the trajectory bounds and adjust only the initial locations when the prediction is not accurate. Specifically, the AU treats its actual locations (the locations of the boundary objects) at that time as the initial locations of the two trajectory bounds and follow the same movement vector (e.g. slope of the bounds) as the previous bounds to provide more accurate predicted trajectory bounds. In this way, the predicted trajectory bounds can be effectively revised with few costs. Figure 3b shows the adaptation of the trajectory bounds. $t_q$ is the adaptation time of future trajectory bound and the $d_1$,$d_2$ are the actual locations of the first object and last object respectively in the AU. The trajectory bounds are revised according to the actual locations and the

original bounds' slopes. Therefore, without executing more prediction, the prediction accuracy of the objects' future trajectories can be kept high.

Since the R-Tree indexes the GCA, it remains fixed, and the update of the AU index scheme restricts to the update of AUs. Specifically, an AU is usually created at the start of one edge and dropped at the end of the edge. Since the AU is a one-dimensional structure, it performs update operations much more efficiently than the two-dimensional indexes. We will describe these operations in details.

### 4.4 Update operations

The update of an AU can be of the following form: creating an AU, dropping an AU, adding objects to an AU and removing objects from an AU.

**Creating an AU** To create an AU, we first compose the *objSet* - a list of objects traveling in the same direction with similar velocities (velocity difference is not larger than a speed threshold), and in close-by locations (location difference is not larger than a distance threshold). We then predict the future trajectories of the AU by simulation and compute its trajectory bounds. In fact, we treat the AU as one moving object (the object closest to the center of the AU) and predict its future trajectory bounds by predicting this object. The prediction starts when the AU is created and ends at the end the edge. Finally, we write the created AU to the disk page and insert the AU entry to its summary structure. Factually, AU is created in two cases: 1) at the initial time with on bulk-loading at each network edge and 2) when the objects leave original edge with a single object described in Algorithm 1.

**Dropping an AU** When objects in an AU move out of the edge, they may change direction independently. So we need to drop this AU and create new AUs in adjacent edges to regroup the objects. When the front of an AU touches the end of the edge, some objects in the AU may start moving out of the edge. However, the AU cannot be dropped because a query may occur at that time. Only after the last object in the AU enters another edge and joins another AU, can the AU be dropped. Dropping an AU is simple. Through its entry in direct access table, we find the AU and delete it.

**Adding and removing objects from an AU** When an object leaves an AU, we remove this object from the AU and find another AU in the neighborhood to check if the object can fit that AU. If it can, the object will be inserted into that AU, otherwise, a new AU is created for this object. Specifically, when adding an object into an AU, we first find the direct access table of the edge that the object lies and, by its AU entry in the table, access the AU disk storage. Finally, we insert into the objects list of the AU and update the AU entry in the direct access table. Removing an object from an AU has the similar process.

Therefore, when updating an object in the AU index scheme, we first determine whether the object is leaving the edge and entering another one. If it is moving to another edge, we delete it from the old AU (if it is the last object in the old AU, the AU is also dropped) and insert it into the nearest AU to the object in terms of the network distance or create a new AU in the edge it is entering. Otherwise, we do not update the AU that the object belongs to unless its position exceeds the bounds of the AU. In that case, we execute the same updates as those when it moves to another edge. When the AU is not updated, we check whether the object is the boundary

object of the AU and whether its actual position exceeds the predicted bounds of the AU to a precision threshold $\varepsilon$ (explained in the experiments of prediction accuracy), for the purpose of adapting the trajectory bounds of the AU. Factually, we find, from the experiment evaluation, that the chances that objects move beyond the trajectory bounds of its AU on an edge are very slim. The algorithm 1 shows the update algorithm when updating an object in the AU. Like the node capacity parameter in the index tree, MAXOBJNUM in the algorithm 1 is also used to restrict the number of object entries in an AU. It is set according to the object entry storage size and AU storage size.

In summary, updating the AU-based index is easier than updating the TPR-tree. It never invoke any complex node splitting and merging. Moreover, thanks to the similar movement features of objects in an AU and the accurate prediction of the SP method, the objects are seldom removed or added from their AU on an edge, which reduces the number of index updates.

### 4.5 Query algorithm

In this part, we propose an algorithm for predictive range query in the AU index scheme. It can also be extended to support the (K) Nearest Neighbor query

**Algorithm 1** Update($objID$, $position$, $velocity$, $edgeID$)

**input** : $objID$ is the object identifier, $position$ and $velocity$ are its position and velocity, $edgeID$ is the edge identifier where the object lies

Find $AU$ where $objID$ is included before update;

**if** $AU.edgeID \neq edgeID$ *or (objID is not the boundary object of AU and (position* $<$ $AU.lowerBound$ *or position* $>$ $AU.upperBound$*))* **then**

// The object moves to a new edge or exceeds bounds of its original AU

Find the nearest AU $AU_1$ for $objID$ on $edgeID$;

**if** GetNum($AU_1.objSet$) $<$ $MAXOBJNUM$ *and* $ObjectFitAU(objID, position, velocity, AU_1)$ **then**

InsertObject($objID$, $AU_1.auID$, $AU_1.edgeID$);

**else** $AU_2 \leftarrow$ CreateAU($objID$,$edgeID$);

**if** GetNum($AU.objSet$) $>$ 1 **then**
DeleteObject($objID$, $AU.auID$, $AU.edgeID$);

**else** DropAU($AU.edgeID$, $AU.auID$);

**else if** ($objID$ *is the low boundary object of AU and position exceeds* $AU.lowerBound$ *to* $\varepsilon$*) or (objID is the high boundary object of AU and* $AU.upperBound$ exceeds *position to* $\varepsilon$*)* **then**
AdaptAUBounds*(*$AU$, *position)*;

and continuous query. A predictive range query captures all objects moving in a road network whose locations are inside a specified region $R$ during time interval $[T_1, T_2]$ in the future. Given a spatiotemporal window range with $(X_1, Y_1, X_2, Y_2, T_1, T_2)$, the query algorithm on the AU index scheme consists of the following steps:

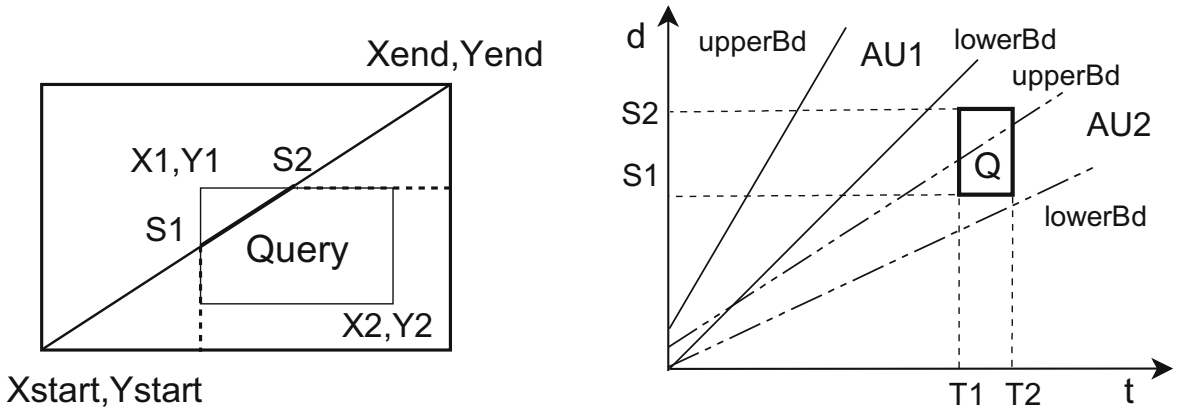
1) We first perform a spatial window range search $(X_1, Y_1, X_2, Y_2)$ in the top level R-Tree to locate the edges (e.g. $e_1, e_2, e_3, \ldots$) that intersect the spatial query range.
2) For each selected edge $e_i$, we transform the original 3D search $(X_1, Y_1, X_2, Y_2, T_1, T_2)$ to a 2D search $(S_1, S_2, T_1, T_2)$ $(S_1 \leq S_2, T_1 \leq T_2)$, where $S_1$ and $S_2$ are the relative distances from the start vertex along the edge $e_i$. Figure 5a gives an example when the query window range only intersects one edge $e_1$. In the case of multiple intersecting edges, we can divide the query range into several sub-ranges by edges and apply the transformation method to each edge. The method is also applicable to the various modes the query and edges intersect. For space limitation, we only illustrate the case in Fig. 5a and compute its relative distances $S_1$ and $S_2$. It can be easily extended to other cases. Suppose $X_{start}, Y_{start}, X_{end}, Y_{end}$ are the start vertex coordinates and the end vertex coordinates of the edge $e_1$. According to Thales Theorem about similar triangles, we obtain $S_1$ and $S_2$ as follows:

$$r = \sqrt{(X_{start} - X_{end})^2 + (Y_{start} - Y_{end})^2}$$

$$S_1 = \frac{X_1 - X_{start}}{X_{end} - X_{start}} r$$

$$S_2 = \frac{Y_1 - Y_{start}}{Y_{end} - Y_{start}} r$$

3) We further find the adjacent edges of $e_1$ on which objects are possible to move into the window range during the future period $[T_1, T_2]$. For supporting future spatio-temporal range queries, the TPR-tree expands MBRs towards every direction according to the maximum speed of objects, which, when applied to the network, will result in large candidate result set including some objects that

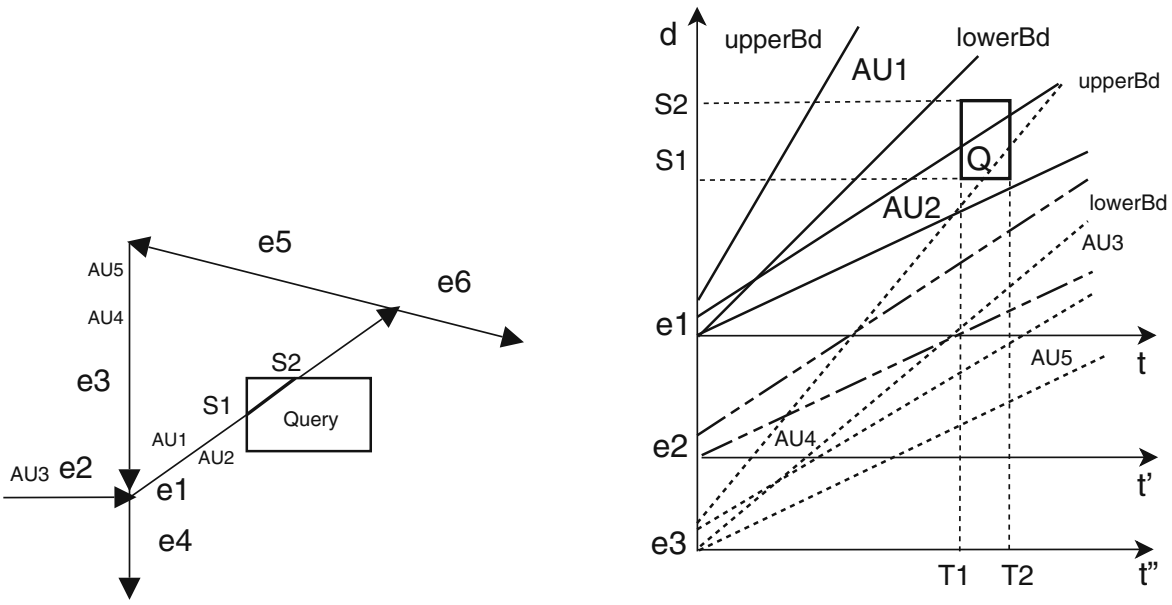



(a) Query transformation (b) Query process in AUs

**Fig. 5** Window range query in the AU index scheme

are impossible to move into the query range due to network constraint. There are limited possibilities of objects' movement in the road network. Therefore, we filter the candidate AUs in the adjacent edges possibly intersecting the window range by expanding along the network according to the maximum speed allowed in the network, adjacent table of edges and future query time. Figure 6 gives an example of network expanding in the query processing where arrow denotes the direction of edge. Let $V_max$ the maximum speed and $T_0$ ($= 0$ in our example) the current time, it expands the network from the point of edge $e_1$ intersecting the spatial window (e.g. locations of $S_1$ in Fig. 6a) towards the reverse direction of $e_1$ and then continue to the adjacent edges obtained from the reverse adjacent table of $e_1$ until a expanded distance $V_{max} * (T_2 - T_0)$ is reached. The traversed edges $e_2, e_3$ in this example are returned. The AUs on these edges (e.g. AU3 on $e_2$ and AU4, AU5 on $e_3$ in Fig. 6a) will be further checked whether they are possible to intersect the query range during $[T_1, T_2]$. In this way, we can avoid the false negative for objects in the other edges during the query processing.

4) The transformed query $(S_1, S_2, T_1, T_2)$ is executed in each of the AUs in the direct access table of the corresponding edge $e_1$. As illustrated by Fig. 5b, an AU is suitable to the query only if the 2D window range intersects the area between the upper and lower trajectory bounds of the AU. Otherwise when the query is below the lower bound (e.g. $\beta_l * T_1 + \alpha_l > S_2$) or above the upper bound (e.g. $\beta_u * T_2 + \alpha_u < S_1$) of the AU, the query cannot contain objects in this AU. The computations of transformed queries in adjacent edges $e_2$ and $e_3$ are also together showed in Fig. 6b. For the adjacent edge $e_2$ with the length of $l(e_2)$, we revise the transformed query to $(S_1 + l(e_2), S_2 + l(e_2), T_1, T_2)$ and filter AUs suitable to the query by linking $e_2$ and $e_1$, which is showed in the $t'$-$d$ coordinate plane of Fig. 6b. We use the same method to filter AUs on the adjacent edge $e_3$ by linking $e_3$ and $e_1$, which is showed in the $t''$-$d$ coordinate plane of Fig. 6b. This is reasonable to treat these AUs as candidates since the objects in them are also




(a) Network Expanding (b) Query process in AUs

**Fig. 6** Network expanding in query processing

likely to move to the query range in the future time. In our example, the query returns AU2 in edge $e_1$ and AU4 in adjacent edge $e_3$. By the trajectory bounds of the AU, we can determine whether the transformed query intersects the AU, thus filtering out the unnecessary AUs quickly.

5) Finally, we access the selected AUs in disk storage and return the objects satisfying the predictive query window.

Future spatio-temporal queries in a road network are more difficult to compute when considering the objects cross the different road segment edge because the future movement of objects in the road intersection is complex, but has limited possibilities due to the network constraint. We compute the query results which are likely to move to the road segment edge and location range in the near future that query intersects.

## 5 Performance analysis

In this section, we analyze the performance of the AU index scheme. We first analyze the I/O cost of the query and update algorithm, and then perform experimental evaluation.

### 5.1 Algorithms analysis

We follow the main assumptions of [33] in our analysis, in particular we assume that rectangles, including the whole map, are square. Let $M$ be the total number of edges of the GCA, $W$ be the width of the map, $N$ be the total number of objects and $n$ be the average number of objects in an AU. The average number of AUs in an edge is $N/(nM)$. We assume that $B$ is the maximum number of objects in a disk page. The average number of AUs in a page is $ceiling(B/n)$.

For a spatio-temporal query window $(X_1, Y_1, X_2, Y_2, T_1, T_2)$, a spatial search is first performed in the top level R-tree to locate the edges that intersect the spatial window. Let $N_r$ be the number of data rectangles of the R-tree, $f$ be its average fanout, $h = 1 + \lceil \log_f \frac{N_r}{f} \rceil$ its height, and $S_{l,x}$, $S_{l,y}$ the average extents of node rectangles at level $l$ on $X$ and $Y$ coordinates. Assume that each node is in one disk page, the average number of disk accesses for the spatial search $(X_1, Y_1, X_2, Y_2)$ is given by [33]:

$$\sum_{l=1}^{h-1} \frac{N_r}{f^l} \frac{\left(S_{l,x} + |X_2 - X_1|\right)\left(S_{l,y} + |Y_2 - Y_1|\right)}{W^2}$$

Since each entry in the leaf node of the R-tree only contains one edge, the average number of edges intersecting the spatial query is given by:

$$M \frac{\left(S_{1,x} + |X_2 - X_1|\right)\left(S_{1,y} + |Y_2 - Y_1|\right)}{W^2}$$

For each selected edge, we scan its direct access table for the purpose of only accessing relevant AUs. We compute the average number of AUs intersecting the transformed query $(S_1, S_2, T_1, T_2)$. In Fig. 3b, the two trajectory bounds of one AU divide the coordinate plane into three parts: upper area (above the upper bound),

middle area (between the upper and lower bounds) and lower area (below the lower bound). We assume that the upper and lower areas represent respectively the percentage of $\mu_u$, $\mu_l$ of the total area of the plane. Factually, $\mu_u$ and $\mu_l$ are the probabilities that a transformed point query is respectively above and below the trajectory bounds of the AU. The probability that the query intersects the AU is $(1 - \mu_u^2 - \mu_l^2)$. It is not difficult to compute the average probabilities $\overline{\mu}_u$, $\overline{\mu}_l$ of the AUs on the edge using their bound functions and the length of the edge. Here we use the same analysis method for the computations of transformed queries in adjacent edges and ignore the cost of network expanding for simplicity. Now, we can get the average number of relevant AUs as follows:

$$\left(1 - \overline{\mu}_u^2 - \overline{\mu}_l^2\right) \frac{N}{nM}$$

Finally, for each relevant AU, we need to find the moving objects satisfying the predictive query range. Since the AUs on the same edge are likely clustered in the same disk page, the average I/O cost of accessing relevant AUs and moving objects on each selected edge is given by:

$$\left(1 - \overline{\mu}_u^2 - \overline{\mu}_l^2\right) \frac{N}{MB}$$

Therefore, the total I/O cost for a spatiotemporal window query in the AU index scheme is given by:

$$\frac{1}{W^2}\left(\sum_{l=1}^{h-1} \frac{N_r}{f^l}\left(S_{l,x} + |X_2 - X_1|\right)\left(S_{l,y} + |Y_2 - Y_1|\right) + \frac{N}{B}\left(S_{1,x} + |X_2 - X_1|\right)\left(S_{1,y} + |Y_2 - Y_1|\right)\left(1 - \overline{\mu}_u^2 - \overline{\mu}_l^2\right)\right)$$

For improving the efficiency of the prediction of AU, the trajectory bounds of AU are computed based on the simulation not of all objects in it but of the object closest to the center of the AU. In this way, it seems that the query processing in the AU index will possibly not return correct query results (false negative) since the extrapolated position of the object at the query time will be outside of the bounds of the AU. However, this seldom happens for the following three reasons. 1) AU is constructed by a group of moving objects with similar moving pattern and maintained by tightly bounding enclosed moving objects for some time in the future. It is reasonable to approximate the AU by its center object. 2) In the SP method, to refine the prediction accuracy, we simulate two trajectories based on different assumptions on the traffic conditions (e.g. laminar and congested traffic) and translate the regressed lines outside to contain all possible future position of the object as soon as possible. 3) The adaptation of the trajectory bounds of AU also further improves the accuracy of trajectory prediction over time. From the experiments in Section 5.2.3, it is proved that such the approximation of AU simulation by its center object can achieve high efficiency improvement by causing very slim possibility of incorrect query results.

We then analyze the cost of updates in the AU index scheme. In order to update an object, we first scan AUs from the entries in the direct access table corresponding

to its edge. The average number of disk accesses to find the AU page that the object belongs to (scan the pages of AUs on an edge) is $N/2MB$. If the object is entering a new edge or exceeds bounds of its AU, it will be inserted into the nearest AU and deleted from the old AU. This needs 1 disk accesses to read the nearest AU and 2 disk access to write the pages of the old AU and the new AU. If the object is the last object in the old AU, the old AU with the last object will be dropped which costs 1 disk access to write the page of the old AU. If the object cannot be inserted into the existing AU, a new AU will be created with 1 disk access. Therefore, the average cost of an update is $N/2MB + 3$.

## 5.2 Experimental evaluation

Since the R-tree in our structure only indexes the static spatial information of road networks, we compare it in the experiments with the TPR-tree-like method (taking the most popular TPR-tree for example) in which the R-tree is used to index the continuously changing moving objects. We measure the update performance with the individual update, update frequency and total update costs and the query performance of AU index scheme (denoted as "AU index"), the TPR-tree and the AU index scheme when the direct access table is not used (denoted as "AU index without DT"). We then study the effect of parameter $P_d$ on the SP and finally compare the prediction accuracy of the SP method with that of the linear prediction method.

### *5.2.1 Datasets*

We use two datasets for our experiments. The first is generated by the CA simulator, and the second by the Brinkhoff's Network-based Generator [5]. We use the CA traffic simulator to generate a given number of objects in a uniform network of size $10000 \times 10000$ consisting of 500 edges. Each object has its route and is initially placed at a random position on its route. The initial velocities of the objects follow a uniform random distribution in the range [0, 30]. The location and velocity of every object is updated at each time-stamp. The Brinkhoff's Network-based Generator is used as a popular benchmark in many related work. The generator takes a map of a real road network as input (our experiment is based on the map of Oldenburg including 7035 edges). The positions of the objects are given in two dimensional X-Y coordinates. We transform them to the form of (`edgeid`, `pos`), where `edgeid` denotes the edge identifier and `pos` denotes the object relative position on the edge. The generator places a given number of objects at random positions on the road network, and updates their locations at each time-stamp. We implemented both the AU index scheme and the TPR-tree in Java and carried out experiments on a Pentium 4, 2.4 GHz PC with 512 MB RAM running Windows XP. To improve the performance of the index structure, we employed a LRU buffer of the same size (200K) as the one used in the TPR-tree [29]. Especially, for the AU index with DT, the LRU buffer is used for the R-tree nodes, AU pages, and DAT pages. The same amount of main memory is allocated to buffers for all of the three compared index structure. Since the map of Oldenburg is a relatively small network and the direct access tables are frequently accessed, most of them are kept in the main memory in our experiments.

**Table 1** Parameters and their settings

| Parameters | Settings |
|---|---|
| Page size | 4K |
| Node capacity | 100 |
| Numbers of queries | 200 |
| Numbers of mo(cars) | 10K, ... , 50K, ... , 100K |
| Numbers of updates | 100K, ... , 500K, ... , 1M |
| Dataset generator | CA simulator, network-based generator |

We summarize workload parameters in Table 1, where values in bold are default values.

### *5.2.2 Update cost*

We compare the cost of index update for the AU index and the TPR-tree in terms of the average individual update cost, update frequency and total update cost.

**Individual Update Cost** We study the individual update performance of the index while varying the number of moving objects and updates. Figure 7 shows the average individual update cost when increasing the data size from 10K to 100K. Figure 8 shows how the performance varies over time. Clearly, updating the TPR-tree tends to be costly, and the problem is exacerbated when the data size increases. In each case of different data size and different number of updates, the AU index has much lower update cost than the TPR-tree. The main reason can be explained as follows. Each update of the TPR-tree involves the search of an old entry and a new entry, as well as the modification of the index structure (node splitting, merging, and the propagating of changes upwards). The cost increases with larger data size due to more overlaps among MBRs. The changes of index structure with the increase of data updates also affect the performance of the TPR-tree. However, the AU index has better performance because its update only restricts to the AU's update and as a one-dimensional access structure, the AU has few overlaps and incurs no cost associated with node splitting and the propagation of MBR updates.

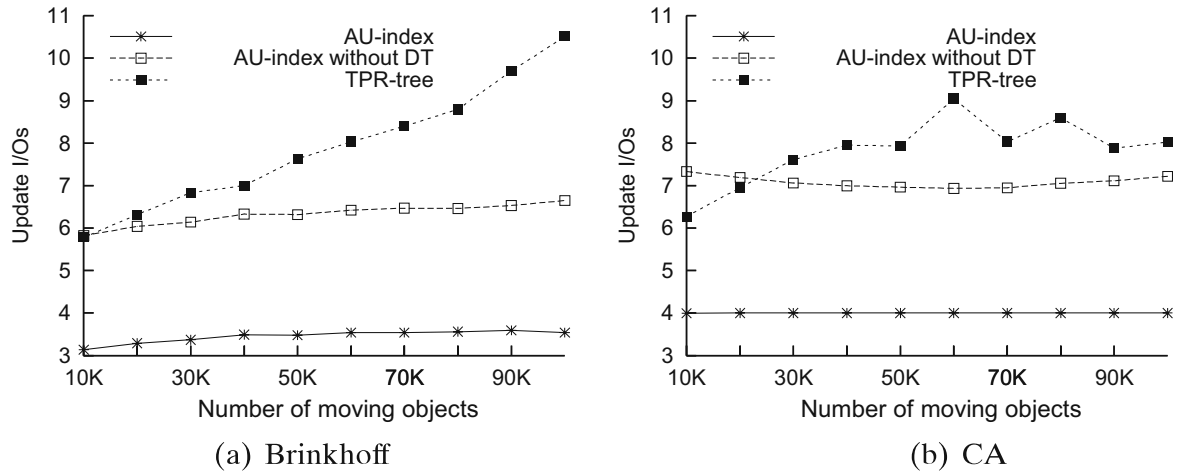



**Fig. 7** Individual update cost with different datasize

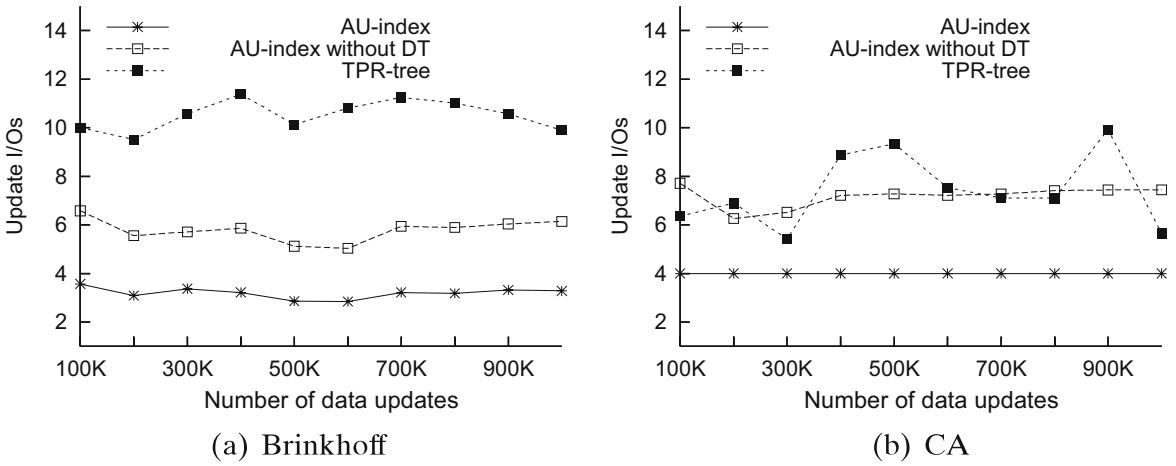



**Fig. 8** Individual update cost over time

The direct access table of the AU index has a significant contribution in improving update performance. This is because searching the specific AU is accelerated by the secondary index structure.

**Update Frequency** Frequent updates of moving objects (a.k.a. data updates) may lead to frequent updates of index. All data updates (that are received according to some object update policy) should be recorded in the index, but may lead to different numbers of "index update" for the AU and TPR-tree respectively. In our experimental evaluations, "index update" for AU denotes an update of an object that invalidates the objects AU. For example, when an object's position exceeds the bounds of AU, the index needs to be updated to delete the object from the old AU and insert it to another one. For the TPR-tree, the bounding rectangles are recomputed even if they are not invalidated. We did not count such re-computations as "index update" and only counted the updates which invalidated the bounding rectangles of TPR-tree. In this experiment, we measure the "index update rate", which is the ratio between number of such index update and number of data update, for every 100K data updates and different data size. Figures 9 and 10 show that the

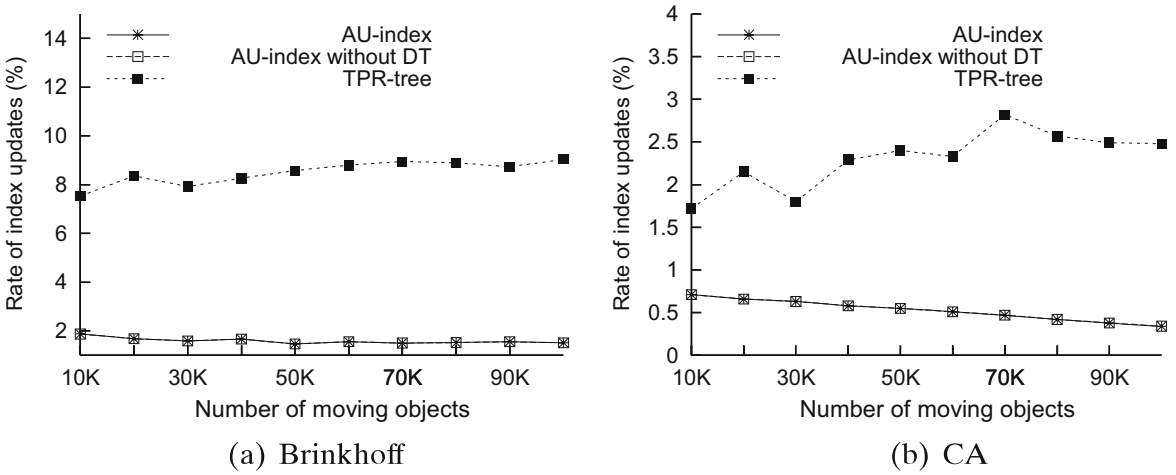



**Fig. 9** Index update frequency with different datasize

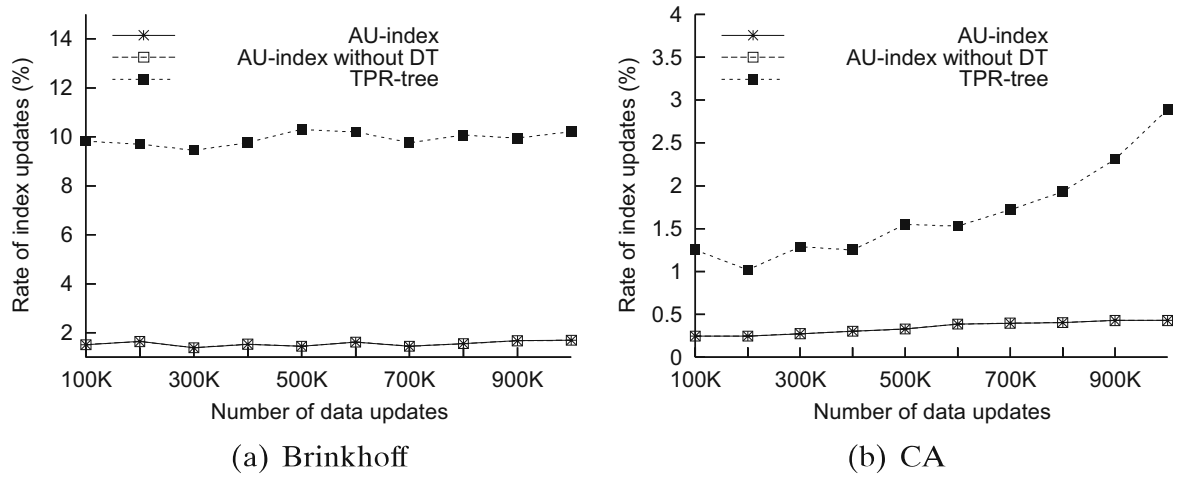



**Fig. 10** Index update frequency over time

update rate of the TPR-tree is nearly 4 to 5 times more than that of the AU index. The index update rate depends on the prediction method. In the AU index, the future positions of the object are predicted more accurately, so the object is likely to remain in its AU, which leads to fewer index updates. As to the reduction in the object update frequency when the SP method is used, we also evaluate the performance in paper [8].

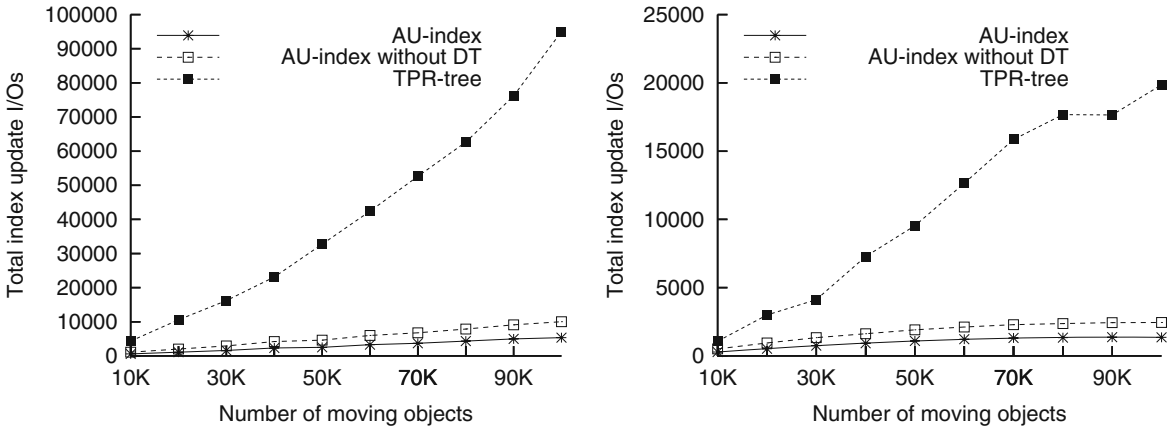
**Total Update Costs** The total update costs depend on the update frequency and the average individual update cost, and it can reflect the index update performance more accurately. From both Figs. 11 and 12, we can see that although the AU index has to deal with the creation and dropping of AUs, the TPR-tree incurs much higher update costs than the AU index and its performance deteriorates dramatically as data size increases. This is mainly due to the inaccuracy of the linear prediction model and the complex reconstruction of the TPR-tree (e.g. splitting and merging).

For each data size, the update costs of the two indexes in the Brinkhoff's dataset are both higher than those in the CA dataset due to the higher complexity of road network and skewed spatial distribution of objects in the Brinkhoff's dataset.

### *5.2.3 Query cost*

**Effect of Data Size** We study the window range query performance of the TPR-tree and the AU index while varying the number of moving objects from 10k to 100k. Figure 13 shows the average number of I/O per query with query window size 50. In each case, the query cost increases as the data size increases. However, the AU index has much lower cost than the TPR-tree. This is because the AUs in the AU index have much less overlaps than the MBRs in the TPR-tree, and the overlaps to a large extent determine the range query cost.

The AU index with the direct access table achieves better performance than the AU index without it. This is because the secondary index structure enables us to filter some unnecessary AUs during the search of AUs that intersect the range query. However, for the Brinkhoff's dataset the benefit of the direct access table is not obvious because the large number of small edges in the network reduces chances of

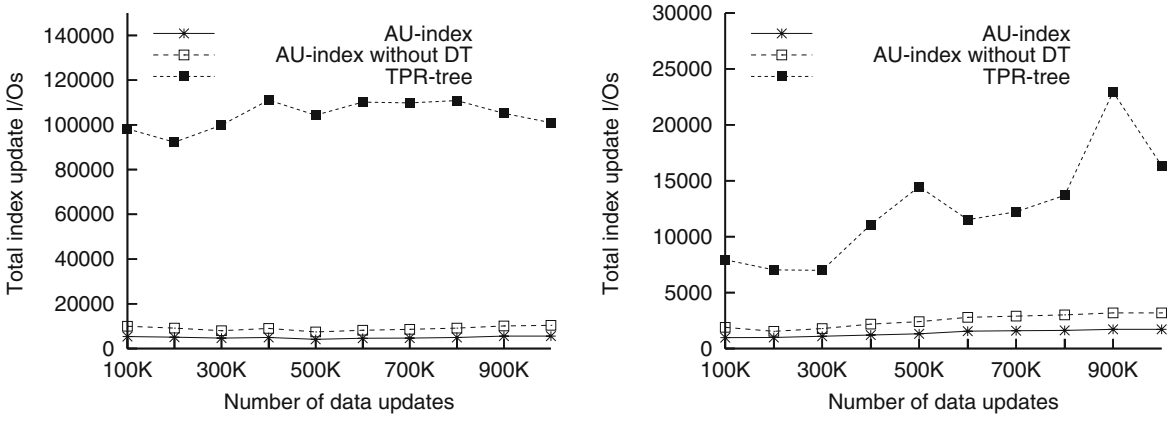



(a) Brinkhoff (b) CA

**Fig. 11** Total update cost with different datasize

filtering the AUs not included in the range query. The search costs of the two indices in different datasets are also quite different for the same reasons as mentioned in update performance.

**Effect of Update** We then study the window range query performance of the TPR-tree and the AU index with different update settings. We increase the number of updates from 100K to 1M to examine how query performance is affected. We issued 200 range queries with window size 50 for every 100K updates in a 1M dataset. Figure 14 shows that the cost of the TPR-tree increases much faster as the number of updates increases. The cost of the AU index is considerably lower and is less sensitive to the number of updates. This is because as objects move apart, the amount of dead space in the TPR-tree increases, which makes false hits more likely. Also, updates lead to the expanding and overlaps of MBRs, which further deteriorate the performance of the TPR-tree. For the AU index, the increase of the updates hardly affect the total number of AUs, and the chances of overlaps of different AUs are very slim.




(a) Brinkhoff (b) CA

**Fig. 12** Total update cost over time

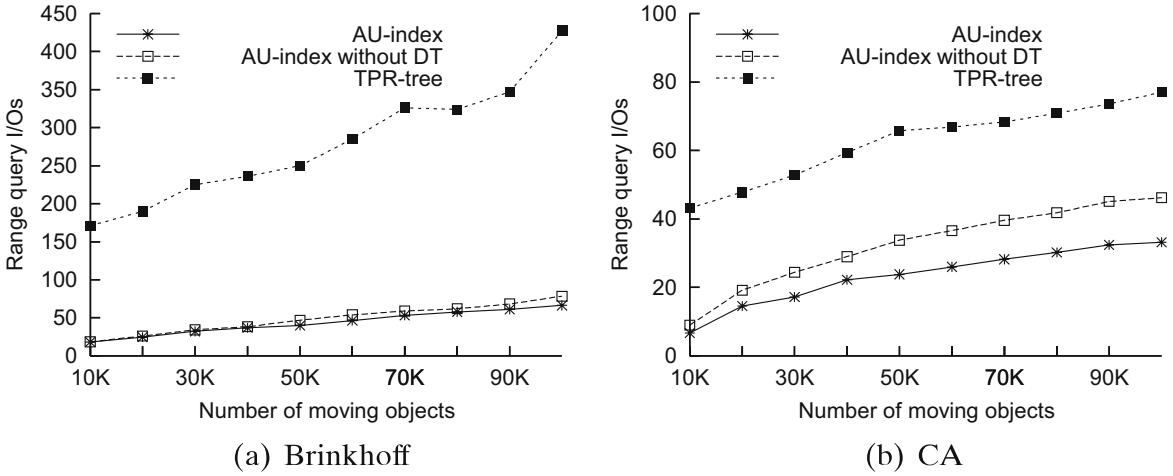



**Fig. 13** Query performance with data size

**Effect of Query Window Size** To study the effect of query window size on performance, we increase the window size from 10 to 100 (the fraction of the total space from 1/1000 to 1/100) for 100K data size with a workload of 200 range queries. Figure 15 shows the query cost as a function of the query window size. It is clear that for all the indexes, query cost increases with the query window size. This is so because larger windows contain more objects and therefore lead to more node accesses. However, this effect is more obvious on the TPR-tree.

**Query Recall** With the same dataset and the window size 50, we measure the query recall in AU index with the approximation of AU simulation by its center object. By referencing the query results in the TPR-tree, we compute the false negative and compare the efficiency of the AU index with approximation and without approximation. The results show that the approximation of AU simulation by its center object can achieve high efficiency improvement (average 25% increase of efficiency) by causing very slim possibility (average 5% possibility) of incorrect query results (average 6% false negative).

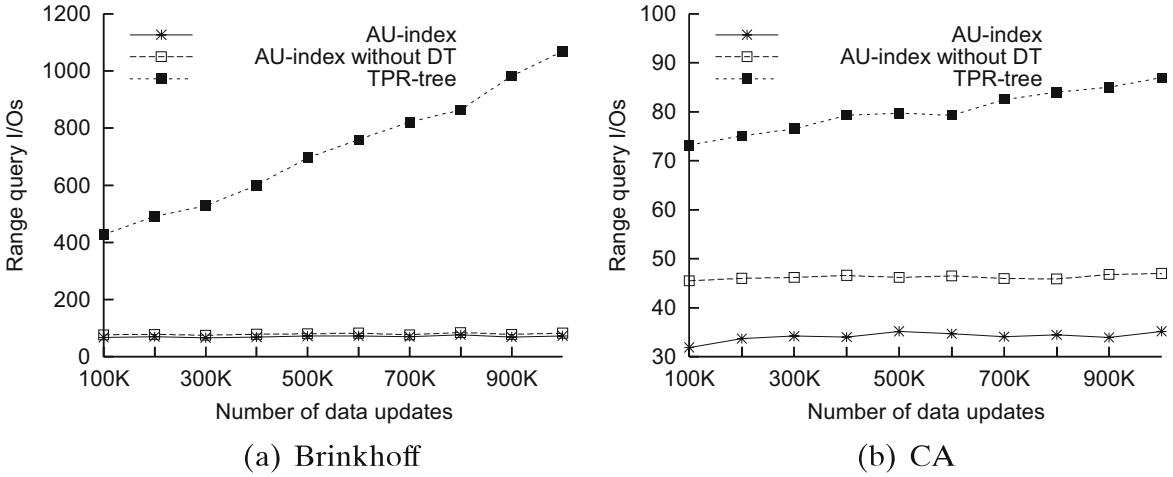



**Fig. 14** Effect of updates on query

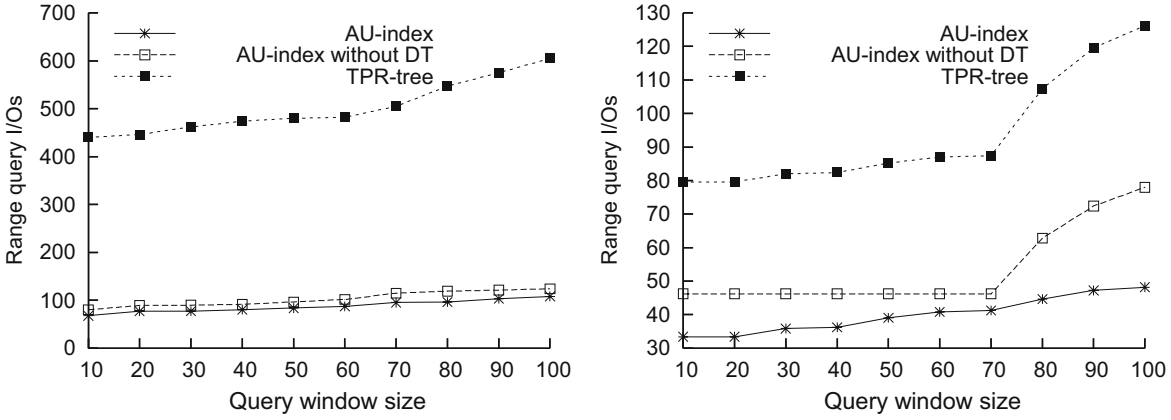



**Fig. 15** Effect of query window size on query performance

### *5.2.4 Prediction accuracy*

The SP method can be used in the AU index structure to reduce the indexing updates cost in that it is more accurate in predicting the future trajectories of AU with some similar objects than the linear prediction method. For simplicity, we study the effect of simulation parameter and evaluate the prediction accuracy by applying prediction method to the location update policy of object [8].

**The Slowdown Rate** $P_d$ The simulation has an important effect on the prediction accuracy and therefore affects the efficiency of query and update. We study the effect of the choice of different $P_d$, which determines the two predicted trajectories corresponding to the fastest and slowest movement. We test on the Brinkhoff's dataset with different data size and use $P_d$ from 0 to 0.5 and measure the average prediction accuracy by "average error" and "overflow rate". The average error is the average absolute error between the predicted and actual positions, and the overflow rate represents the probability of predicted positions exceeding the actual positions. The purpose of this metric is to find the closest two trajectories binding the actual one

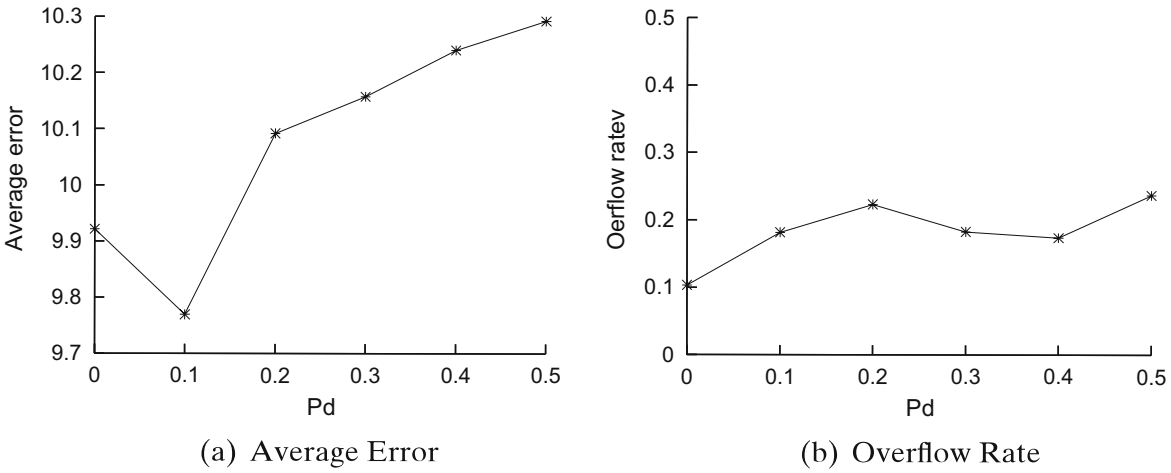



**Fig. 16** Prediction accuracy with $P_d$

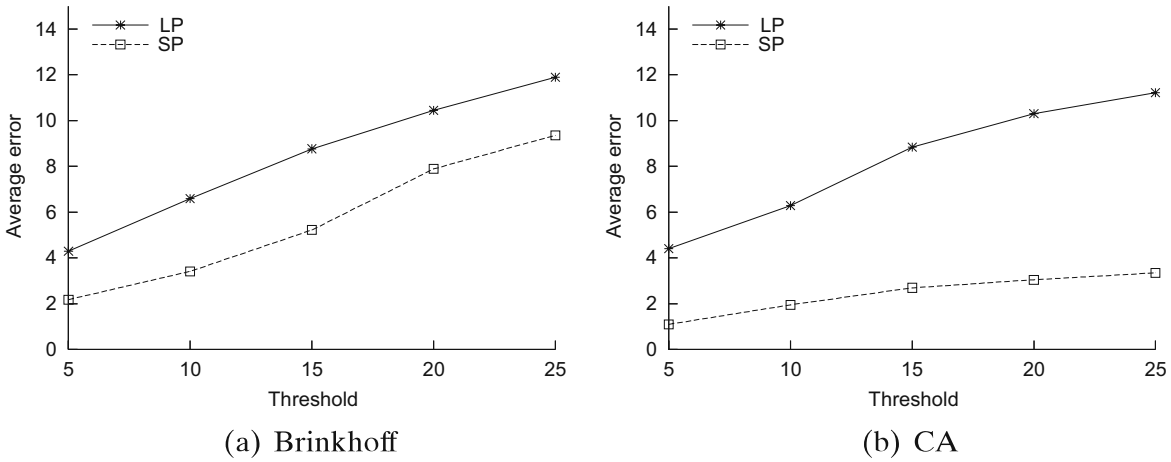



**Fig. 17** Prediction accuracy with threshold

as future trajectories. In this way, we can choose the $P_d$ both with lower average error and overflow rate. Figure 16 shows the prediction accuracy of the SP with different slowdown rates. We can see that when $P_d$ is set to 0 and 0.1, both the average error and overflow rate are lower than others. Therefore, we use the value 0 and 0.1 as slowdown rates for the fastest movement bound and the slowest movement bound to obtain better prediction results.

**Prediction Accuracy and Cost** Finally, we compare the precision of the SP method with the LP method. We measure the prediction accuracy by "average error" but with different threshold $\varepsilon$. The threshold $\varepsilon$ represents the maximum deviation between the predicted locations of an object and its real locations allowed in the prediction. That means when the deviation exceeds the threshold $\varepsilon$, we make another prediction. From Fig. 17, we observe that average error will increase when threshold increases. This is because the larger the threshold is, the larger the deviation becomes, which leads to the more errors. It is tenable in both the LP and SP method. However, the SP method predicts more accurately than the LP method with any threshold $\varepsilon$.

To measure the time cost of the prediction, we compute the average CPU time when simulating and predicting the movement of one object along the edge with length 1000 in different dataset sizes. The results show that the average cost of one SP is about 0.25 ms. This is quite acceptable.

## 6 Conclusions and future work

Indexing objects moving in a constrained network especially the road network is a topic of great practical importance. We focus on the index update issue for the current positions of network-constrained moving objects. We introduce a new access structure, AU that exploits as much as possible the characteristics of the movements of objects. The updates of the structure are minimized by an accurate prediction method which produces two trajectory bounds based on different assumptions on the traffic conditions. The efficiency of the structure also results from the possible reduction of dimensionality of the trajectory data to be indexed. Our experimental

results performed on two datasets show that the efficiency of the index structure is one order of magnitude higher than the TPR-tree.

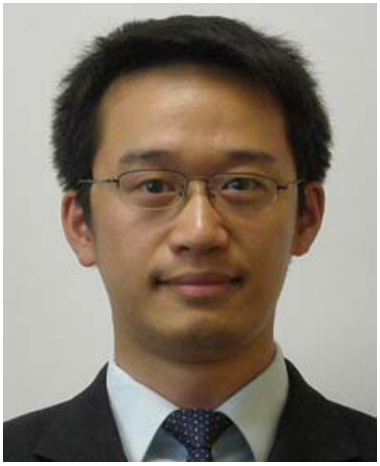
We will extend the query algorithms to support the KNN query and continuous query for moving objects in the road network.

**Acknowledgements** This research was partially supported by the grants from the Natural Science Foundation of China under grant number 60573091; China 863 High-Tech Program (No. 2007AA01Z155); China National Basic Research and Development Program's Semantic Grid Project (No. 2003CB317000); Program for New Century Excellent Talents in University (NCET). Program for Creative PhD Thesis in University. The authors would like to thank Jianliang Xu and Haibo Hu from Hong Kong Baptist University and Stéphane Grumbach from CNRS, LIAMA in China for many helpful advice and assistance. The author also appreciates Yanyan, Guo and Zhen Xiao from Renmin University of China for helping the implementation of the experiments and Wenbo, Mao from EMC Research China for English improvement.

## References

1. Aggarwal C, Agrawal D (2003) On nearest neighbor indexing of nonlinear trajectories. In: Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART symp. on principles of database systems, San Diego, 9–11 June 2003, pp 252–259
2. Agarwal PK, Arge L, Erickson J (2000) Indexing moving points. In: Proc. of the 19th ACM SIGMOD-SIGACT-SIGART symp. on principles of database systems, Dallas, 15–18 May 2000, pp 175–186
3. Almeida VT, Güting RH (2005) Indexing the trajectories of moving objects in networks. Geoinformatica 9(1):33–60
4. Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The R*-tree: an efficient and robust access method for points and rectangles. In: Proc. of the ACM SIGMOD int. conf. on management of data, Atlantic City, May 1990, pp 322–331
5. Brinkhoff T (2002) A framework for generating network-based moving objects. Geoinformatica 6(2):153–180
6. Cho H, Chung C (2005) An efficient and scalable approach to CNN queries in a road network. In: Proc. of the 31st int. conf. on very large data bases, Trondheim, 30 August–2 September 2005, pp 865–876
7. Chen J, Meng X, Guo Y, Grumbach S, Sun H (2006) Modeling and predicting future trajectories of moving objects in a constrained network. In: Proc. of the 7th int. conf. on mobile data management (MDM), Nara, 9–13 May 2006, pp 156 (MLASN workshop)
8. Chen J, Meng X, Li B, Lai C (2006) Tracking network-constrained moving objects with group updates. In: Proc. of the 7th int. conf. on web-age information management (WAIM), Hong Kong, 17–19 June 2006, pp 158–169
9. Cheng R, Xia Y, Prabhakar S, Shah R (2005) Change tolerant indexing for constantly evolving data. In: Proc. of 21st int. conf. on data engineering, Tokyo, 5–8 April 2005, pp 391–402
10. Ding Z, Güting RH (2004) Managing moving objects on dynamic transportation networks. In: Proc. of 16th int. conf. on scientific and statistical database management (SSDBM), Santorini Island, 21–23 June 2004, pp 287–296
11. Frentzos E (2003) Indexing objects moving on fixed networks. In: Proc. of the 8th intl. symp. on spatial and temporal databases, Santorini Island, July 2003, pp 289–305
12. Guttman A (1984) R-trees: a dynamic index structure for spatial searching. In: Proc. of the ACM SIGMOD int. conf. on management of data, Boston, June 1984, pp 47–57
13. Jensen CS, Kollios J, Pedersen TB, Timko I (2003) Nearest neighbor queries in road networks. In: Proc. of the 11th ACM int. symp. on advances in geographic information systems, New Orleans, 7–8 November 2003, pp 1–8
14. Jensen CS, Lin D, Ooi BC (2004) Query and update efficient B+-tree based indexing of moving objects. In: Proc. of 30th int. conf. on very large data bases, Toronto, 29 August–3 September 2004, pp 768–779
15. Kolahdouzan MR, Shahabi C (2004) Voronoi-based K nearest neighbor search for spatial network databases. In: Proc. of 30th int. conf. on very large data bases, Toronto, 29 August–3 September 2004, pp 840–851

16. Kollios G, Gunopulos D, Tsotras VJ (1999) On indexing mobile objects. In: Proc. of the 8th ACM SIGMOD-SIGACT-SIGART symp. on principles of database systems, Philadephia, 31 May–2 June 1999, pp 261–272
17. Kim K, Kim S, Kim T, Li K (2003) Fast indexing and updating method for moving objects on road networks. In: Proc. of 4th int. conf. on web information systems engineering, Los Alamitos, 10–12 December 2003, pp 34–42
18. Kwon D, Lee SJ, Lee S (2002) Indexing the current positions of moving objects using the lazy update R-tree. In: Proc. of the 3rd int. conf. on mobile data management, Singapore, 8–11 January 2002, pp 113–120
19. Lee ML, Hsu W, Jensen CS, Cui B, Teo KL (2003) Supporting frequent updates in R-trees: a bottom-up approach. In: Proc. of 29th int. conf. on very large data bases, Berlin, 9–12 September 2003, pp 608–619
20. Mouratidis K, Yiu ML, Papadias D, Mamoulis N (2006) Continuous nearest neighbor monitoring in road networks. In: Proc. of 32nd int. conf. on very large data bases, Seoul, 12–15 September 2006, pp 43–54
21. Nagel K, Schreckenberg M (1992) A cellular automaton model for freeway traffic. J Physique 2:2221–2229
22. Nascimento MA, Silva JRO (1998) Towards historical R-trees. In: ACM symposium on applied computing, Atlanta, 27 February–1 March 1998, pp 235–240
23. Patel JM, Chen Y, Chakka VP (2004) STRIPES: an efficient index for predicted trajectories. In: Proc. of the ACM SIGMOD int. conf. on management of data, Paris, 15–17 June 2004, pp 637–646
24. Pfoser D, Jensen CS, Theodoridis Y (2000) Novel approaches in query processing for moving object trajectories. In: Proc. of 26th int. conf. on very large data bases, Cairo, 10–14 September 2000, pp 395–406
25. Pfoser D, Jensen CS (2003) Indexing of network constrained moving objects. In: Proc. of 11th ACM int. symp. on advances in geographic information systems, New Orleans, 7–8 November 2003, pp 25–32
26. Papadias D, Zhang J, Mamoulis N, Tao Y (2003) Query processing in spatial network databases. In: Proc. of the 29th int. conf. on very large data bases (VLDB), San Fransisco, May 2003, pp 790–801
27. Saltenis S, Jensen CS (2002) Indexing of moving objects for location-based service. In: Proc. of 18th int. conf. on data engineering, San Jose, 26 February–1 March 2002, pp 463–42
28. Speicys L, Jensen CS, Kligys A (2003) Computational data modeling for network-constrained moving objects. In: Proc. of the 11th ACM int. symp. on advances in geographic information systems, New Orleans, 7–8 November 2003, pp 118–125
29. Saltenis S, Jensen CS, Leutenegger ST, Lopez MA (2000) Indexing the positions of continuously moving objects. In: Proc. of the ACM SIGMOD int. conf. on management of data, Dallas, 16–18 May 2000, pp 331–342
30. Shababi C, Kolahdouzan MR, Sharifzadeh M (2003) A road network embedding technique for K-nearest neighbor search in moving objects databases. Geoinformatica 7(3): 255–273
31. Tao Y, Papadias D (2001) The MV3R-tree: a spatiotemporal access method for timestamp and interval queries. In: Proc. of 27th int. conf. on very large data bases, Roma, 11–14 September 2001, pp 431–440
32. Tao Y, Papadias D, Sun J (2003) The TPR*-tree: an optimized spatiotemporal access method for predictive queries. In: Proc. of 29th int. conf. on very large data bases, Berlin, 9–12 September 2003, pp 790–801
33. Theodoridis Y, Stefanakis E, Sellis TK (2000) Efficient cost models for spatial queries using R-trees. IEEE Trans Knowl Data Eng 12(1):19–32
34. Tao Y, Faloutsos C, Papadias D, Liu B (2004) Prediction and indexing of moving objects with unknown motion patterns. In: Proc. of the ACM SIGMOD int. conf. on management of data, Paris, 15–17 June 2004, pp 611–622
35. Vazirgiannis M, Wolfson O (2001) A spatiotemporal model and language for moving objects on road networks. In Proc. of 7th int. sym. on spatial and temporal databases (SSTD), Redondo, 12–15 July 2001, pp 20–35
36. Xiong X, Aref WG (2006) R-trees with update memos. In: Proc. of 22nd int. conf. on data engineering, Atlanta, 3–7 April 2006, pp 22
37. Yiu ML, Tao Y, Mamoulis N (2008) The $B^{dual} - tree$: indexing moving objects by space-filling curves in the dual space. VLDB 17(3):379–400



**Chen Jidong** is a Research Scientist at EMC Research China, one of the research groups at EMC Corporation. He received his bachelor and master from Southwest Petroleum University, and his Ph.D. degree from Renmin University of China, all in computer science. His research interests include personal information management, collaborative knowledge management and mobile information management. He has published over 15 papers in refereed international journals and conference proceedings.

He has held visiting research positions at the PRISM Lab of Versailles University in Paris (2005) and Hong Kong Baptist University (2006). He has served as the local organizing committee of MDM 2008 conference and as a reviewer to journal of Future Generation Computer Systems. He will also serve as the program committee for EDBT 2009 industrial and applications track.

**Meng Xiaofeng** is a full professor at School of Information, Renmin University of China. He received a B.S. degree from Hebei University, M.S. degree from Remin University of China , Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, all in computer science. He is currently the Vice Dean of School of Information and the Head of Department of Computer Science. He is the Secretary General of Database Society of the China Computer Federation (CCF DBS), which is the largest national academic organization in China established in 1977. He has held visiting research position at the Chinese University of Hong Kong (1994,1995,1998), City University of Hong Kong (1997), visiting scientist at the National University of Singapore (2003), and visiting professor at Prism Lab of Versailles University, Paris(2004).

His research interests include query processing, natural language interface, web data extraction, native XML databases, mobile data management. He has published over 100 papers in refereed international journals and conference proceedings.

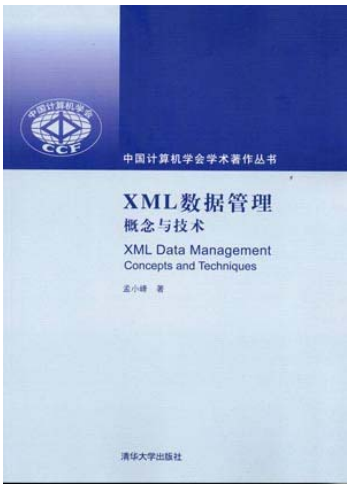
He has served on the program committee SIGMOM, ICDE, ER, DASFAA, MDM, DEXA, WISE, WAIM, etc. and as a reviewer to journals including IEEE Transactions on Knowledge and Data Engineering.

# 科研成果

## 一、学术专著

学术专著名称：《XML 数据管理：概念与技术》

学术专著作者：孟小峰




孟小峰教授所领导的课题组依据 10 年来在 XML 数据管理领域取得的研究成果，完成学术论著《XML 数据管理：概念与技术》。本书从数据库系统实现的角度，全面系统地介绍了纯 XML 数据库系统相关技术。内容涵盖了 XML 数据库存储管理技术（包括存储、编码、索引等方法）；XML 查询处理与优化技术（包括 XML 查询代数、结构查询处理、整体查询处理、查询优化等）；以及 XML 数据管理新技术（包括 XML/Update 处理、访问控制、关键字查询、近似查询处理等）；最后介绍了典 型的 XML 数据库系统和基准测试。

学术专著名称：《移动数据管理：概念与技术》

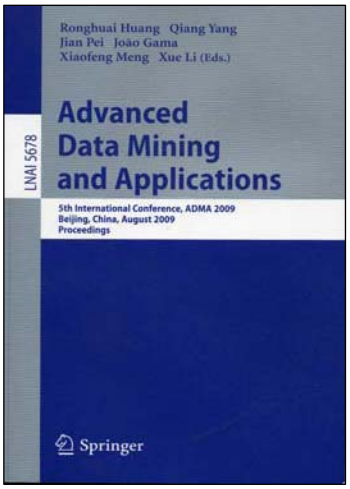
学术专著作者：孟小峰 丁治明


孟小峰教授所领导的课题组依据多年的研究成果，完成学术论著《移动数据管理：概念与技术》。本书总结了国内外有关移动数据管理的研究工作和具有代表性的关键技术，详细介绍了课题组近年来的一些研究成果。内容涵盖嵌入式移动数据库技术（包括移动事务处理、移动数据库复制、移动数据广播等内容），移动对象管理技术（包括移动对象建模、移动对象索引、移动对象更新、移动对象查询、移动对象聚类、移动对象预测、移动数据不确定性研究等内容），位置相关的信息技术（包括位置相关数据的管理技术、空间数据库与交通网络数据库、位置隐私保护等内容）。

## 二、论文集

论文集名称：Advanced Data Mining and Applications

论文集编辑者：Xiaofeng Meng 等




第五届高级数据挖掘及其应用国际会议(ADMA 2009)于 2009 年 8 月 17 日-19 日在北京师范大学举行，该会议论文集由 Springer 出版，孟小峰教授为此次国际会议的联合主席,也是该会议论文集编委之一。

## 三、论文列表

1. X. Pan, X. Meng, J. Xu: Distortion-based Anonymity for Continuous Query in Location-Based Mobile Services. Accepted for publication in the proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009):256-265, November 4-6, 2009, Seattle, Washington.

2. W. Liu, X. Meng, and W. Meng: ViDE: A Vision-based Approach for Deep Web Data Extraction. Accepted for publication in IEEE Transactions on Knowledge and Data Engineering (TKDE).

3. J. Lu, J. Han, X. Meng: Efficient Algorithms for Approximate Member Extraction Using Signature-based Inverted Lists. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009):315-324, November 2-6, 2009, Hong Kong, China.

4. D. Zhou, X. Meng: RS-Wrapper: Random Write Optimization for Solid State Drive. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009):1457-1460, November 2-6, 2009, Hong Kong, China.

5. Y. Li, X. Meng: Supporting Context-based Query in Personal DataSpace. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM2009):1437-1440, November 2-6, 2009, Hong Kong, China.

6. X. Zhang, Z. Wang, J. Ai, J. Lu, X. Meng: An Efficient Multi-Dimensional Index for Cloud Data Management. In Proceedings of the CIKM Workshop on Cloud Data Management (CloudDB2009), November 2, 2009, Hong Kong, China.

7. Y. Li, X. Meng: Exploring Personal CoreSpace for DataSpace Management. In proceeding of 2009 Fifth International Conference on Semantics, Knowledge and Grid (SKG 2009):168-175, October 12-14, 2009, Zhuhai, China.

8. Y. Li, X. Meng, Y. Kou: An Efficient Method for Constructing Personal DataSpace. In Proceedings of the 6th Web Information Systems and Applications Conference (WISA2009), September 18-20, 2009, Xuzhou, China.（获优秀学生论文奖）

9. D. Zhou, X. Meng, Z. Liang: A New Cache Management Approach for Transaction Processing on Flash-based Database. In Proceedings of the 6th Web Information Systems and Applications Conference(WISA2009), September 18-20, 2009, Xuzhou, China.

10. Hui Ding, Nan Yang: An Approach of Community Detecting Based on Block Level Link Analysis. In Proceedings of the 6th Web Information Systems and Applications Conference(WISA2009), September 18-20, 2009, Xuzhou, China.

11. J. Zhou, X. Meng, T. Ling: Efficient Processing of Partially Specified Twig Pattern Queries, Science in China Series F: Information Sciences, Vol.52(10):1830-1847, Oct. 2009. Chinese version: 39(10):1034-1049.

12. J. Chen and X. Meng: Update-efficient Indexing of Moving Objects in Road Networks. Geoinformatica, Vol.13 (4):397-424, September, 2009.

13. J. Zhou, Z. Bao, T. Ling, X. Meng: MCN: A New Semantics towards Effective XML Keyword Search, In Proceedings of the 14th International Conference on Database Systems for Advanced Applications (DASFAA 2009)：511-526, April 21-23, 2009, Brisbane, Australia.

14. F. Jiang, W. Meng, X. Meng: Selectivity Estimation for Exclusive Query Translation in Deep Web Data Integration, In Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA 2009)：595-600, April 21-23, 2009, Brisbane, Australia.

15. C. Zhou, X. Meng: Complex Event Detection in Pervasive Computing. The Third SIGMOD PhD Workshop on Innovative Database Research (IDAR2009), June 28, 2009, Providence, USA.

16. Z.Bao, T.W. Ling, B.Chen, J.Lu: Effective XML Keyword Search with Relevance Oriented Ranking.IEEE International Conference on Data Engineering (ICDE2009): 517-528, March 29- April 3, 2009, Shanghai, China.

17. A. Behm, S. Ji, C. Li, J. Lu: Space-Constrained Gram-Based Indexing for Efficient Approximate String Search. IEEE International Conference on Data Engineering (ICDE 2009):604-615, March, 29- April, 3, 2009, Shanghai, China.

18. J. Lu, Z. Bao, T. W. Ling, X. Meng: XML keyword query refinement. Proceedings of the First International Workshop on Keyword Search on Structured Data.(KEYS 2009):41-42, June 28, 2009, Providence, Rhode Island, USA.

19. Shaoyi Yin, Philippe Pucheral, Xiaofeng Meng, PBFilter: A Sequential Indexing Scheme for Flash-Based Embedded Systems, In Proceedings of 12th International Conference on Extending Database Technology (EDBT2009): 588-599, March 23-26 2009, Saint-Petersburg, Russia.

20. 黄静, 陆嘉恒, 孟小峰: 高效的 XML 关键字查询改写和结果生成技术. 第二十六届中国数据库学术会议论文集：1-7，,2009.10. (第二十六届中国数据库学术会议, 南昌)（获萨师煊研究生优秀论文奖)

21. 王伟, 郭青松, 富丽贞, 孟小峰: 基于代数的 Transform 查询优化策略. 计算机研究与发展, 第 46 卷(增刊)：74-80, 2009.10. (第二十六届中国数据库学术会议, 南昌)

22. 潘晓，郝兴，孟小峰：基于位置服务中的连续查询隐私保护研究. 第二十六届中国数据库学术会议论文集：24-30，2009.10. (第二十六届中国数据库学术会议, 南昌)

23. 艾静, 王仲远, 孟小峰: C-Rank: 一种 Deep Web 数据记录可信度评估方法.计算机科学与探索，第 3 卷，第 6 期：585-593, 2009.11.(第二十六届中国数据库学术会议, 南昌)

24. 周大, 梁智超, 孟小峰: HF-Tree: 一种闪存数据库的高更新性能索引结构. 第二十六届中国数据库学术会议论文集：68-74, 2009,10. (第二十六届中国数据库学术会议, 南昌)

25. 梁智超, 周大, 孟小峰：Sub-Join：一种闪存数据库的查询优化算法. 第二十六届中国数据库学术会议论文集：322-330, 2009,10. (第二十六届中国数据库学术会议, 南昌)

26. 寇玉波, 李玉坤, 孟小峰, 张相於, 赵婧：个人数据空间管理中的任务挖掘策略. 计算机研究与发展，第 46 卷(增刊)：446-452, 2009.10（第二十六届中国数据库学术会议 南昌）

**毕业生学位论文**

1. 周军锋，XML 查询处理关键研究(Key Techniques on XML Query Processing)，中国人民大学，博士生毕业论文，2009.5.8。
2. 姜芳艽，Deep Web 数据集成中查询处理的若干关键问题研究(Key Techniques on Query Processing in Deep Web Data Integration)，中国人民大学，博士生毕业论文，2009.5.8。
3. 贾琳琳，基于字典的近似查询匹配技术研究(Research on Fast Approximate Membership Checking)，中国人民大学，硕士生毕业论文，2009.5.12。
4. 黄静，XML 关键字查询技术研究(Research on XML Keyword Search Techniques)，中国人民大学，硕士生毕业论文，2009.5.12。
5. 朱金清，XML 近似查询处理技术研究(Research on XML Approximate Query Processing Techniques)，中国人民大学，硕士生毕业论文，2009.5.12。
6. 王伟，XML 查询关键技术研究(Research on Key Techniques of XML Query Processing)，中国人民大学，硕士生毕业论文，2009.5.12。
7. 向锂，闪存数据库恢复技术研究(Research on Recovery Techniques for Flash-based DBMS)，中国人民大学，硕士生毕业论文，2009.5.12。

# 四、专利

## 已授权专利

1. 一种基于聚类来监控交通拥堵状况的系统及其方法

   发明名称：一种基于聚类来监控交通拥堵状况的系统及其方法

   申请人：　孟小峰

   专利号：　2008100560991

   申请时间：2008-1-11

   获批时间：2009-9-25

2. 基于视觉的 Web 数据抽取系统和方法

   发明名称：基于视觉的 Web 数据抽取系统和方法

   申请人：　孟小峰

   专利号：　2008100561034

   申请时间：2008-4-3

   获批时间：2009-11-27

## 已申请专利

1. 一种个人数据空间环境下的任务挖掘系统和方法

   发明名称：一种个人数据空间环境下的任务挖掘系统和方法

   申请人：　孟小峰

   申请号：　2009100900362

   申请时间：2009-7-30

2. 一种基于用户特征的个人核心数据空间查询系统和方法

   发明名称：一种基于用户特征的个人核心数据空间查询系统和方法

   申请人：　孟小峰

   申请号：　2009100900358

   申请时间：2009-7-30

## 五、科研项目

**课题来源：国家自然科学基金重点项目**

课题名称：闪存数据库技术研究

课题负责人：孟小峰

课题起止年限： 2009 年 1 月 至 2012 年 12 月

课题简介：

随着电子技术的发展，闪存作为新型存储介质被广泛应用于移动通信、工业控制、航空航天、笔记本电脑等嵌入式系统和便携式设备上。伴随着闪存容量的快速增长，闪存上的数据管理问题已成为新的挑战性问题。从数据库技术的角度研究闪存数据管理，构建新型的闪存数据库系统，对于促进闪存数据管理理论与闪存应用的进一步发展具有重要的理论意义和实用价值。本课题从闪存的器件特性入手，针对闪存应用的数据存取特点和现有数据库技术在闪存数据管理上的缺陷，从系统性和通用性角度研究闪存数据库的基本理论和设计方法，重点突破闪存数据库的体系结构、存储管理、查询处理、索引等关键问题，建立全新的闪存数据库系统理论和方法体系。课题从 DBMS 角度阐明闪存数据管理的特性，从高性能、高可用性、可剪裁性等方面构建闪存数据库的理论框架，为闪存数据库的进一步研究与应用提供理论方法和技术支撑，为数据库理论和技术的发展提供新思路。

**课题来源：国家高技术研究发展计划(863 计划)重点项目**

课题名称：普适计算基础软硬件关键技术及系统-隐私保护技术

课题负责人：杨楠 孟小峰

课题起止年限： 2009 年 3 月 至 2010 年 12 月

项目简介：

普适计算是一种超越桌面计算的新型计算模式，目标是在计算和通信无所不在的条件下建立以人为中心的计算环境——这种计算环境紧密结合于人们的生产生活环境，通过自然便捷的交互方式主动提供人们适用的信息服务，以辅助人们的活动。当前，信息技术日益普及，计算设施的泛在性逐渐显现，本项目的总体目标是研究分析制约普适计算技术和产业发展的关键技术问题，着重研发泛在设备自发互联互通的框架与协议，研发以人为中心的普适服务技术与系统，建立支持我国普适计算研究和相关产业发展的基础技术体系，项目研发产生的标准、协议和软硬件技术成果将促进相关产业的可持续发展。本项目组织产学研有机结合的研究团队，突破普适计算环境中设备、软件和用户三个层面上的基础性关键技术，构建支撑系统，并通过应用示范验证关键技术的有效性和促进产业进步。 WAMDM实验室负责项目中的异变环境中的数据管理技术和隐私保护匿名模型、算法和评价标准等关键技术。

**课题来源：IBM 大学共享研究项目（IBM SUR 项目）**

课题名称：Cloud-based Database Systems（云数据库系统）

课题负责人：孟小峰 陆嘉恒

课题起止年限：2009 年 9 月 至 2010 年 9 月

课题简介：

云计算是当今信息产业最受关注的一种计算模式，在这种模式下，企业和个人可以根据自己的需要购买存储设备和计算能力，而不是花费大量资金购买大规模高性能计算机。作为云计算的一项关键技术，云数据存储和云数据管理为业界带来巨大的潜在商用价值。随着信息产业的发展，企业和公司产生的数据量快速增长，通常数据规模可以达到TB甚至PB级别。如何管理和分析海量数据是目前很多领域所面临的问题，例如在医疗、通信和互联网领域。云环境是由大量的性能普通、价格便宜的计算节点组成的一种无共享大规模并行处理环境，所以从成本和性能两方面考虑，越来越多的企业更愿意把自己的数据中心从昂贵的高性能计算机转移到共有或私有云环境中。我们的项目主要致力于解决云计算环境下数据库的技术难点，实现一种具有高可用性、高容错性、可扩展性和高性能的云数据库系统。该项研究课题的目标是为了解决下一代海量分布式“大数据”(Big Data)的管理问题，从而支持下一代大数据应用系统以及大规模分析处理系统。在研究的第一年，将主要侧重于解决云计算平台上数据模型、存储与检索、同步与容错等一系列关键性问题。

**课题来源：国家自然科学基金**

课题名称：移动环境中关键词搜索的关键技术研究

课题负责人：陆嘉恒

课题起止年限：2010 年 1 月 至 2012 年 12 月

课题简介：

随着社会的不断发展，人们的移动性日益增强，对信息的需求也日益高涨，为有效地满足人们这种需求，在移动环境中搜索技术正逐步兴起。这种新兴的搜索是搜索技术在移动平台上的延伸，通过PDA、手机等移动通信终端，以短信、WAP 上网、语音通话等方式接入移动通信网络，来获取WEB、WAP站点信息、本地信息等信息服务。移动环境中搜索技术的出现，真正打破了地域、网络和硬件的局限性，满足了用户随时随地的搜索服务需求。而且搜索市场广阔，据统计，移动用户近是互联网用户的十倍。同时，移动增值服务业务的快速成长、互联网搜索技术的不断成熟以及3G 带来移动网络带宽为在移动环境中搜索行业的发展提供了机遇。因此，作为互联网搜索技术与移动通信技术相结合的产物，近年来，移动搜索技术在日益走向成熟，应用前景十分看好。本课题提出的“移动环境中关键词搜索的关键技术研究”，旨在研究面对移动环境中，关键词检索的一些关键技术，主要包括，（1）定义高效的关键词检索语义。搜索结果简约，只有若干条，且高度符合用户需求；(2)关键词的近似查询考虑同音替换和同义替换以包容数据输入中的错误；(3)在移动环境下考虑位置的相关性，解决位置表示模型框架下的数据表示、存贮与索引，最近邻居查找的查询处理等关键技术问题。三者结合提供高效准确的移动搜索技术。 我们的目标是基于可行的技术建立一个创新的面向移动环境中关键词检索原型系统。这个系统应该能够根据用户的位置，包容用户数据中的错误，实现近似查询。

**课题来源：国家高技术研究发展计划(863计划)**

课题名称：XML数据的复杂查询与模糊结构关键字检索

课题负责人：陆嘉恒

课题起止年限：2009 年 9 月 至 2010 年 12 月

课题简介：

可扩展标记语言（eXtensible Markup Language，XML）是 Internet 上一种新的数据表示、存储和交换标准。XML 推荐标准 1.0 版发布于 1998 年 2 月，之后迅速在全球掀起了 XML 应用的浪潮。XML 数据查询和检索在近十年以来，一直是数据库研究领域中的一个热点问题。虽然在国际和国内，企业界和学术界都投入了大量人力物力进 行技术攻关，并开发出了一些原形系统，但是海量 XML 数据的高效复杂查询和检索问题仍远远没有得到完美解决，功能瓶颈和性能瓶颈两大问题仍大大制约着 XML 数据的广泛应用。所以我们相信对于 XML 数据的复杂查询和高效检索的研究，具有巨大的理论研究必要和现实应用价值。 我们希望通过该课题的研究，在下一代 XML 数据管理技术方面有所突破，发表国际一流的学术论文，并结合我国的实际情况和产业需求，开发出具有自主知识产权的先进的 XML 查询和检索系统。不断提升我国数据库应用的水平，使之更好地为我国信息化建设服务。

# 学术交流

## 一、学术活动任职

**Prof. Xiaofeng Meng:**

**DB Vice Co-chair,** ACM 18th Conference on Information and Knowledge Management **(CIKM2009),** October 27-31, 2009, Hong Kong

**Program Committee Co-Chair,** The Fifth International Conference on Advanced Data Mining and Applications(ADMA2009), August 17-19,2009**,** Beijing

**General Vice Chair,** 25th International Conference on Data Engineering( **ICDE2009** ), March 29 - April 4, 2009, Shanghai, China

**Workshop Chair**, The 3rd International Workshop on Privacy-Aware Location-based Mobile Services (PALMS2009), May 18-21, 2009,Taipei, Taiwan

**Regional Chairs**, The 14th International Conference on Database Systems for Advanced Applications(DASFAA2009), April 21-23, 2009, Brisbane, Australia

**Program Committee member,** MDM2009 The 10th International Conference on Mobile Data Management: Systems, Services and Middleware(MDM2009), May 18-21, 2009,Taipei, Taiwan

**Program Committee member,** PAKDD2009 The 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining. 27-30 April 2009 Bangkok, Thailand

**Program Committee member,** MCPC 2009 The international Conference on Mobile Communications and Pervasive Computing (MCPC 2009), 23-25 March 2009 , Leipzig , Germany

**Program Committee member,** DEXA2009 The 20th International Conference on Database and Expert Systems Applications(DEXA2009), 31 August - 4 September 2009, Linz, Austria

**Program Committee member,** APWeb-WAIM2009 The joint conference on Asia-Pacific Web and Web-Age Information Management(APWeb/WAIM 2009), 1-4 April, 2009, Suzhou, China

**Program Committee member,** WI2009 IEEE/WIC/ACM International Conference on Web Intelligence (WI2009), 15-18 September, in Milano, Italy

**Dr. Jiaheng Lu**

**Program Committee member,** APWeb-WAIM2009 The joint conference on Asia-Pacific Web and Web-Age Information Management(APWeb/WAIM 2009), April 1-4, 2009, Suzhou, China

**Program Committee member，** EDT2009 The First International Conference on Emerging Databases(EDT 2009), August 27-28,2009, Bexco(Pusan),Korea

**Program Committee member,** IDAR2009 The 3rd SIGMOD PhD Workshop on Innovative Database Research(IDAR2009), June 28, 2009,Providence,Rhode Island,USA

**Program Committee member,** WISE2009 The Tenth International Conference on Web Information Systems Engineering(WISE2009), October 5-7, 2009 ,Poznan,Poland

**Local Arrangement Co-Chairs,**CloudDB2009 The First Internationa Workshop on Cloud Data Management(CloudDB2009), october 19, 2009, Hong Kong

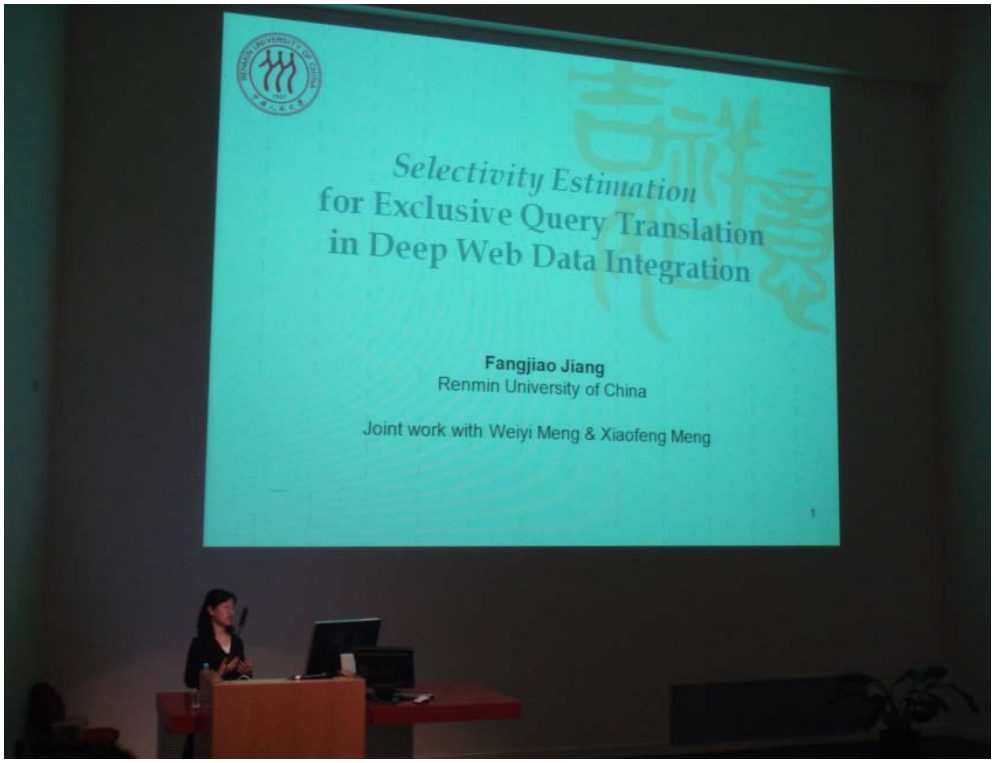
## 二、学术交流

**2009.11.1--2009.11.7，孟小峰教授与实验室 4 位研究生参加在香港举办的信息与知识管理国际会议（ACM CIKM2009），担任 DB Vice Chair。**

孟小峰教授与实验室同学在 CIKM2009 合影

**2009.4.21--2009.4.23，孟小峰教授与实验室 2 位研究生参加在澳大利亚布里斯班举办第 14 届数据库系统与高级应用国际会议（DASFAA2009）。**

孟小峰教授、Ling Tok Wang 教授、周军锋在 DASFAA09 合影




姜芳艽在 DASFAA09 上做学术报告

**2009.4.1--2009.4.5，孟小峰教授参加在上海举办的第 25 届国际数据工程大会（ICDE2009），担任大会副主席。**


孟小峰教授参加 ICDE 2009


实验室李玉坤和参会人员讨论

**2009.3.23--2009.3.26，孟小峰教授参加在俄罗斯彼得堡举办的国际数据库技术会议（EDBT2009）。**

孟小峰教授、韩家炜教授及学生在 EDBT2009 合影

**2009.12.3-2009.12.4，孟小峰教授在武汉大学参与成立 ACM SIGSPATIAL 中国分会。**

ACM SIGSPATIAL 中国分会成立合影

**2009.12.18，孟小峰教授参加 ACM SIGSPATIAL 中国分会举行 YOCSEF（中国计算机学会青年计算机科技论坛）报告会，并做了题为《基于位置服务中的隐私保护》的报告。**


孟小峰教授与谢幸博士合影

**2009.10.15-10.18，孟小峰教授和实验室几位研究生参加了在江西南昌举办的第二十六届全国数据库学术会议（NDBC2009）。**


孟小峰教授在 NDBC2009 上做报告

**2009.9.26-9.28，实验室三位同学在中国徐州参加了第六届全国 Web 信息系统及其应用学术会议（WISA2009）**


实验室三位博士生在 WISA2009 合影

**2009.7.7，孟小峰教授作为学术主任组织了首期的中国计算机学会（CCF）创建的《学科前沿讲习班》，并在讲习班上做了“网络与移动数据管理”的报告。**


中国计算机学会《学科前沿讲习班》第 1、2 期讲师、学员合影

**2009.6.5-2009.6.6，孟小峰教授在成都参加中国数据库发展研讨会暨 NDBC2009 审稿会。**

孟小峰教授在中国数据库研讨会上做报告

## 三、学术报告

**2009 年 12 月 29 日，应孟小峰教授邀请，微软亚洲研究院王海勋博士到信息学院做学术报告，并受聘为信息学院兼职教授。**

12 月 29 日，信息学院在信息楼 4 层学术报告厅举行了微软亚洲研究院王海勋博士受聘中国人民大学兼职教授仪式。随后，王海勋教授作了题为 “Relational DBMS for Cloud Computing: An Eventual Consistency Approach”的学术报告。中国人民大学副校长冯惠玲教授为王海勋教授颁发了聘书。聘书颁发仪式由信息学院副院长孟小峰教授主持。信息学院院长杜小勇教授、陆嘉恒副教授等多位老师出席了仪式。冯惠玲副校长为王海勋教授颁发了聘书，并代表学校，欢迎王海勋教授加盟中国人民大学。

在聘书颁发仪式之后，王海勋教授作了题为“Relational DBMS for Cloud Computing: An Eventual Consistency Approach”的学术报告。王海勋教授介绍了当今云计算发展所带来数据管理的挑战性问题，以及为应对此挑战学术界所提出的两套技术路线：“Key/Value Pair”的数据管理方法，以及类似于微软所推出的 SQL Azure 的云数据管理方法。王海勋教授主要介绍了在后一种基于 RDBMS 的云数据库系统中所面临的数据一致性问题，以及他们所提出的在保证“最终一致性”以及“保留因果关系”的特性下，最大化并发性的算法。在他们的实验中，证明了此方法能够直接构建在现有数据库管理系统之上，从而避免修改数据库引擎，并且能够在云计算平台上的数据管理中，取得高并发行以及高可用性的结果。在学术报告的过程中，王海勋教授还详细回答了老师和同学们提出的各种问题。


王海勋博士受聘中国人民大学兼职教授

**2009 年 12 月 24 日，应孟小峰教授邀请，微软亚洲研究院谢幸博士到实验室做学术报告。**

谢幸博士做了题目为“Build Intelligence from the Physical World”的学术报告，这是实验室策划的“移动和云计算主题系列学术报告”的第二讲。谢幸博士介绍了微软亚洲研究院关于移动数据库的研究项目，介绍了一些新的研究热点，引起了同学们热烈的讨论。


孟小峰教授向谢幸博士赠送纪念品

**2009 年 10 月 11 日，美国纽约州立大学 Binghamton 分校孟卫一教授访问 WAMDM 实验室。**

在和孟卫一教授的交流中，Web+Mobile 小组的同学汇报了课题“面向移动用户的 Web 数据集成系统”的研究进展。实验室的赵婧同学做了“结构化 Snippet 抽取”的报告，胡享梅同学做了“Web 数据库选择”的报告。孟卫一教授分别给两位同学的研究工作做出了指导。此外，孟卫一教授在选题和研究方法上给实验室同学们做了指导。

**2009 年 9 月 27 日，应孟小峰教授邀请，AT&T 的董欣博士到实验室做学术报告。**

董欣博士做了题为“Data Integration with Uncertainty”的报告。在这次报告中，董欣博士首先分析和讨论了各种数据管理应用中的实际问题，然后讨论了不确定数据集成中存在的挑战性问题，以及介绍了现有的一些解决方法，同时在报告中介绍她们开发的一个 self-configuring 数据集成管理系统。其中还重点讨论了数据集成中的 probabilistic schema mappings, probabilistic mediated schemas, semantic schema mapping 和 probabilistic functional dependencies 等问题。

孟小峰教授向董欣博士赠送纪念品

**2009 年 4 月 7 日至 4 月 10 日：应孟小峰教授邀请，加拿大 Simon Fraser University 的裴健副教授到实验室讲授 “不确定性数据研究”的课程。**

裴健老师首先通过大量实例详细而生动地分析和讨论了什么是不确定数据，不确定数据的来源，以及分析和讨论不确定数据的必要性，并介绍了在不确定数据研究中的重要数学工具 Density estimation。然后，裴健老师和同学们探讨了不确定性数据的建模方法和两种常用的模型（Uncertain object model 和 Probabilistic table model）。接下来，他讲解了如何在不确定数据上做简单 SQL 查询、join、ranking 以及 spatial 查询和建立索引的问题。最后，他就不确定性数据的挖掘方法进行了深入讲解，包括 OLAP、频繁模式挖掘、聚类和检测等，并且探讨了不确定数据领域存在的挑战和未来的研究发展方向。

**2009 年 6 月 3 日：应孟小峰教授邀请，香港科技大学陈雷博士到实验室做学术报告：“Efficient Query Processing over Uncertain Databases”。**

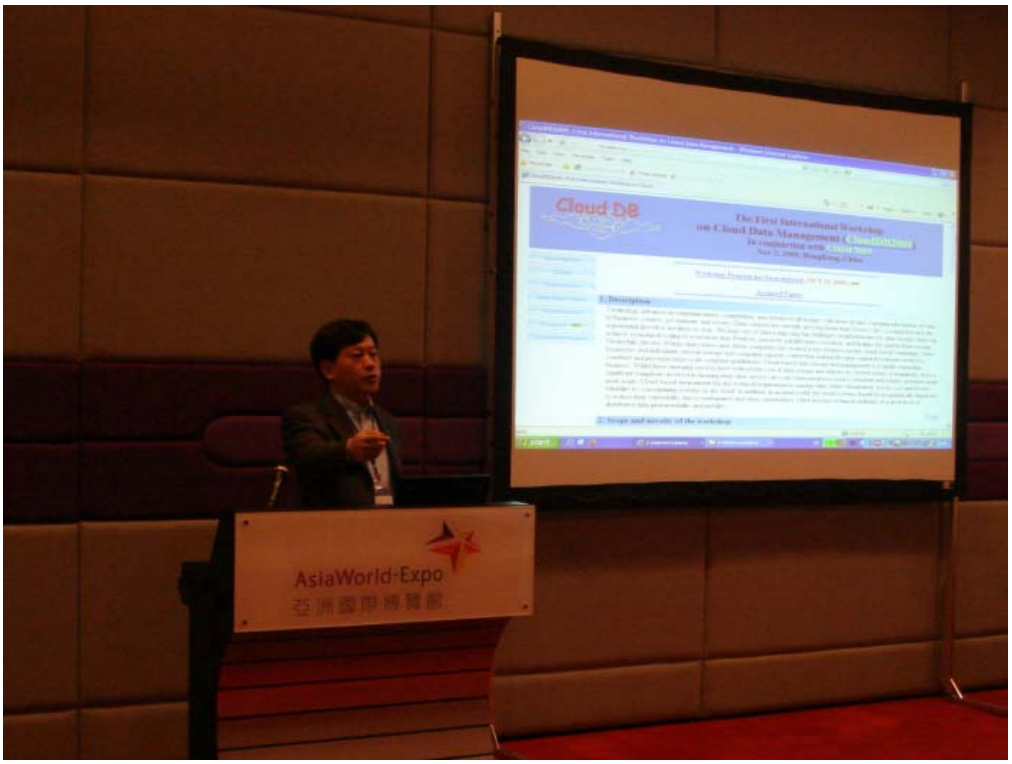
陈雷博士首先分析和讨论了研究不确定数据的必要性，介绍了一种不确定性数据的建模方法，并且基于这种模型进行了多种类型的查询研究，包括不确定性数据上的预计算查询，RNN(reverse nearest neighbor)skyline 查询，ranked 查询，以及 group nearest neighbor 查询等。主要讨论了一些 pruning 和 indexing 的技术和算法。最后，陈雷老师给出了不确定数据领域存在的挑战和未来的研究发展方向。同时陈雷博士还就自己的体会谈了高水平论文的发表之道。


陈雷博士在 WAMDM 实验室做学术报告

## 四、举办会议情况

### 1、第 18 届信息与知识管理国际会议上举办了首届云计算 Workshop--The First International Workshop on Cloud Data Management

第 18 届信息与知识管理国际会议 CIKM2009 于 2009 年 11 月 1 日至 11 月 7 日在中国香港召开。WAMDM 实验室承办了本次会议的 workshop，孟小峰教授担任本次 workshop 的联合主席。这是国际上第一个云计算相关的 workshop，本次会议主题是：基于云计算平台的大规模数据管理，主要包括：大规模数据管理系统设计、并行算法以及云计算系统的应用等方面，从而最大化地提高大规模数据管理系统的性能、减少成本。本次 workshop 汇集了中国人民大学、复旦大学、华东师范大学等知名大学以及 EMC、IBM 等主流研究机构的科研成果，最终共录用 5 篇长文，3 篇短文。会议期间，实验室的张相於同学做了“An Efficient Multi-Dimensional Index for Cloud Data Management”的报告，受到与会人员的广泛好评。




孟小峰教授主持 workshop on cloudDB

### 2、首届中国云计算研讨会

首届中国云计算研讨会（China Cloud Computing Symposium，CCCS2009）于 2009 年 10 月 26 日在北京召开，本次会议由中国计算机学会和国际商业机器（中国）有限公司主办、北京大学、中国人民大学、电子工业出版社协办。

本次研讨会采用主论坛报告、分论坛演讲和技术展示相结合的方式，邀请著名云计算学者和专家从不同角度对云计算的实质及发展趋势进行报告、交流和讨论。研讨会期间还举办与云计算相关的前沿研究展示和先进技术演示等活动。

孟小峰教授应邀担任本次会议的程序委员会共同主席，并主持分论坛的演讲和讨论。

参加会议的有来自国内外的 100 多位学者。

### 3、第二届 Flash 数据库系统 workshop

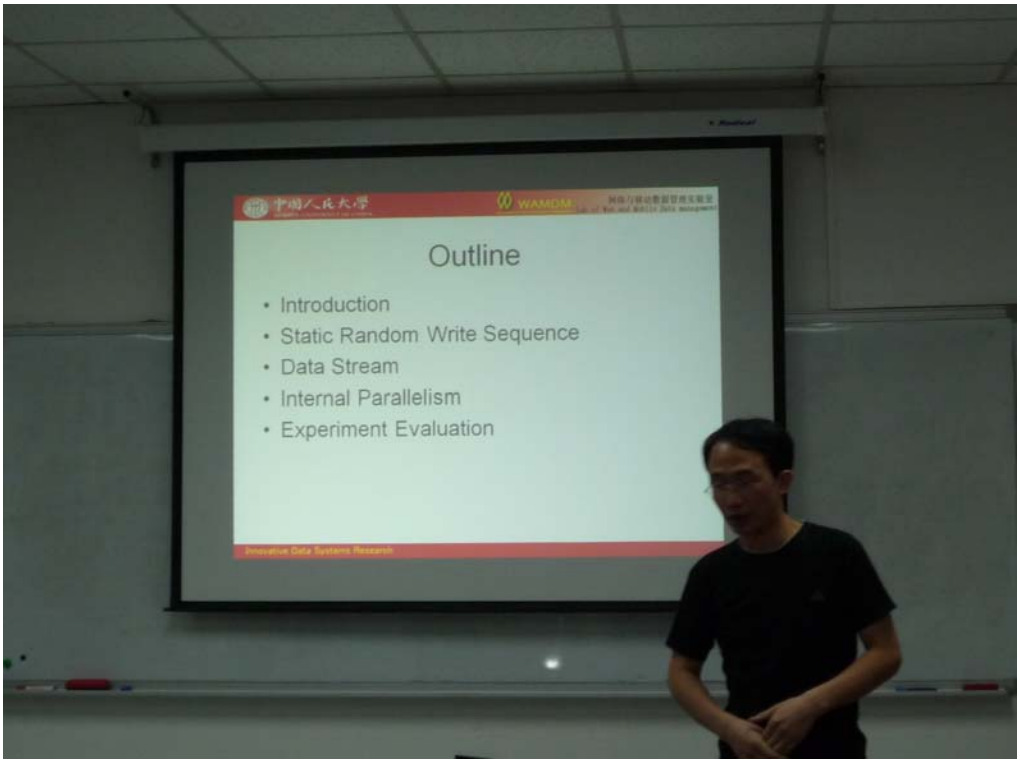
为了探讨闪存数据库存储管理、缓冲区管理、索引等关键问题，以孟小峰教授为负责人的课题组（获国家自然科学基金重点项目“闪存数据库技术研究”（NSFC Key Project：60833005）的资助）于 2009 年 6 月 1 日在中国人民大学举办了第二届闪存数据库系统专题讨论会。

与会人员有来自于中国人民大学的孟小峰教授、杨楠副教授、路嘉恒副教授、中国科技大学的岳丽华教授、香港浸会大学的徐建良副教授及三所高校相关的研究生。本次研讨会还就目前的三种不同的闪存实验环境进行了专门的讨论。

参会人员合影

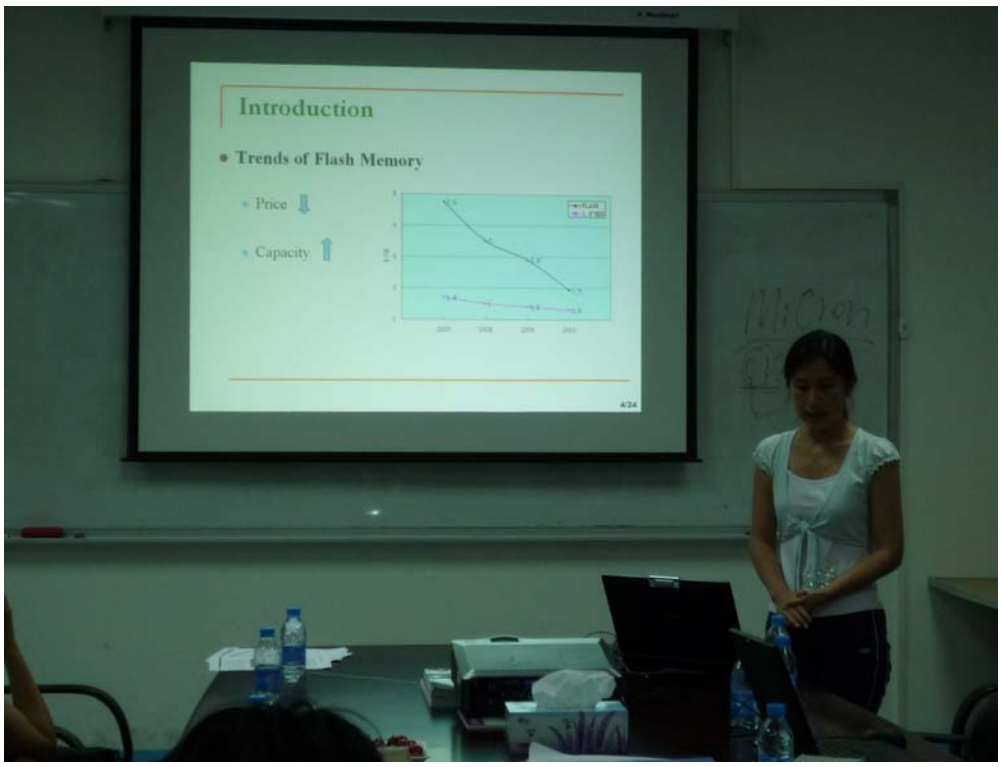
会议包含以下几个报告：1）存储管理：SSD 随机写优化；2）缓冲区管理：基于代价的缓冲区置换策略以及基于数据冷热进行置换的策略；3）存储介质问题：闪存存储板设计；4）索引问题：延迟更新索引策略。WAMDM 实验室有两名同学在研讨会上做了报告。

周大同学作了题为“SSD 随机写优化方法”的报告，该方法通过将随机写转化为随机读和连续写的方式，从而从根本上避免了 SSD 的随机写。




**周大做报告**

实验室的汤显同学作了题为“基于代价的缓冲区管理策略”的报告，提出了一种新的基于 Flash 的缓冲区管理策略——CBLRU。




**汤显做报告**

本次研讨会对于项目以后的研究会有很大的推动作用，但是仍有许多未知的领域需要去探讨和研究，这就需要同学们投入更多的研究工作和热情。

**4、第三届 Flash 数据库系统 workshop**

第三届闪存数据库系统研讨会（The 3rd Workshop on Flash-based Database Systems）于 2009 年 12 月 13 日在合肥中国科学技术大学举行，这是在以中国人民大学孟小峰教授为负责人的国家自然科学基金重点项目“闪存数据库技术研究”的支持下创立的学术交流平台，也是课题组探索的一种新的课题组织方式。

研讨会主要与会人员包括课题承担单位中国人民大学的孟小峰教授课题组、中国科技大学的岳丽华教授课题组、香港浸会大学的徐建良副教授课题组，并特别邀请百度公司参加。

全体与会人员合影

本次研讨会主要报告了课题组在闪存数据库存储管理、缓冲区管理、查询处理和事务处理，以及闪存开发板、闪存硬件测试等方面的最新研究进展和技术成果。同时，研讨会还就将来硬件的发展对数据库系统带来的变化进行了专门的讨论，百度公司技术人员报告了利用闪存技术提升数据库新能的经验，我校柴云鹏博士提出了在云存储中利用闪存提升性能和节省能源的研究课题。课题组的研究表明，目前闪存对现有数据库的性能提升在 10 倍左右，课题组的研究目标是将这一性能再提升 5-10 倍。

# 暑期研讨

**2009 年 8 月 1 日-8 月 2 日，WAMDM 实验室全体师生赴秦皇岛燕山大学举办了第一届暑期研讨会。**

WAMDM 实验室全体师生于 2009 年 8 月 1 日至 8 月 4 日在美丽的海滨城市秦皇岛成功的举办了第一次暑期研讨会。在这次研讨会中，这次研讨会有两个主要目的：首先是与燕山大学信息学院的郭景峰教授及其弟子进行深入的学术研讨；其次，经过了一个学期辛苦的学习，希望在这里让大家放飞一下心情，为新的学期积蓄更多的能量。

在这四天里，我们与燕山大学信息学院的师生进行了一整天充分而深入的学术交流，内容涉及到个人数据管理、数据空间、数据集成、图数据处理、隐私保护和闪存数据管理等方向的关键问题，充分体现了当前国际上数据管理的新思路和新方法，为数据库的进一步研究与应用拓宽了视野，为数据库理论和技术的进一步发展提供新思路。学术研讨我们采用学术报告与集中讨论等多种形式。

我们实验室的李玉坤、张相於、寇玉波、王仲远、潘晓六位同学做了精彩的报告，内容如下：

1）个人数据空间技术研究回顾：数据空间的由来和面临的问题等等；

2）OrientSpace 开发进展报告：OrientSpace 的设计理念和功能介绍等；

3）大型稀疏图的 Top-K 查询：社会网络中的子图搜索等；

4）舆情监控系统：监督分析网络舆情，建立起一个畅通的网络沟通平台；

5）闪存数据管理：针对缓冲区管理的问题，提出基于代价的缓冲区管理策略；

6）隐私保护：针对路网中的隐私问题，提出增量式的基于团的匿名方法 ICliqueCloak；

WAMDM 实验室的全体师生同兄弟院校的师生对报告的内容，以及目前数据管理的新方法和新思路进行了深入的讨论，在讨论的过程中我们进一步加深了相互的了解，为进一步合作奠定了基础。

潘晓同学与燕山大学信息学院同学交流

孟小峰教授与燕山大学的任家东教授交流

杨刚老师与同学讨论交流

研讨会过后，实验室全体师生和燕山大学的同学们到秦皇岛连峰山公园、祖山公园和鸽子窝公园游玩。在青山绿水之间，同学们放松了心情，也展现了活跃、可爱的一面。

8 月 2 日，实验室师生们游览了祖山。祖山号称燕山之宗祖，百岭之发端，其最高峰“天女峰”海拔 1424 米。沿途美景不断，山谷两侧，峰峦叠嶂，绵延不绝，山峰错落有致，层次分明，一步一景，景景不同，每一座山峰都如刀削斧劈一般，崖岸危岩，高不可攀。在美景之中，同学们心情格外放松，流露着自然的心性。

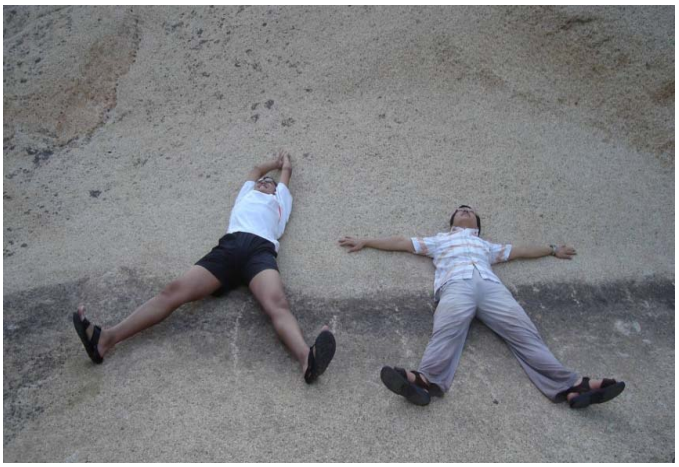
实验室全体师生与燕山大学师生合影

孟小峰教授和杨刚老师

杨刚老师与陆嘉恒老师

相於版山寨“如来佛”

希望，就在前方



不管到哪里，我们都是“人大人”

山路崎岖，风光独好

肩并肩，手牵手

8 月 3 日-8 月 4 日，同学们游览了连峰山公园和鸽子窝公园。联峰山公园里树木森森，浓荫遮天，点缀有多个庙宇、佛洞。相比山险景美的祖山，连峰山的美景更显温和。鸽子窝公园是秦皇岛著名景点，紧邻大海。


实验室师生在连峰山的合影

祝实验室在新的一年里蓬勃发展，更上一层楼！

# 附录

# 王仲远接受《中国人民大学》校报和《信息月刊》采访

## 《中国人民大学》校报采访稿：

简介：王仲远系我校信息学院计算机应用技术专业07级硕士研究生，曾获2007年ACM SIGMOD Undergraduate Scholarship国际奖学金（全球共7人）。他在本科及研究生阶段共获计算机世界奖学金、中国惠普优秀学生奖学金等国家及省部级奖励3次；光华奖学金、校级优秀毕业生等校级荣誉与奖励10次；以及院系级荣誉奖励3次。他领导开发的计算机领域中文文献集成系统C-DBLP一年多来访问量已经超过150万人次，在国内计算机研究领域具有相当大的影响力。他在《中国计算机报》上组织刊登《云计算时代的数据管理》专题文章，受到广泛关注。他也将成为1999年人民大学成立计算机科学与技术系以来首位进入微软亚洲研究院工作的学生。

# 仲子说：做一名自豪的人大IT人！

文/赵晓丽　王鲲鹏

走进理工配楼一层，便可清楚地看到信息学院数据工程与知识工程教育部重点实验室下属的网络与移动数据管理（WAMDM）实验室。在这个走廊里贴满了研究成果介绍、学术氛围浓厚的实验室里，承载了王仲远最近这四年来的欢笑与汗水，也见证了他与信息学院共同成长进步的步伐。

## 多线程处理器，与信息学院共同运转

王仲远自大二暑假起便进入孟小峰教授所领导的网络与移动数据管理实验室，其主要研究方向为计算机领域前沿课题：Web数据管理研究及云计算平台上的数据管理问题研究。

初入实验室的王仲远未曾料到自己后来的成就，当时的他只是抱着一颗平和的心，努力吸取知识，做好每一份工作。但就是这些一步一个脚印、扎扎实实的工作，使得他迅速成长。实验室一直追踪着世界范围内最前沿的研究课题，最初参加每周六上午的研讨会时，王仲远对这些前沿课题的报告内容并不能完全理解。但经过长期的训练及自身的努力，他慢慢进入状态，融入到实验室的大环境中。大四之后，他便为实验室做一些具体的工作，包括系统的开发，对课题提出自己的解决办法等等。而在世界范围内都非常前沿的云数据管理问题研究，也是由他在周六的研讨会上最初介绍并逐步引入到实验室的课题研究当中。

网络与移动数据管理实验室在孟小峰教授的倡导下，有着非常鲜明的研究风格：那就是既做前沿的研究，也做创新性的系统，并且所作的研究要反映到系统上来，以帮助人们解决一些实际的问题。在这种思想的影响下，王仲远在大四的时候，便在研三师兄的指导下，开发了自己的第一个系统，名字为JobTong。这个系统可以自动从中华英才网、51job、智联招聘以及各高校就业信息网上抽取集成招聘信息，并重新组织，从而为应届生提供一个快速浏览信息的统一入口。这便是他前几年一直研究的一个课题：Web数据集成。在不到两个月的时间完成该系统核心功能的开发后，经过不断地完善，系统的数据量很快突破了百万。从最初的原型系统，到完善发展为一个对外提供集成信息的网站，使得王仲远在其自身的能力以及编程实践方面有了非常迅速的成长和提高。

源于斯，而高于斯。在研究生期间，他们利用这个系统的核心模块，通过不断地完善，在其他各个领域开发了多个系统。例如学术领域的计算机中文文献集成系统C-DBLP，网络购物领域的图书价格比较网，以及与我校新闻学院合作开发的舆情监控平台。

王仲远也凭借此课题研究荣获由美国计算机协会 ACM 颁发的全球仅有 7 名获奖者的”SIGMOD Undergraduate Scholarship”奖学金。

自 08 年以来，王仲远同时关注另外一个研究课题：云数据管理研究。由于在国内较早地开展相关技术的研究，我校信息学院已就此方面与中国移动、华为、联想、IBM、EMC 等公司进行多次交流，并最终选择在云计算领域最具实力之一的 IBM 合作，共同开展云计算平台上的数据库管理系统的核心技术研究。此项研究也获得了 IBM 共享大学研究(Shared University Research 简称 SUR)项目的资助。

由于实验室众多研究项目的开展，系统的开发，以及他在学院里所承担的各项学生工作，王仲远常常忙得分身乏术。许多事情常常同步进行，他形象地把自己描述成类似于计算机中的“多线程处理器”。而这个“多线程处理器”的运转，其实也见证信息学院近些年来所取得的长足发展。

## 所谓成长，创新与自信

在谈到创新话题时，王仲远不由自主笑了起来，带着孩童般狡黠的智慧。“我从小学开始，一直到现在，都喜欢看郑渊洁的童话故事。”在郑渊洁的童话中，总会强调着想象力、开阔思维的重要性，或许王仲远的创新能力没有被禁锢，有一部分原因便是由于他一直喜欢读郑渊洁的童话。但他谈到，创新其实并没有那么遥远，因为我们是站在巨人的肩膀上向前发展。“当面临问题时，一种方法不能解决，我们就会去不断尝试，想更好的方法。而当这个问题解决时，便是创新的突破。”

科研的工作就是如此，并非只有特别聪明的人才做得到。王仲远谈到自己近几年获得的成果也是通过自己的努力取得的，他放弃了大部分的周末，五一长假甚至暑假的休息。“每个人面临的时间是一样的，要想比别人做得好，只有通过更多努力。”

从进入大学伊始的放平心态，到通过一步一步努力而后发现自己取得的成绩，这都让王仲远增加了更多自信。“当我所开发的系统能够服务到更多人时，当周围的老师同学对我一次次地肯定时，我便慢慢自信起来。”

王仲远也谈到自信心培养的重要性。他在大学学习生活中培养起来的自信，在他的求职过程中给予了他极大的帮助。在他面试过的近十个公司中，所有的面试他都顺利通过，“当我参加微软亚洲研究院、李开复的创新工场、IBM、百度等公司的面试时，我的诚恳、自信给予了我极大的帮助，面试官们能够从谈话中感受到我的自信与能力，自然愿意给我更多的机会”，最终，王仲远拿到了多家顶级互联网、IT 公司的 Offer，并主动放弃了许多公司的面试、笔试邀请，从而留给其他同学更多的机会。

然而，自信并并等于自负。王仲远强调说自己的自信并不盲目，他非常信奉两句话，“吾日三省吾身：为人谋而不忠乎？与朋友交而不信乎？传不习乎？”以及“一个聪明人并非不犯错误，但他不会在同一个地方犯相同的错误。”正是如此恰当的自信，让他在学习生活中得意，在工作实践中奋进，在学术科研道路上大步向前。

## 所谓青春，奉献与感恩

进入大学伊始，王仲远便参加了校青协以及院青协，参与各种志愿服务及社会实践活动。而对于其以后的科研创新成果，如工作通数据集成系统，舆情监控平台等，也都以公益性质的成果提供给大家使用，方便网络用户获取有效信息。此外，他还利用自己的技术能力，为实验室、计算机系、数据库专委会建设与维护网站，“我希望可以通过自己的技术，来帮助更多有需要的人”。

在各种学生工作中，他也同样出色，无私贡献。这些年来，他担任过班长、辅导员、党支部书记以及信息学院学生党总支组织委员等职务，“最让我感到自豪的是，通过我与其他同学的共同努力，我所在的班级及党支部都获得了多种荣誉称号，这是对我工作的最大肯定。”在王仲远担任辅导员的两年工作中，针对刚进入大学的新生，他通过班级活动等多种形式对同学进行正确的专业观念培养、心理转型教育、良好班风、学风的建立；针对即将毕业的高年级同学，他根据高年级学生自主独立性较强、个人事务较多的特点，摸索出依靠班委、党支部，依托网络交流平台，以及点对点辅导的模式，有效地开展辅导员工作。在学年末的总结会上，他经过票选，荣获了信息学院优秀辅导员称号。

王仲远谈到，其实最初人大计算机系并非自己的首选专业，但几年之后他便彻底爱上了这个专业。这些得益于很多贵人的帮助，包括学院的领导老师、班级的同学，尤其是他的导师孟小峰教授。孟小峰教授对他在学术辅导以外，在人生发展的规划、为人处世的准则等方面都给予了他许多指导。“导师讲到的做事方法、做人准则让我受益终身。”他也将自己获得的成就更大程度上归功于信息学院这些年来的发展，学院老师的帮助，以及孟小峰教授所提供的广阔发展平台。他说，孟小峰教授参加国际会议带来的国际前沿视角，以及他与一些研究机构企业的合作，都为实验室的同学提供了更多机会。也正是在这样的思想交流中，王仲远慢慢找到了自己的喜欢的研究课题，并且在这种兴趣的驱动下，一步步走向成功。

“虽然作为一名人大的理工科学生，我时常觉得自己像是人大的‘少数派’，但我依然感到无比的自豪，我的成长史是伴随着信息学院这些年来快速发展的历程的。在这个过程中，学院的领导、党委老师以及我的导师所给予我的培养与帮助，都让我永远铭记于心。毕业后，以往所有的这些荣誉都将成为历史，我又将站在一个新的起跑线上，与其他人一起竞争。我将永远贴着人大 IT 人的标签，通过对自己的不断提升，来奉献社会，我也一定会尽自己所能，来帮助人大信息学院不断地扩大影响力。”

**《信息月刊》采访稿：**

12 月 4 日，本刊的主编和几个干事有幸采访到了王仲远师兄。王仲远师兄是信息学院本科学生的榜样，虽然现在还是研三的学生，但却已经有丰富的科研和项目经历，并获得了我校学生的最高荣誉——吴玉章奖学金。他曾参加与 IBM 中国研究院、联想研究院、诺基亚西门子研究院合作的多个项目，并已被微软亚洲研究院录用。他在孟小峰教授的网络与移动数据管理实验室中工作时，曾在国际及国内会议上发表多篇论文，在《中国计算机报》上组织发表专题文章，并开发过数个有影响力的创新性系统。在采访中，他热情地回答了我们的问题，主动地分享了他的成功经验，还强调希望他的经验能对新生有所帮助。师兄对问题深刻的理解、缜密思维的逻辑和侃侃而谈的风度让我们印象深刻。

## 关于学习生活

**问：好像大学生活和想象中有很大不同。以前高中老师一直都是说上了大学会很轻松，会有很多很多空闲的时间，现在发现完全不是这样。**

答：我以前上高中的时候，也听说过这样的论调，不过高中生们大概都被老师欺骗了。可能是为了让学生们能够更加集中精力地面对高考，老师总是渲染大学生活的轻松惬意。但是在一所优秀的大学里想得到更好的发展，确实需要付出很大努力。课程不能落下，作业也需要认真完成。你当然可以选择整日悠闲地度过，可以荒废时间，但如果你想有收获，你就必须努力，用勤奋弥补先天的差距。

**问：我们学院被称为“四大疯人院”之一，在人大这样一所以文科为主的学校里，我们有什么优势？**

答：我认为我们能成为“四大疯人院”之一也是一种荣幸。所有的付出必然都会有回报。我们的确很累，而且理工科在人大不是主流，但事实上，我们就业的情况却排在全校前列。可能看到其他学院的悠闲生活使我们难免心浮气躁，但只要我们能静下心来，就会发现自己有很大收获。

**问：您曾担任班长，又曾担任党支部书记，请问如何更高效的分配时间而不引起学习与学生活动之间的冲突？**

答：每个人每天都只有二十四个小时，无论你面对多少事情，也只能利用这有限的时间去处理它们。这二十四个小时中，除去做琐碎事情的时间，剩下的时间的确十分有限，很显然，这时候你必须做出选择。我想，我今天能取得这样的成绩，更多的是凭借对这些时间的有效利用。勤奋的确能够弥补很多方面的不足。比如说，有些人在面对诸多事情的时候有所选择，专心做好其中一件，而另一些人事事都想沾边，但每一种都浅尝辄止，就不大可能取得成功。而我就采取折中的办法，选择一些自认为比较重要的，同时压缩娱乐以及一部分睡眠的时间，才能够完成这些事情。这些年来，我放弃了各种假期和周末，这样才有可能比别人多做一些事情。

上天是公平的，它给予每个人相同的时间，而这些时间就由我们个人去调配。当面临诸多选择的时候，我通常采取的方式是按照事情重要程度排序，然后从上到下，尽自己所能的去做。其实，这是很多人都能想到的办法，但确确实实也是最有效的办法。所以必须要放弃掉一些事情，想把每件事都做好，这也是不可能的。

**问：大学学习与高中有很大不同，作为新生，我们对大学的学习方法、习惯及技巧不是很了解，请问师兄，我们如何能平稳的过渡到大学学习中？**

答：第一个学期，大家对这方面的问题会特别的关注，比如不知道大学的考试到底是什么样子，要怎么去准备，与高中的区别到底有哪些。有一句话非常有道理：如果能拿出高中 80%的劲头来学习大学的课程，那就很容易达到前几名的水平。其实，问题的实质并不是大家的学习方法不好，而是大学中会有很多的社团、讲座以及各种各样的活动，我们有时不能在这种环境中静下心来学习。就人大的氛围而言，平时许多文科生都能够将大量时间投入到学生活动中去，而我们只能在自习室中钻研，尤其是临近期末，通宵自习室中总能看到信息人的身影。其实能够考来这里，我们都会有自己的一套学习方法，这种方法在大学中是完全可行的。但时间是很公平的，投入了多少时间去学习，就会有多少收获。另外，本科学习阶段更多注重的是对知识点的掌握，所以完全没有必要刻意钻研偏题怪题。

其实，大学更注重培养的是学习的能力，课本知识尤其是计算机的相关知识更新换代是很快的，我们现在学习的课程可能几年内就会发生重大变化，所以我们应该通过课程的学习培养一种自学能力，培养一种在短期之内掌握一门新课程的能力，这个或许可以归结为大学与高中的不同之处，这种区别是要大家花时间好好体会的。

同时，大学应该培养的是“做人、做事、做学问”的能力。所谓“做人”，绝不是说做一个老好人或是阿谀奉承的人，而是一个有道德的人，“国民表率”就是指这个要求，只有有了这种最基本的准则，我们走向社会才能成为有用之才；而“做事”，就是要做负责任的事，比如班级的事情或者社团的事情，只有真正地为之付出才会有收获，才配得上“社会栋梁”的称呼；至于“做学问”，就是要在学习上有所追求。如果能做到这些，那就不仅是合格的学生，而是优秀的学生了。

**问：大学的班级和高中有很大不同，很多人都感觉大学的同学关系不像高中那样亲密了，您曾在本科期间担任能班长，您能说说您促进班级团结的经验么？**

答：这也是我作为辅导员期间特别关注的问题。由于不在同一个教室上课，同学们之间的交流较高中来说少了很多，即使是同宿舍的同学，可能也只是在每晚睡觉之前聊聊天。但是大学是四年，如果把握的好，这四年的大学生活会成为你人生中最难忘的一段时光。

树立班级集体观念必须从刚入学之后就开始关注，这个问题的领导者就是各班的辅导员，而解决问题的唯一途径就是建立一个强有力的班委。一个好的班委能使班级具有很强的凝聚力和优良的班风及学风，这样，每个人的大学收获就会很多。比方说以前辅导的班级，除了大家经常见到的组

织出游、聚餐以外，我们更注重小的方面，比如过生日的同学会收到班级贺卡，这虽然是很小的事情，但能使同学们感到班级的存在，感觉到周围人的在意和关心，通过这些，就能使班级充满爱。再有，让我印象深刻的是期末复习，同学之间互相交流学习笔记，大家共同复习讨论。我想，这些都是班级观念形成的重要步骤和条件。

同时，我也想告诉每一位师弟师妹，大家到大学以后个人的事情必然会非常多，但我还是建议每一次的班级活动都应该尽量参加。如果大学四年有意逃避这些活动，那么你的大学生活就是失败的。因为你即使个人发展的再好，没有了同学没有了班级没有了集体观念的大学生是不完整的。大学不仅是个学习的地方，也是大家要学会慢慢融入社会的场所，而班级就是这样一个载体。等大家毕业以后，同班同学的观念就会愈发强烈，当我们工作之后，走到某一个城市，如果遇到同班同学，一定有非常亲切与开心的感觉。

**问：您认为和其他理工科学校相比我们有什么优势？**

答：我们有一个很大的优势，就是许多公司人力资源部的工作人员都是人大的校友，看到校友还是会很亲切的。因为人大有劳动人事学院，毕业生中有很多人从事相关工作。而且人大的人文环境很好，有利于我们全面发展，成为“具有技术背景的复合型人才”。找工作时可能会觉得周围有很多外校的学生而本校学生较少，但这只是因为我们学院本身规模小，影响力可能也不够大。事实上，我们的就业情况是很好的。已毕业的师兄师姐也为信息学院建立了良好的口碑。因此，我们有很好的机会。但能否成功，最终还是取决于自己。

## 关于科研

**问：我们都知道进入实验室的机会是很稀少的，那么一个大学生应该具备什么样的素质才能进入实验室呢？**

答：其实就我所知，至少计算机系的老师们是越来越愿意让一些本科的同学参与到实验室项目的研制中来，从学校的范围来讲，也给大家提供了很多机会，比如“挑战杯”、“创新杯”以及一些国家创新实验平台，通过这些，鼓励大家投身到实际的科研的项目中去，这样不仅对实验室有所帮助，同时也是对大家能力的提高。

实际上，现在本科生想进入实验室并没有非常大的障碍，更多取决于大家的兴趣。如果有意愿，只要主动去联系学院的老师，他们都会非常欢迎。当然，从技术上说，大家还是要掌握一定的基础才适合进入实验室。在技术和心理上都准备好之后，就可以联系相关老师进行申请了。其实，学院的网站上也能了解到各个老师所研究的课题，不妨可以通过这种方式进行了解。还有，任何的疑惑都可以让每个宿舍配备的导师帮助解答。另外，大学与高中很大的区别是大学的信息是要靠大家去主动获取的，而没有人会保证你了解到需要的信息，比如我们可以在学院或校外的网站以及师兄师姐那里了解相关计划、比赛等等。


**问：我们信息学院在科研方面的实力如何？您目前在从事什么研究？**

答：咱们整个信息学院在研究生阶段对于科研是比其他学校要重视得多。可能像北邮或者是其他的一些学校，他们研究生可能更多接的是工程性的项目，所以可能很多的研究生实际上是给老板打工的。而咱们学院对于科研这一块还是比较重视的，我进入研究生学习之后，在实验室里一直都在做与科研有关的东西。另外，咱们这边的科研氛围也非常好，每一年都有许多知名的国外教授，尤其是一

些国外著名华人教授，他们会到我们学院来做很多讲座，从这些讲座我们就可以了解到很多我们这个领域的非常前沿的知识。

大家也知道人大信息学院在数据库方面还是非常有优势的，在这方面经常会有各种各样非常牛的专家过来，他们给我们做讲座，我们就可以了解到这个领域内最前沿的一些研究课题，跟着去追踪这些最前沿的课题。

此外，近些年越来越多的研究生有机会能够到香港、美国、欧洲各个国家去参加各种学术会议或者做短期的学术访问，与国外的研究机构做合作与高端的研究。学校和学院都在给大家提供非常好的机会，至于每个人能够达到怎样的高度，那就要取决于个人了。我这些年来，在我们导师的带领下，主要做两个方面的研究题目，一个是 Web 数据集成，就是把网络上分散的数据有效的组织和管理，方便用户访问。另一个是“云计算平台上的数据管理问题研究”，这个概念其实现在被炒作的有些过头了，但是它的一个核心的思想就是这种分布式地对海量数据进行处理和管理的一个理念，这是它最核心的一个东西。这些东西今后大家都会慢慢接触到。

**问：师兄刚才提到了学习与科研。您能否具体解释一下这二者的关系？它们能否互相融合，相辅相成，共同促进个人素质的提高？**

答：大家都知道，计算机不是一门古老的学科，它是从应用中发展出来并由人加以总结而形成的一门年轻的科学。总的来说，信息科学发展的时间非常短暂。有关的教材，实际上就是人们把实践中总结出的经验加以整合形成的特定体系。因此，教材中有错误和未解决的问题是难免的。严格来说，发现并改正这些错误或是解决未解决的问题都可以算做科研和创新。因此，科研和学习并没有本质的区别。比如我知道的大三的一些同学，在学习过程中常常会有自己的想法，这些想法可能就是一些科研性问题。如果把这些想法深入研究下去，实际上也就是科研和创新。

从科研返回到学习上来，要知道所有的科研其实都离不开最基本的知识，没有非常扎实的基础就无法做科研上的创新。其实，许多计算机科学问题归根结底都是数学问题。而对于大一的数学学习，一定要扎扎实实。学数学分析、做吉米多维奇是很痛苦的，但这些东西在将来都会一步一步影响到研究上来，所以学习和科研是非常有关联的。

## 关于职业生涯

**问：师兄，您当时成绩非常好，有没有考虑申请外校保研呢？**

答：这就要说起我的导师孟小峰教授了。我非常感谢我的导师孟小峰教授，从本科开始，我就进入他的实验室。在导师的指导下，我接触了我们这个领域最前沿的研究课题。同时，孟老师提供给我很多的研究机会，使我能够实际的参与到一些项目的开发当中，大大提高了我的综合能力。每一个人都要学会感恩，我也相信自己在人大能得到更好的发展，于是就放弃了当年外保的一些机会。

其实人大和信息学院为大家提供的平台已经足够供大家发挥了，当你失败的时候，你没有必要去责怪机会的不足，而应去反省自己有没有努力。大家在做一些事情的时候或许感到很痛苦，但是只要坚持下来，就会发现收获是会大于付出的。

**问：您认为作为信息学院的学生，我们所学的专业有利于就业吗？选择什么职业比较有利？**

答：这也是个人职业规划的问题了。以计算机专业为例，如果你对编程非常感兴趣，我们学院的毕业生完全有能力能够进入任何一家著名的 IT 公司，当然，我们不一定就要去从事编程等职业。进入证券、保险、银行等行业的相关部门

也可以有很好的发展——他们需要懂技术的人，但你不一定直接从事开发等工作。

还有就是咱们院的女生可能也会比较关注这个问题，但是这也是涉及到自己职业生涯的规划，女生们编程和男生没什么不同，不像有些人担心的那样，说不能从事编程或只能做测试员，关键还是看个人意愿，她们既可以进入 IT 公司从事核心业务的开发，也可以进入政府机关部门、进出口贸易公司、银行、保险、证券等金融行业公司，就业范围还是非常广泛的。

**问：前几天看到一份数据，“失业率最高的十大行业”，计算机排在首位。对此您怎么看？**

答：可能这个数据并不能反映实际情况。IT 行业是一个流动性极强的行业，很多人会频繁“跳槽”以求更好的发展，可能这部分人也被统计在“失业人群”之内。还有一种情况，就是一些中小型企业运行若干年后会倒闭，这时企业中的人员也会被视为“失业人员”，但他们用他们的经验很容易找到新工作。在我看来，IT 行业虽免不了失业但总体来说就业形势还是很好的。

**问：有一种说法是“编程是吃青春饭的。”对此您有什么想法？**

答：我不认为这种观点是正确的。我认为编程是一种实践性工作，多年的工作经验会让人积累经验，这些经验是任何书本也无法教给你的。我不认为年纪大了的人就不再适合做编程工作；相反，多年的经验会让他们更加出色。

**问：您能谈谈您应聘微软亚洲研究院的细节吗？**

答：到了今年我找工作的时候，我也是有选择性地对一些公司进行挑选，主要是互联网行业的，比如说百度、腾讯、创新工场、IBM、EMC、Sogou 等公司，当然在谈到这些公司的时候自然就不能忽略整个世界范围内最好的一些公司，包括 Google 和微软。 关于 Google，可能他们公司今年正在进行战略调整，几乎不从公开的校园招聘渠道招人，还有微软，今年招的人也不是特别多。但是微软和 Google 是所有学计算机，或者说跟信息专业相关的同学最想进去的两家公司。那么我非常幸运能得到去微软亚洲研究院面试的机会，接着是七轮的面试。他们那边的面试很有特色，但或许是国际化的做法，就是根据你的面试表现，一天就有可能安排你参加多达六轮的面试，包括吃午饭的时候都在面试。当然面试的内容非常多样，也包括一些编程题、数学题、逻辑推理题、开放式问题，还有包括对你研究上的一些成果的考察，总的来说，通过这么多轮的面试，他们对于一名候选者的考察还是比较综合的。在这种面试形式下，是没有办法通过临时抱佛脚的方式侥幸通过面试的，还必须是靠平时一点一滴的积累，才能够在这样的面试中厚积薄发。

微软亚洲研究院是在国内建成但是是世界一流的计算机研究机构，为推动信息技术与交互计算的进步做出了卓越的贡献。我能拿到他们的 offer，感到非常的幸运，自然也就放弃了其他的一些 offer，最后就选定他们了。

**问：您对计算机专业的兴趣是怎样形成的？**

答：我高考时第一志愿也不是计算机专业，对这一学科的热爱也是上了大学后逐渐形成的。可能平时你会觉得编程颇为枯燥，但当你真正接触到一些实际系统并投入其中时，你会发现程序的美妙之处。在一个程序中，尤其是被广泛应用的程序，可能你做的每一件小事都会产生难以形容的巨大影响，甚至影响到数千万人的生活方式——想到这些，你又怎能不激动，怎能不热爱编程这一项工作呢？

**问：师兄如果将来您有机会会不会考虑向出国方向发展或者往香港、上海等地发展？**

答：对，你这个问题很好，其实我觉得关于出国这个事情，到大二大三的时候同学们应该就会考虑的比较多了。因为到了那个时候会遇到一个坎，类似于选择人生道路的一个分化点。我们不能说哪一条路是最好的，每个人会有每个人不同的选择。

我当时想的是，出国要有很明确的认识，大家现在可能很多人选择出国是随大流，或者说是从众心理。你出去读个博士的目的是为了做什么，是为了做研究吗？那么就应该看你对做研究这方面感不感兴趣。读完博士之后很大的可能就是在某个研究机构做研究，或者回到国内到某个高校去当老师。其实这是出国读博最有可能的结果。我当时觉得自己在心理上并没有这方面的准备，而且出国读博的时间上还有些久。当然，可能有些人经过折算觉得出国的时间是值得的，或者说出国的性

价比是很高的。很多人也会有这方面的考虑。我认为，无论是做什么事情，肯定是需要和外面进行交流的，但交流有很多的方式。比如今后我进入微软亚洲研究院肯定会有很多的机会出去，到国外的各个研究机构或者是一些学术机构做一些交流访问。我觉得通过这些已经足够让我达到我去了解国外一些研究动向和研究成果的目的。我们国家现在也已经非常开放了，如果大家关注学院网站的话，就会看到咱们每个月都会有一些非常牛的人的讲座。另外学校也在建立国际小学期，这些其实都是为了扩宽大家的视野，去了解最先进的一些东西。这些都是大家与外界交流的一些方式。

此外我自己并不认同，我也并不完全喜欢那种到国外定居或先到国外留学再想办法留在国外的生活，我认为并不完全适合我。我觉得即使他们那边初始的福利条件或者是生活条件看起来比国内要好，但是我觉得从长远的发展和自己能否融入这种社会来讲，以及自己这种心理定位上来讲，我还是更愿意在国内发展，为祖国的繁荣昌盛尽自己的一份力量。这就是我当时为什么没有选择出国的原因，而选择留在国内。那么留在国内，现在我觉得能够进入了这样一个国内领先、世界一流的研究机构，我已经达到了相同的目的。出国不出国只是一种形式而已。

大家可能不知道，我们院的计算机系本科专业是 1999 年才成立的，整个信息学院以及计算机系的发展非常迅速。现在学院的领导和老师也在不断地给大家提供交流的平台和空间，以及可以选择的道路。比如咱们现在有交换生制度，本科生到了大二大三可以选择到香港去交换，到了保研的时候有和纽约州立大学合办的“3+2”双学士学位保送的制度，然后还有直接免试直接去新加坡国立大学学习的制度等等。我觉得学院这边已经给大家提供了这么多的机会，我想只要大家抓住一两个就很不错的。学院对大家外出交流访问的资助力度也是在逐渐加强的，比如说现在学院的老师的项目越来越多，研究生越来越有机会到各个研究机构，到各个国家去参加学术会议或参加短期的学术的访问，在那边和国外的一些研究机构来做合作做一些高端的研究。我觉得这些机会学校、学院在不断努力地给大家提供，我们已经有一个非常好的平台了，至于每个人能够达到怎样的一个高度就完全取决于个人了。

**问：请您略谈一下 IT 行业的发展前景？**

答：我认为 IT 行业是一个蓬勃发展的行业，其中有很多很多机会，前景应该是比较光明的。我认为作为人大信息学院的学生，我们应该找好自己的定位，规划好自己的职业生涯。这样，凭借“人大”的品牌和自己的努力，我们一定会有很好的发展。

**问：您即将离开信息学院，有什么想对师弟师妹们说的。**

答：其实我非常喜欢跟本科同学尤其是新生一起交流，随着自己的成长，或许在思想上跟本科同学的代沟会越来越深，但是我总觉得通过这些交流，一方面可以把我的经验传递给大家，另一方面也能从大家那里得到很多新鲜的东西。这也就是为什么我在进入研究生学习的第一年就申请做辅导员。其实我觉得时间过得很快，自己入学第一天的情形还时常浮现在脑海中。但是一晃现在已经在人大待了六年多，今年是第七个年头了。刚才说的都是我自己的一些经验，没有什么保留，因为我觉得这些东西如果能跟大家分享，能让后面的人受益的话，对我自己也是一种提高。

我即将离开人大，离开信息学院，说实话，在这么多年的学习生活中，我已经和人民大学、信息学院结下了深深的感情，我希望人民大学越办越好，信息学院也越办越好。我希望每个信息学院的学生都能树立自豪的观念，为自己是人大 IT 人而自豪。信息学院未来的辉煌要靠我们所有人的共同努力；我也坚信，在学院领导、老师的带领下，在大家的共同努力下，信息学院规模会越来越大，发展会越来越好。

# 实验室研讨会

| | |
|---|---|
| **2009.12.26 Venue: FL1, Meeting Room, Information Building** | |
| Rui Zhang | **Continuous Intersection Join on Moving Object**<br>Abstract:<br>The continuous intersection join query is computationally expensive yet important for various applications on moving objects. No previous study has specifically addressed this query type. We can adopt a naive algorithm or extend an existing technique (TP-Join) to process the query. However, they compute the answer for either too long or too short a time interval, which results in either a very large computation cost per object update or too frequent answer updates, respectively. This motivates us to optimize the query processing in the time dimension. In this study, we achieve this optimization by introducing the new concept of time-constrained (TC) processing. |
| Jinchuan Chen | **Data Integration with Uncertainty**<br>Abstract:<br>A survey of data integration with uncertainty. This report introduced existing methods of data management with uncertainty. |
| **2009.12.19 Venue: FL1, Meeting Room, Information Building** | |
| Haiping Wang (Cloud Computing Group) | **cassandra and sigmod contest**<br>Abstract:<br>Cassandra is a highly scalable second-generation distributed database, bringing together Dynamo's fully distributed design and Bigtable's ColumnFamily-based data model.The task of sigmod programing contest 2010 is to implenment a simple distributed query executor built on top of the last year's main-memory index. |
| Ying Lu (Mobile Group) | **Hammer & Nail**<br>Abstract:<br>"Research is actually a process of hammers(methods) hammer nails(problem)". This report first presents three hammers, i.e.three kinds of hash functions, which are signature, OPMPHF(Order Preserving Minimal Perfect Hash Function) and LSH(Location Sensitive Hashing).Then it introduces a nail using the hammers above.It is called Reveser k Spatial and Textual Nearest Neighbor(RkSTNN). |
| **2009.12.12 Venue: FL1, Meeting Room, Information Building** | |
| Yingjie Shi (Web Group) | **Survey on Data Management in the Cloud**<br>Abstract: |

| | |
|---|---|
| | With the development of computer and communication technology, a large scale of data are produced. Cloud-based database is one solution to efficiently store and analyze these data. In this talk, we present some cloud-based database and summarize them from different aspects. |
| Bingbing Liu (Cloud Computing Group) | **Hive – A Warehousing Solution Over a MapReduce Framework**<br>Abstract:<br>Introduce a system which support managing and querying structured data and builded on the top of hadoop and the query language. |
| **2009.12.05 Venue: FL1, Meeting Room, Information Building** | |
| Ruxia Ma (Web Group) | **Trust Metric on Social Network**<br>Abstract:<br>This report introduces five trust metric mechanisms on social network, such as: Advogato, Appleseed and TidalTrust, etc. We mainly describe the main ideas of those algorithms and their realization. |
| Wei Chen (Web Group) | **Data Fusion-Resolve Data Conflicts in Integration**<br>Abstract:<br>In this talk we gave a brief introdution to data fusion, including data conflict types, conflict resolution strategies, the role played by data fusion in integration programs and current approaches to data fusion. Then we addressed some challenges and open problems in data fusion research. Finally we presented a brief summary to this talk. |
| **2009.11.28 Venue: FL1, Meeting Room, Information Building** | |
| Xian Tang (mobile Group) | **ACR: an Adaptive Cost-Aware Buffer Replacement Algorithm for Flash Storage Devices**<br>Abstract:<br>In this talk, we propose an adaptive cost-aware buffer replacement algorithm--ACR, which adapt to various access patterns on flash disks. |
| Yulei Fan (Mobile Group) | **Multi-version Concurrency Control of Database Based on Flash Memory**<br>Abstract:<br>Data may have multiple versions as because of the feature of not-in-place update and in-page logging store mechnism in flash memory. Multi-version concurrency control has to be implented based on the Serialization theory, and it includes MV2PL(multi-version 2PL), MVTO(multi-version TO), MVSGT(multi-version SGT), TW(time warp) and ROMV(read-only multi-version). We evaluated the performance of these algorithms by implementing experiments on existing DBMS such as MS SQLServer, MySQL and Postgres. Finally, |

| | |
|---|---|
| | we proposed some future work in Multiple-version Concurrency Control. |

| **2009.11.21 Venue: FL1, Meeting Room, Information Building** | |
|---|---|
| Junjing Xu (XML Group) | **Efficient String Similarity Search Using Synonyms**<br>Abstract:<br>This report introduces the gram_based string matching functions and the new similarity funcion. |
| Lizhen Fu (XML Group) | **Reachability Queries on Large Directed Acyclic Graphs**<br>**Abstract:**<br>In particular, graph reachability has attracted a lot of research attention as reachability queries are not only common on graph databases, but they also serve as fundamental operations for many other graph queries. In this reprot, I introduce my new graph label to speed up the processing of reachablity queries on DAG， which index is small and which can be constructed easily |
| Jinzeng Zhang(XML Group) | **Information Retrieval Model and Relevance Feedback**<br>**Abstract:**<br>This report first introduces four classic information retrieval models. Based on those models, we present two methods of improving retrieval results |

| **2009.11.14 Venue: FL1, Meeting Room, Information Building** | |
|---|---|
| Yukun Li(Web Group) | **Review our studies on dataspace**<br>**Abstract:**<br>Reviewed our works on dataspace research, and introduced a work we are doing. |
| Xiangyu Zhang(Web Group) | **Dataspace Research Report**<br>Abstract:<br>Introduced research and system implementation progress on Dataspace research. |
| Yubo Kou(Web Group) | **Leveraging Feature Context to Facilitate Sub-graph Query in Graph Database**<br>Abstract:<br>Previous techniques focus on feature selection strategy to filter false graphs as more as possible. This approach has met a bottleneck, that as the feature is becoming more and more complicated, precision is still low. Thus we propose to investigate into how feature context could help improve pruning power in sub-graph query. |

| 2009.11.08 Venue: FL1, Meeting Room, Information Building | |
|---|---|
| Yukun Li(Web Group) | **About CIKM2009 Story**<br>Abstract:<br>Give a short summary on CIKM 2009 based on my impression on this conference, especially introduced the three keynotes. |
| Da Zhou(Mobile Group) | **Review of CIKM 2009**<br>Abstract:<br>CIKM is a high level international conference. There are three tracks |
| Zhongyuan Wang(Web Group) | **Summary of CIKM2009**<br>Abstract:<br>In this talk, I presented three papers and one tutorial related to Web data management and click log mining in CIKM2009. Then give some summary of CIKM2009. |
| Xiangyu Zhang(Web Group) | **IR is Interesting-CIKM 2009 Report**<br>Abstract:<br>In this presentation, I gave a brief summary and introduction to the CIKM 2009 conference and some of my own experience on this conference. |
| **2009.10.31 Venue: FL1, Meeting Room, Information Building** | |
| Xiangyu Zhang(Web Group) | **An Efficient Multi-Dimensional Index for Cloud Data Management**<br>Abstract:<br>In this presentation, I introduced our work of multi-dimensional index structure for Cloud Computing platforms. |
| Yukun Li (Web Group) | **Supporting Context-based Query in Personal DataSpace**<br>Abstract:<br>Many users need to refer to content in existing files (pictures,tables, emails, web pages and etc.) when they write documents(programs, presentations, proposals and etc.), and often need to revisit these referenced files for review, revision or reconfirmation. In this paper, we propose an efficient solution for this problem. We firstly define a new personal data relationship |
| Da Zhou(Mobile Group) | **Pre-Report for CIKM 2009**<br>Abstract:<br>Solid State Drive (SSD), emerging as new data storage media with high random read speed, has been widely used in laptops, desktops, and data servers to replace hard disk during the past few years. However, poor random write performance becomes the bottle neck in practice. In this |

| | |
|---|---|
| | paper, we propose to insert unmodified data into random write sequence in order to convert random writes into sequential writes, and thus data sequence can be flushed at the speed of sequential write. |

## 2009.10.24 Venue: FL1, Meeting Room, Information Building

| | |
|---|---|
| Xiangye Xiao (Web&Mobile Group) | **Overview of Talks in NDBC 2009**<br>Abstract:<br>Dr. Xiangye Xiao gave a brief review of invited talks in NDBC 2009 which includes Dr. Xin Dong from AT&T, Prof. Weiyi Meng from Binghamton Univ., Haixun Wang from MSRA and Lei Chen from HKUST. |
| Yukun Li(Web Group) | **Report on SKG2009**<br>Abstract:<br>Give an introduction on SKG2009, and focusing on introducing the two keynotes of this conference. |
| Zheng Huo(Mobile Group) | **A new topic: queries with geo-information**<br>Abstract:<br>Discovering users' specific and implicit geographic intention in web search can greatly help satisfy users' information needs. Research on queries with geo-information has becoming hot these years. There are several methods. First, the training data based methods, these methods need big data of query logs; another is spatial and texual information retrieval methods, but these methods can only deal with local geo-informaiton. The challege is how to discover users' implicit geo-information in queries. |
| Xiangmei Hu(Web Group) | **Trajectory pattern mining**<br>Abstract:<br>The pervasiveness of mobile devices and location based services is leading to an incresing volume of mobility data.This side effect provides the opportunity to analyse the behaviors of movements.With this background,trajectory pattern mining has been a popular topic.This report mainly introduces some representative work about this topic and points out some defects. |

## 2009.10.11 Venue: FL1, Meeting Room, Information Building

| | |
|---|---|
| Jing Ai(Web Group) | **C-Rank -- A Credibility Evaluation Method for Deep Web Records**<br>Abstract:<br>How to identify and evaluate information credibility ranking has become an increasing important problem. To address the issue, an effective credibility evaluation method called C-Rank to compute trust |

| | |
|---|---|
| | values of records in Deep Web databases is proposed, which constructs an S-R Credibility Graph for each record. |
| | |
| Xing Hao(Mobile Group) | **Privacy Preserving towards Continuous Query in Location-based Services**<br>Abstract:<br>With advances in wireless communication and mobile positioning technologies, location-based mobile services have been gaining increasingly popularity in recent years. Privacy preservation, including location privacy and query privacy, has recently received considerable attention for location-based mobile services. A lot of location cloaking approaches have been proposed for protecting the location privacy of mobile users. However, they mostly focus on anonymizing snapshot queries based on proximity of locations at query issued time. Therefore, most of them are ill-suited for continuous queries. In view of the privacy disclosure (including location and query privacy) and poor quality of service under continuous query anonymization, a δp-privacy model and a δq-distortion model is proposed to balance the tradeoff between privacy preserving and quality of service. Meanwhile a temporal distortion model is proposed to measure location information loss during a time interval, and it is mapped to a temporal similar distance between two queries. Finally, a greedy cloaking algorithm (GCA) is proposed, which is applicable for both anonymizing snapshot queries and continuous queries. Average cloaking success rate, cloaking time, processing time and anonymization cost for successful requests is evaluated with increasing privacy level (k). Experimental results validate the efficiency and effectiveness of the proposed algorithm. |
| | |
| Wei Wang(XML Group) | **Algebra-based Transform query optimization strategy**<br>**Abstract:**<br>XQuery/Update defines a special Transform query, which is similar to be hypothetical query in relation databases, and can be expressed as“Q when {U}”. In other words, the results of query Q are the same as the results after executing hypothetical update {U} on the original database, without actually updating database. The Transform queries need to copy the nodes in XML database and then update copied nodes, so it doesn’t affects the database. But Transform queries will usually copy and update a lot of nodes which are useless for query Q and result in high cost. It is critical for query optimization to decrease the number of copied nodes and the update operation. In this paper, we propose a set of rules for Transform query optimization techniques based on OrientXA. Which are implemented in OrientX3.0. |
| | |
| Da Zhou(Mobile | **HF-Tree--An Update-Efficient Index for Flash Memory** |

| | |
|---|---|
| Group) | Abstract:<br>Due to the expensive write cost of flash memory, traditional disk-based indexes have a poor update performance when directly applied to flash drives. In this talk, Da Zhou proposed a novel index called HF tree to improve the update performance of Flash memory, which integrates BF -tree with Tri-hash. |
| | |
| Zhichao Liang(Mobile Group) | **Sub-Join--A Query Optimization Algorithm for Flash-based Database**<br>Abstract:<br>Compared with Hard Drive Disk (HDD), SSD has a lot of advantages, such as high random read performance, low power consumption and lightweight form. Therefore it is envisioned to be next generation data storage instead of HDD. However, the enhancement of query performance for flash-based database is not the same as the IO ratio of SSD to HDD. The reason is existing databases which are designed for HDD can not take full advantage of high IO performance of SSD. In this paper, a new join algorithm, Sub-Join, is proposed. Sub-Join first projects the column of join and primary key as Sub-Table, and then executes join operations on Sub-Tables. Finally results are gotten from original table according to the result of join on Sub-Tables. The compared experiments with Oracle Berkeley DB show Sub-Join outperforms original indexed nested-loop join at the ratio of about 40%~100%. The result strongly shows the high efficiency of this method. |
| | |
| **2009.09.28 Venue: FL1, Meeting Room, Information Building** | |
| | |
| Xin(Luna) Dong(AT&T Research) | **Data Integration with Uncertainty**<br>Abstract:<br>Dr. Xin (Luna) Dong from Data Management Department at AT&T Research visited Web And Mobile Data Management (WAMDM) lab and gave an invited talk about Data Integration with Uncertainty. Her talk mainly focused on some important and valuable topics in uncertain data integration. |
| | |
| Xiangye Xiao(Web&Mobie Group) | **Efficient Co-Location Pattern Discovery**<br>Abstract:<br>Dr. Xiangye Xiao gave a brief talk about her research topics when she was a PHD candidate in the Hong Kong University of Science and Technology. Her talk included efficient co-location pattern discovery and Web browsing on mobile devices. Besides, Dr. Xiangye Xiao proposed some ideas about future research. |

| | |
|---|---|
| Jiaheng Lu(XML Group) | **Keyword Search Techniques in Mobile Web**<br>Abstract:<br>Dr. Jiaheng Lu received an a funding award about "keyword search in mobile web" from National Science Foundation China (NSFC). He gave a detailed demonstration about the project and proposed some possible topics. |
| **2009.07.25 Venue: FL1, Meeting Room, Information Building** | |
| Qingsong Guo(XML Group) | **OrientX4.0 - Supproting Keyword Search**<br>Abstract:<br>With the developing of xml technology, more and more pepole using xml data. In traditional, we use the standard query lanaguage XQuery to find the data we need, but we need to learn the "XQuery" and we must know the structure and content of the xml document. It is great challenge of naive users. For this popose, in the new edition-OrientX4.0, we supporting the xml keyword-search , which can solve the problem we meet by using XQuery and make pepole using xml more easier. |
| Wei Wang(XML Group) | **OrientX4.0 System Development Report**<br>Abstract:<br>the implement of XML keyword search |
| **2009.07.18 Venue: FL1, Meeting Room, Information Building** | |
| Xing Hao(Mobile Group) | **Probabilistic kNN Query in Road Network**<br>Abstract:<br>Queries for moving objects in road network, especially kNN(k Nearest Neighbor) queries are very important and have received considerable attention. This speech discusses how to model the uncertainty data and process kNN queries in road network. |
| Yi Huang(Mobile Group) | **Report on Privacy Protection Demo Appplication Development**<br>Abstract:<br>In order to apply the current privacy protection algorithms and integrate them in the 863 Pervasive Computing project, we decided to develop a demonstration application.This report introduced the technical and functional characteristics of the application as well as the development plan. |
| Chunjie Zhou(Mobile Group) | **Query Processing over Interval-based Out-of-order Event Streams**<br>Abstract:<br>Complex event processing has become increasingly important in modern applications, ranging from supply chain management for RFID |

| | |
|---|---|
| | tracking to real-time intrusion detection. A key aspect of complex event processing is to extract patterns from event streams to make informed decisions in real-time. However, network latencies and machine failures may cause events to arrive out-of-order at the event processing engine. In addition, existing temporal pattern mining assumes that events do not have any duration. However, events in many real world applications have durations, and the relationships among these events are often complex. In this work, we propose solution to process both sequence and parallel pattern queries on out-of-order event streams. First, we analyze the preliminaries and the problems caused by out-of-order data arrival. We then propose a method to detect out-of-order event patterns. A new solution including time-interval to solve out-of-order problems is also introduced. Lastly, we conduct an experimental study demonstrating the effectiveness of our approach. |
| **2009.07.11 Venue: FL1, Meeting Room, Information Building** | |
| Zhichao Liang(Mobile Group) | **System Development Report of Flash Group**<br>Abstract:<br>Our target is to develop a special flash-based DBMS,and we decide to do some changes on an existing open source DBMS to work it out. However, as a matter of fact,there are lots of open source systems. Which one is the best choice? After a detailed analysis, we believe MySQL,which contains the Berkeley DB as one of its storage engines,is the answer to our problem. |
| **2009.07.04 Venue: FL1, Meeting Room, Information Building** | |
| Yukun Li (Web Group) | **SIGMOD2009 Overview**<br>Abstract:<br>Analyze the current hot research issues based on the accessed papers of SIGMOD2009, and introduce two papers of this conference. |
| Da Zhou (Mobile Group) | **Flash Research Report**<br>Abstract:<br>Flash-based database systems research becomes more and more hot. In sigmod2009 and VLDB2009, we are glad to see that there are some papers about the indexing, query processing and transaction processing. This report gives a coarse overview to the motivations and ideas of these papers. |
| Lizhen Fu (XML Group) | **XML Labeling and Query Optimization in Sigmod09**<br>Abstract:<br>Optimization of complex XQueries combining many XPath steps and joins is currently hindered by the absence of good cardinality |

| | |
|---|---|
| | estimation and cost models for XQuery.Labeling schemes lie at the core of query processing for many XML database management systems. Designing labeling schemes for dynamic XML documents is an important problem that has received a lot of research attention. This presention introduce a new labeling scheme DDE and a new Runtime Optimization approach ROX in sigmod09. |
| **2009.06.27 Venue: FL1, Meeting Room, Information Building** | |
| Zeping Lu (Mobile Group) | **Logging in Flash-based Database Systems**<br>Abstract:<br>Synchronous transactional logging is the central mechanism for ensuring data persistency and recoverability in database systems. In this report,we discussed the solutions about exploiting different kinds of flash drives for synchronous logging and the recovery processing technologies related with them. |
| Xiangmei Hu (Web Group) | **Location-based Database Selection**<br>Abstract:<br>Location_based database selection is a new topic,This report mainly gives an introducton about this topic,including why we choose this topic,what the problem is,some related work and how to solve the problem. |
| Jing Zhao (Web Group) | **Snippet of Structured Data**<br>Abstract:<br>It is expected that more and more people will search the web when they are on the move. But there are many limitations when we browsing the web page in mobile devices, especially small screen. A record in database usually contain lots of information, which is not useful for user and is so much for small screen. So we try to extract the most useful attributes to return to user. |
| **2009.06.20 Venue: FL1, Meeting Room, Information Building** | |
| Qingsong Guo Wei Wang(XML Group) | **XML Keyword-Search engine**<br>Abstract:<br>XML has already became the de-facto of data exchange. So, how to query XML data is becoming very important. We can use the query language XQuery and XPath, which is the standard query language of XML recommended of W3C, to get what we need. But the user must be familiar with the query languages, and know the content and structure of XML data at first, so that the users can write the accurate query. It is not easy for most users, and it forcing the study of XML keyword-search, With it, we needn't learn the XML query language, |

| | |
|---|---|
| | and also, we needn't known the content and structure of XML. It make the query easier. The main features of next edition of OrientX(edition 4.0) is to supprot the keyword-search, in the presentation, qingsong guo analized the existing XML keyword-search engine and made a comparison and get their features in common . And based it, we defined the main features of OrientX 4.0 . Wei wang analized the key technologies of xml keyword-search, such as the priciple and algorithms of computing SLCA, the ranking of query results. |
| **2009.06.13 Venue: FL1, Meeting Room, Information Building** | |
| Lizhen Fu (XML) | **Query Processing over Graph-structured XML Data**<br>Abstract:<br>When XML documents are modeled as graphs, many research issues arise. In particular, there are many new challenges in query processing on graph-structured XML documents because traditional query processing techniques for tree-structured XML documents cannot be directly applied. |
| Yulei Fan (Mobile Group) | **MVCC on Flash Memory**<br>Abstract:<br>First, Flash has the characteristic of Out-of-Place Updating, which lead to multiple version of data on Flash. Second, I introduce the basic priciple and some protocols of MVCC, such as MVSR, MVCR, MVTO, MV2PL and so on. Finally, I present some information of transaction in BDB and PG. |
| **2009.06.06 Venue: FL1, Meeting Room, Information Building** | |
| Xiao Pan (Mobile Group) | **Location,Location, Location**<br>Abstract:<br>This talk focuses on the dicussion of Keynote of Christian S. Jensen on MDM2009. |
| Yukun Li (Web Group) | **C-Query: Context-based Query in Personal DataSpace**<br>Abstract:<br>Many users need to refer to content in existing files (pictures,tables, emails, web pages and etc.) when they write documents(programs, presentations, proposals and etc.), and often need to revisit the referenced files for review, revision or reconfirmation. In this paper, we propose an efficient method for users to revisit these refferenced files by identifying a context-based refference relationship. |
| **2009.05.23 Venue: FL1, Meeting Room, Information Building** | |

| | |
|---|---|
| WangWei GuoQingsong (XML Group) | **OrientX system development report**<br>Abstract:<br>The main features of OrientX3.5 version and its implementation. |
| **2009.05.16 Venue: FL1, Meeting Room, Information Building** | |
| Da zhou (Mobile Group) | **Random Write Optimization for SSD**<br>Abstract:<br>Random write of SSD has low IO performance when compared with sequential/random read and write. This paper propose a novel method to avoid the low performance of random write. |
| Xian Tang (Mobile Group) | **buffer management policy**<br>Abstract:<br>In this talk, I introduced several interesting buffer management algorithms, including some algorithms which work well on disk-based DBMS, others are buffer management algorithms on flash-based DBMS. |
| **2009.04.25 Venue: FL1, Meeting Room, Information Building** | |
| Zhongyuan Wang (Web Group) | **An Indexing Framework for Efficient Retrieval on the Cloud**<br>Abstract:<br>The emergence of the Cloud system has simplified the deployment of large-scale distributed systems for software vendors. The Cloud system provides a simple and unified interface between vendor and user, allowing vendors to focus more on the software itself rather than the underlying framework. Existing Cloud systems seek to improve performance by increasing parallelism. This paper explores an alternative solution, proposing an indexing framework for the Cloud system based on the structured overlay. Its indexing framework reduces the amount of data transferred inside the Cloud and facilitates the deployment of database back-end applications. |
| Xiangyu Zhang (Web Group) | **Data Management in the Cloud - Limitations and Opportunities**<br>Abstract:<br>Analysed data management applications that are suitable to move to the cloud platform and discussed remaining challenges of such movement. |
| **2009.04.18 Venue: FL1, Meeting Room, Information Building** | |
| Junfeng Zhou (XML Group) | **MCN: A New Semantics Towards Effective XML Keyword Search**<br>Abstract:<br>In this talk, We propose a new XML Keyword Search Semantics aiming at capturing meaningful results while avoiding returning meaningless |

| | |
|---|---|
| | results. This contribution is based on the observation that when talking about relationship between data elements, users query intension is always based on the relationship of real word entities. |
| Fangjiao Jiang (Web Group) | **Selectivity Estimation for Exclusive Query Translatio in Deep Web Data Integration**<br>Abstract:<br>In Deep Web data integration, some Web database interfaces express exclusive predicate,which permits only one predicate to be selected at a time. Accurately and efficiently estimating the selectivity of each Qe is of critical importance to optimal query translation. In this paper, we mainly focus on the selectivity estimation on infinite-value attribute which is more difficult than that on key attribute and categorical attribute. We start with two observations |
| **2009.04.11 Venue: FL1, Meeting Room, Information Building** | |
| Yukun Li (Web Group) | **Summary of ICDE2009 keynotes**<br>Abstract:<br>This slides give a summary on three keynotes of ICDE2009. |
| Da Zhou (mobile Group) | **ICDE 2009 Introduction**<br>Abstract:<br>ICDE is a very important international meeting about data management. In this conference, there are a lot of works related to flash-based database. transaction becomes an important topic in this field. |
| Zhichao Liang (Flash Group) | **Demo in ICDE 2009 Conference**<br>Abstract:<br>WEST(Web Entity Search Technologies),instead of returning webpages that are related to any people who happened to have the queried name,is to output a set of clusters of webpages,one cluster per each distinct person.Fa is a new system for automated diagnosis of system failures that is designed to address the SLO violations.UQLIPS is a Web-based integrated platform which performs online detection of near-duplicate occurrences over continuous video streams,as well as retrieval of near-duplicate clips from segmented video collections. |
| **2009.04.04 Venue: FL1, Meeting Room, Information Building** | |
| Xiao Pan (Mobile Group) | **Distortion-based Anonymity towards Continuous Query in Mobile Services**<br>Abstract:<br>Privacy preservation has recently received considerable attention for |

| | |
|---|---|
| | location-based mobile services. A lot of location cloaking approaches have been proposed for protecting the location privacy of mobile users. In this paper, we present continuous query privacy disclosed and worst QoS resulting from anonymizing continuous query. |
| Chunjie Zhou (Mobile Group) | **Complex Event Detection in Pervasive Computing**<br>Abstract:<br>In pervasive computing environments, wide deployment of sensor devices has generated an unprecedented volume of atomic events. However, most applications such as healthcare, surveillance and facility management, as well as environmental monitoring require such events to be filtered and correlated for complex event detection. Therefore how to extract interesting, useful and complex events from low-level atomic events is becoming more and more important in daily life. Due to the increasing importance of complex event detection, this paper proposes a framework of Complex Event Detection and Operation (CEDO) in pervasive computing. It gives an event model and extends current detection by incorporating temporal and spatial settings of events and different levels of granularity for event representation. We first show research issues, related works, and main research problems in this area. Then our current research works and the preliminary results are introduced. Finally, the research plan of my PhD project is presented for discussion. |
| **2009.03.28 Venue: FL1, Meeting Room, Information Building** | |
| Fangjiao Jiang (Web Group) | **Deep Web Integration:Querying Structured Data on the Deep Web**<br>Abstract:<br>In this report, I will introduce the background of Deep Web, the key technologies of Deep Web data integration and the active research groups. Then I will compare the metaquerier with metasearch engine. Finally I will give the research problems in the future. |
| Xiangmei Hu (Web Group) | **Database selection**<br>Abstract:<br>Database selection is a important topic,this report gives an introduciton to database selection and then introduces our new problem. |
| **2009.03.21 Venue: FL1, Meeting Room, Information Building** | |
| Yukun Li (Web Group) | **CoreSpace: A personal dataspace framework based on user activity**<br>Abstract:<br>Present a new framework of personal dataspace by hightlighting relationship between users and average objects, which provides more |

| | |
|---|---|
| | effective approaches of querying personal dataspace. |
| Yubo Kou (Web Group) | **An efficient method to Identify personal task**<br>Abstract:<br>Present a new method to identify personal task based on user access activity. |

**2009.03.14 Venue: FL1, Meeting Room, Information Building**

| | |
|---|---|
| Fei Huang (Cloud Computing) | **Research Report on Map/Reduce Framework Based on Hadoop**<br>Abstract:<br>Map/Reduce is the crucial algorithm of Hadoop. It is a easy but powerful algorithm that can solve the problems based on mass data. In this report,I will introduce the concept of Hadoop and Map/Reduce, then the detail of how the Map/Reduce framework do jobs. |
| Yi Hu (Web Group) | **Introduction to HBase**<br>Abstract:<br>As sub-project of Hadooop, HBase focus on providing storage for the Hadoop Distributed Computing Environment. HBase is a table coloum-oriented operating. Its three-layer file system provides the feasible scheme for the distributing data storage while its three-layer architecture solves the problems of region assignment and region location. To get intuitionistic understanding of HBase, comparison with MySQL has been made in the test. |
| Wei Chen (Web Group) | **The Progress of C-DBLP's Development and Future Plans**<br>Abstract:<br>The develop team of C-DBLP system has added some attractive functions and features to the site based on user's feedback and researching demand since the release of C-DBLP. Besides, we are working on some interesting problems such as Name Disambiguation and Mining of Relations among Authors. This report presented the progress of C-DBLP's development and showed intuitive approaches to the research problems in C-DBLP. Also, we made a detailed plan for future work in C-DBLP. |

**2009.03.07 Venue: FL1, Meeting Room, Information Building**

| | |
|---|---|
| Linlin Jia (Web Group) | **Study on Fast Approxmate Membership checking**<br>Abstract:<br>Introduce ISH for approximate membership checking and analyze its disadvantage. We propose a new index and a corrresponding algorithm, the experiments indicate that the new method is more efficient than ISH. |

| | |
|---|---|
| Junjin Xu (XML Group) | **String Similarity**<br>Abstract:<br>This report introduces the methods about counting string similarity, including edit distance and gram_based similarity. |

**2009.02.28 Venue: FL1, Meeting Room, Information Building**

| | |
|---|---|
| Jing Zhao (Web Group) | **Faceted Search**<br>Abstract:<br>A introduction to faceted search, including the evolution of faceted search, the differences between faceted search and navigational search, direct search, and differences between cluster, tag and facet. |
| Wei Chen (Web Group) | **Automatic Construction of Facet Hierarchies**<br>Abstract:<br>Facet hierarchies are the main forms of data organization in facet search system. They are used to support facet-based navigation and refine the search results through different facets. The construction of facet hierarchies is one of the most important research topics in facet search. Since most facet hierarchies in current systems are built mannually, the automatic construction method is in great need. This presentation addressed W. Dakka and P. G. Ipeirotis's research progress in automatic construction of facet hierarchies. |

**2009.01.11 Venue: FL1, Meeting Room, Information Building**

| | |
|---|---|
| Junfeng Zhou (XML Group) | **Survey of XML Database Technology**<br>Abstract:<br>In this talk, I give the main topics about XML database and explain the existing solutions using simple examples. |
| Lizhen Fu (XML Group) | **Graph DataBases**<br>Abstract:<br>This presentation introduces some rearch hotspots on Graph DataBases，including the construction of the index, the processing of containment queryquery and reachability query answering. |

# 实验室网站


http://idke.ruc.edu.cn/wamdm

# 实验室成员

## Faculty Members

**XiaoFeng Meng**
**孟小峰**
**博士，教授，博导**
**WAMDM 实验室负责人**

**Jiaheng Lu**
**陆嘉恒**
**博士，副教授**

**Nan Yang**
**杨 楠**
**博士后，副教授**

**Qing Liu**
**刘 青**
**博士，副教授**

**Yunpeng Chai**
**柴云鹏**
**博士，讲师**

**Gang Yang**
**杨 刚**
**博士，讲师**

## Ph.D. Candidates

**Yukun Li**
**李玉坤**

**Da Zhou**
**周大**

**Xiao Pan**
**潘晓**

**Xian Tang**
**汤显**

**Chunjie Zhou**
**周春姐**

**Lizhen Fu**
**富丽珍**

**Yulei Fan**
**范玉雷**

**Ruxia Ma**
**马如霞**

**Yinjie Shi**
**史英杰**

**Jinzeng Zhang**
**张金增**

**Zheng Huo**
**霍峥**

## M.Sc. Students

**Zhongyuan Wang**
**王仲远**

**Jing Ai**
**艾静**

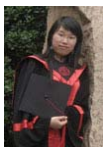
**Junjin Xu**
**徐俊劲**

**Xing Hao**
**郝兴**

**Xiangyu Zhang**
**张相於**

**Yubo Kou**
**寇玉波**

**Qingsong Guo**
**郭青松**



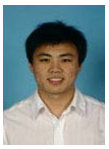
**Jing Zhao**
**赵婧**

**Xiangmei Hu**
**胡享梅**

**Zeping Lu**
**卢泽萍**

**Wei Wang**
**王伟**

**Wei Chen**
**陈威**



**Bingbing Liu**
**刘兵兵**

**Zhichao Liang**
**梁智超**

**Yi Huang**
**黄毅**

**Xiaoying Qi**
**綦晓颖**

**Ying Lv**
**吕瑛**

**Haiping Wang**
**王海平**

## Undergraduates

**Zhe Chen**
**陈喆**

# 已毕业学生

## 2009 年毕业生去向

| 姓名 | 学历 | 时间 | 毕业去向 |
|---|---|---|---|
| 周军锋 | 博士 | 2009年7月 | 燕山大学 |
| 姜芳芃 | 博士 | 2009年7月 | 徐州师范大学 |
| 贾琳琳 | 硕士 | 2009年7月 | 中国农业银行 |
| 黄静 | 硕士 | 2009年7月 | 中国工商银行软件开发中心 |
| 朱金清 | 硕士 | 2009年7月 | 百度 |
| 王伟 | 硕士 | 2009年7月 | 百度 |
| 向锂 | 硕士 | 2009年7月 | 中化集团石油中心 |

内部资料，妥善保存

网络与移动数据管理实验室
地址：中国人民大学原信息楼一层
网址：http://idke.ruc.edu.cn/wamdm
电话：010-62512719
传真：010-62514798
编辑：霍峥 马如霞 史英杰 张金增