数据存取模块概要设计说明书

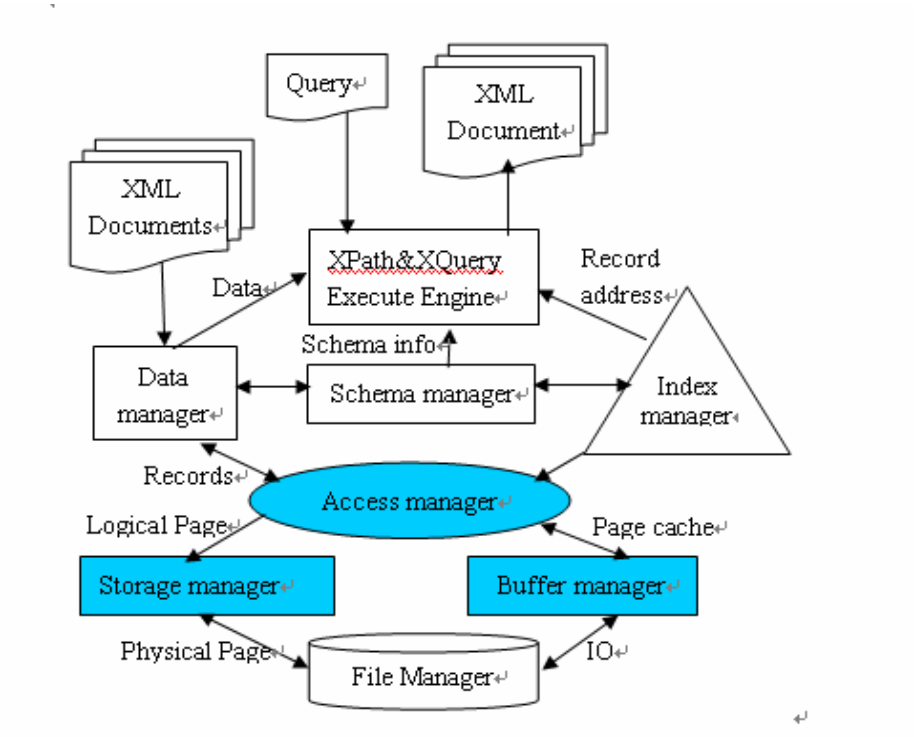
负责人：吴岑
编写人：安靖
系统版本号：OrientX Version 1.5
完成时间：2003/12/16
开发单位：中国人民大学 IDKE 实验室 XML 工作组

1. 引言

本说明书向用户介绍 OrientX 系统中的数据存取模块。该模块对 StorageManager 和 BufferManager 模块进行包装，向上层数据管理模块提供统一的接口。第 2 部分介绍模块的背景知识，模块的功能，以及此模块在整个系统中的地位和作用。第 3 部分介绍整个模块的处理思想，模块内部由哪些小模块构成，以及它们之间的关系。第 4 部分介绍为上层数据管理模块提供的接口。第 5 部分讲述模块内的数据结构的设计和主要成员函数的功能。最后第 6 部分介绍该模块的错误处理机制。

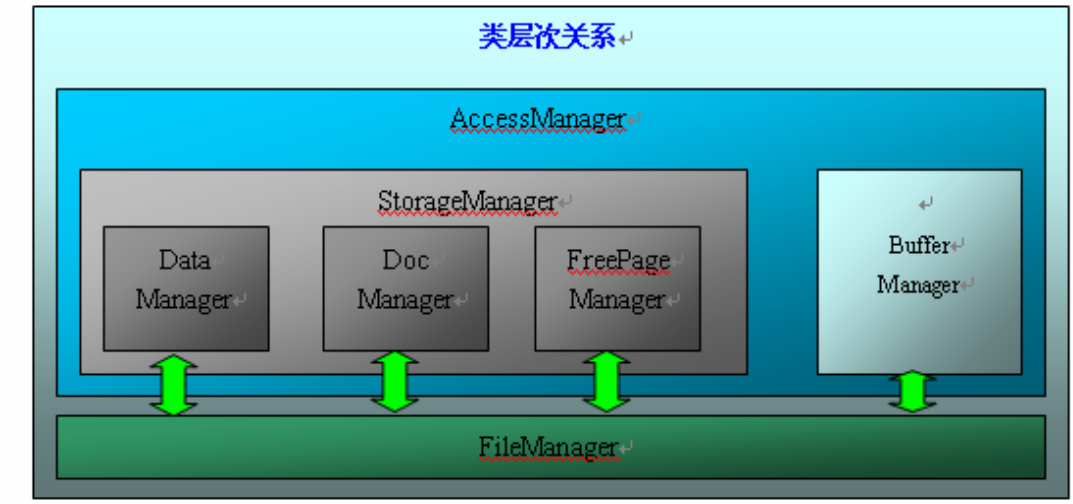
2. 概述

模块功能：对 StorageManager 和 BufferManager 模块进行包装，向上层数据管理模块提供统一的接口。使上层直接通过这个接口来存取数据。



3. 总体设计

该模块有两个成员对象，分别是 StorageManager 和 BufferManager。前者负责将数据存储到磁盘上或者从磁盘上读取数据，后者负责内存缓冲的分配。由 AccessManager 向上层提供统一的接口。因为要处理的数据可能很多需要为之分配缓冲，所以直接将这两个模块统一起来，上层就不用考虑这其中的细节，只用通过 AccessManager 来读写数据就可以了。
其内部类间的关系：



4. 接口设计

为上层数据管理模块提供服务的接口，包括数据集的生成和删除，数据文档的生成和删除等，这些都是调用 StorageManager 模块实现，是其所有 public 函数的外包。寻找空闲页分配内存缓冲区，将缓冲区中的内容刷出保存等，这些是由 BufferManager 模块实现。

5. 数据结构设计

5．1 逻辑结构设计

两个成员对象：StorageManager 和 BufferManager 。

5．2 物理结构设计

主要类成员函数的实现（算法）

函数名	函数功能
int initiate()	调用两个成员对象的initiate函数，进行初始化。系统启动时被调用。
int clear()	调用两个成员对象的clear函数，在退出系统前被调用。

以下是关于数据集的函数，直接去调用StorageManager中的同名函数，参数直接传递给对应的函数。

函数名	函数功能
int CreateDataSet(char* DataSetName);	生成指定名称的数据集
int DeleteDataSet(char* DataSetName);	删除指定名称的数据集
int OpenDataSet(char* DataSetName);	打开指定名称的数据集
NX_SetList GetSetList();	得到数据库中所有的数据集的名字列表

以下是关于Document的函数，也是直接调用StorageManage成员对象。

函数名	函数功能
int CreateDoc(char* DataSetName,char* DocName,int Type);	在指定数据集中生成指定名称的文档
int DeleteDoc(char* DataSetName,char* DocName);	删除指定数据集中的指定文档
int OpenDoc(int SetId,char* DocName);	打开指定的文档
NX_DocList GetDocList(int SetId);	得到指定数据集中的所有文档名
int ReadDocStorageType(char* dataSetName,char* docName,StorageMode & sType);	得到指定文档的存储类型信息，是 DEB，CEB，DSB，CSB 中的一种
int WriteDocStorageType(int setId,int docId, StorageMode sType);	保存指定文档的存储类型信息
int ReadDocRoot(char* DataSetName,char* DocName,int& SetId,int& DocId,int& lpno,int& RecordNo);	取得指定文档的根节点的存储位置
int WriteDocRoot(char* DataSetName,char* DocName,int lpno,int RecordNo);	保存指定文档的根节点的存储位置
int WriteDocEnd(char* dateSetName,char* docName,int lpNo);	保存指定文档的使用的最后页号
int ReadDocEnd(char* dataSetName,char* docName,int& setId,int& docId,int& lpNo);	读取指定文档的使用的最后页号
int WriteBlockRoot(char* dataSetName,char* docName,int lpNo,int recNo);	保存指定文档的 blocktree 的根节点的存储位置
int ReadBlockRoot(int& setId,int& docId,int& lpNo,int& recNo);	读取指定文档的 blocktree 的根节点的存储位置

以下是关于逻辑页的函数，直接调用 StorageManager 的同名函数

函数名	函数功能
int FindEmptyPage(int SetId);	在指定数据集中找到空闲页，返回找到的页号
int FindUsablePage(int SetId);	在指定数据集中找到可用的页，返回找到的页号
int MarkFullPage(int SetId,int lpno);	将指定的逻辑页的状态标记为用满
int MarkEmptyPage(int SetId,int lpno);	将指定的逻辑页的状态标记为空闲
int MarkUsablePage(int SetId,int lpno);	将指定的逻辑页的状态标记为可用

以下是内存缓冲页的函数，直接调用 BufferManager 的同名函数

函数名	函数功能

short GetPage2Buf(int setID, int lpno,UserModes userMode);	将指定的逻辑页读入内存，并配置对应得缓冲页，
short FreeBuf(short bufNo);	释放缓冲页
short GetNewBuf (int setID, int lpNo,UserModes userMode);	为指定的逻辑页分配新的缓冲页，在第一次读入某个逻辑页时调用
short ReadBuf(short bufNo, short offset, short len, void *local);	从指定的缓冲页的指定位置开始读取指定长度的内容
short WriteBuf(short bufNo, short offset, short len, void *local);	在指定的缓冲页的指定位置开始，写入指定的内容
short FlushBuf(short bufNo);	将缓冲页中的内容刷出到磁盘

6. 出错处理

目前的错误处理策略是：预定义了部分错误代码，如果出现这些错误，则打印错误信息到终端，并层层返回。