**CatalogManager 模块设计说明**

作者：2007级计算机科学与技术专业

陈矫彦 3071102293

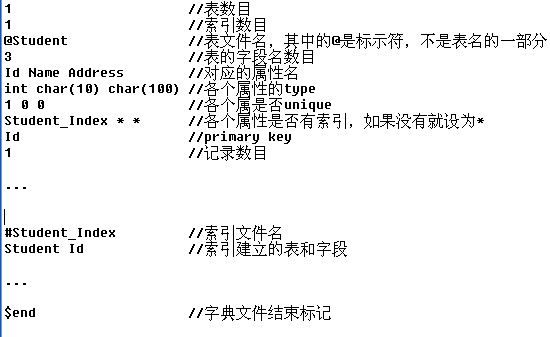
CatalogManager类负责管理数据库的所有模式信息，包括：

1. 数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。
2. 表中每个字段的定义信息，包括字段类型、是否唯一等。
3. 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。
4. 数据表中的记录条数。
5. 数据库内已建的表、索引的数目。

Catalog Manager还必需提供访问及操作上述信息的接口，供BufferManager和API模块使用。

为减小模块之间的耦合，Catalog模块采用直接访问磁盘文件的形式，不通过Buffer Manager，Catalog中的数据也不要求分块存储，存储方式是ASCII码直接存储。

数据字典文件格式如下（其中的注释和’…’是为了表达而加的，实际是没有的）：



**数据结构：**

1. struct Index

{

string indexName; //存放索引名

string searchTable; //存放索引对应的表名

string searchKey; //存放索引对应的关键属性

};

功能：作为记录索引文件信息存放格式。

1. struct Table

{

string tableName; //表名

int recordNum; //存放记录数目

int collNum; //存入字段数目

int recordLen; //存入记录长度

string primKey; //存放主键

vector<string> collName; //存放对应的字段名

vector<string> collType; //存放对应的字段类型

vector<int> collUniq; //存放对应的字段是否惟一

vector<string> collIndex; //存放对应字段上的索引信息

};

功能：作为表文件信息存放格式。

**成员变量：**

1) int tableNum; //表数目

2)int indexNum; //索引数目

3)vector<Table> tv; //表列表

4)vector<Index> iv; //索引列表

**成员函数：**

**具体实现按照语句类型进行分别处理：**

创建数据库语句

语法:

CREATE TABLE 表名 （

列名 类型

列名 类型

…………..

PRIMARY KEY(列名)

）；

根据Interpreter处理生成的初步内部数据形式，提取表名，验证是否有重名现象。若无则建立表文件，同时将表的各种信息记录在数据字典内，若有主键定义，应对主键生成索引，并将索引信息记录在数据字典内。若有重名表，则创建表命令失败，并打印出错信息。其中用到的两个重载函数如下：

int addTable(string tableName,vector<string> col,vector<string> type,vector<int> uniq);

int addTable(string tableName,vector<string> col,vector<string> type,vector<int> uniq,string primKey);

创建索引语句

语法：

CREATE INDEX 索引名 ON 表名（列名）；

根据Interpreter处理生成的初步内部数据形式，提取表名及列名，在数据字典中查找有无该表及属性的定义，且验证是否可对该属性建立索引。若可建，则创建索引文件，同时将索引信息记录在数据字典内。若不可建索引，则根据不同类型打印出错信息供用户参考。其中用到的函数如下：

int addIndex(string indexName,string tableName,string colName);

选择语句

语法：

SELECT \* FROM 表名； SELECT 列1，列2，….列n FROM 表名；

SELECT \* FROM 表名 WHERE 条件； SELECT 列1，列2，…列n FROM 表名 WHERE 条件

Select 语句并不影响字典信息，也就是不用更改字典信息，所以没有直接的函数与之对应。

插入记录语句

语法：

INSERT INTO 表名 VALUES ( 值1，值2，…….,值n)；

根据Interpreter处理生成的初步内部数据形式，提取表名及各属性值，在数据字典中查找该表的定义，并验证各属性值是否满足各属性的数据类型定义。同时将字典信息中的record number增加一。任何的不符都将打印出相应的出错信息供用户参考。对应的函数如下：

int insertRecord(string tableName,int recordNum);

删除记录语句

DELETE FORM 表名；

DELETE FORM 表名 WHERE 条件；

根据Interpreter处理生成的初步内部数据形式，提取表名，在数据字典中查找该表的定义。若有，删除该表文件内的所有记录并更新该表在数据字典中的信息,同时须查找建立在该表之上的所有索引，并将所有索引文件内的节点删除且须更新索引文件在数据字典中的信息。若在数据字典中无该表的定义，则须打印出必要的出错信息供用户参考。用到的两个生载函数如下：

int deleteValue(string tableName);

int deleteValue(string tableName,int deleteNum);

删除索引语句

语法：

DROP INDEX 索引名；

根据Interpreter处理生成的初步内部数据形式，提取索引名，在数据字典中查找有无该索引的定义。若有，则删除该索引在数据字典中的信息，并删除索引文件。若无，则打印出相应的出错信息供用户参考。用到的函数如下：

int dropIndex(string index);

**删除表语句**

语法：

DROP TALBE 表名；

根据Interpreter处理生成的初步内部数据形式，提取表名，在数据字典中查找有无该表的定义。若有，则先查找该表上的所有索引并删除，然后删除该表在数据字典中的信息，最后删除该表文件。若在数据字典中无该表的定义，则打印出相应的出错信息供用户参考。

int dropTable(string tableName);

其它函数的功能描述如下：

1. void getFileInfo(string fileName,int \* n);

功能：查找文件信息，是否为索引或者表，如果后者，则返回包括表长度和记录数。其中返回的信息是放在数组指针int \* n中。

1. int findFile(string fileName);

功能：查找文件，返回是没有文件或者表文件或者索引文件。

3) int existIndex(string tableName,string col);

功能：判断在对应表对应字段上是否有索引

4) int writeBack();

功能：将内存中的字典信息存入文件。如果写成功返回1，否则返回0.

5) int calcuteLenth(string tableName);

功能：通过输入的表名返回这个表的记录长度。

int calcuteLenth(vector<string> v);

功能：通过输入的属性列表，返回对应的记录长度。

int calcuteLenth2(string type);

功能：通过输入的字段，返回字段的长度。

6) int getRecordNum(string tableName);

功能：通过表名查找表的记录数。

7) vector<string> getCollName(string tableName);

功能：通过表名返回这个表的属性列表。

8) vector<string> getCollType(string tableName);

功能：通过表名返回字段类型列表。

9) vector<int> getCollUniq(string tableName);

功能：通过表名返回字段是否惟一列表

10) vector<string> getCollIndex(string tableName);

功能：通过表名返回字段索引列表

11) CatalogManager();

功能：从已经有的字典文件中读入文件信息放入内存，如果失败则提示出错。创建一个CatalogManager对象实例。

~CatalogManager(){};

通过CatalogManager 模块的处理 已经完成了MiniSQL 的数据字典处理功能，通过API接口的功能，即可将信息传递给RecordManager模块、IndexManager模块和BufferManager模块，为这三个模块的实现提供了充足的字典信息。