模块设计报告

章海威

1.概要

我主要负责CatalogManager模块和与此相关的API的设计和编码。

- *Catalog Manager* 负责管理数据库的所有表的元数据信息，包括：
    - 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。
    - 表中每个字段的定义信息，包括字段类型、是否唯一等。
    - 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。

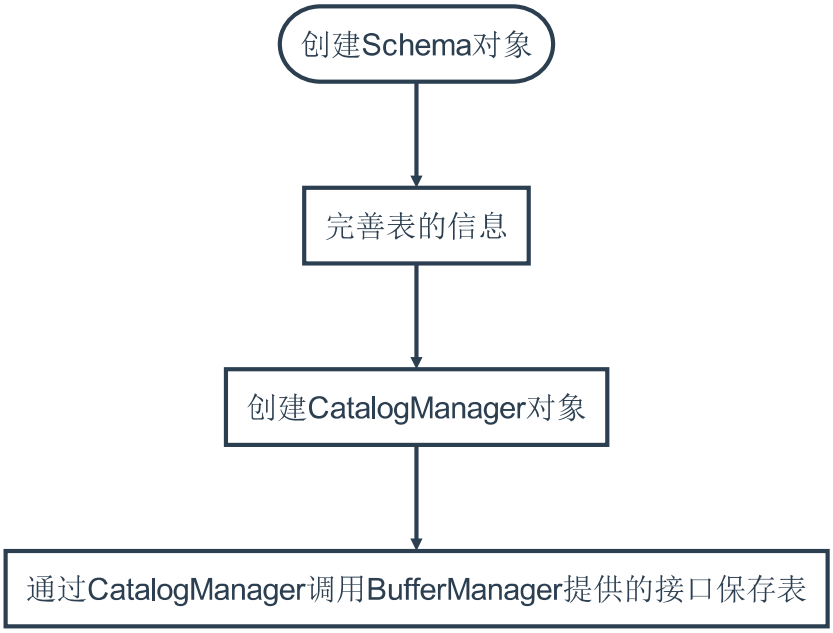
    Catalog Manager 还提供了对上述信息的操作，包括创建表，删除表，查询查询表的信息，插入字段，创建主键等。

- *API* 负责执行SQL语句，根据Interpreter解释生成的命令内部表示，通过调用Catalog Manager提供的接口来实现相关的操作。我主要实现了API层中与表的操作相关的接口，包括`createTable`，`dropTable`，`createIndex`，`dropIndex`，`showTables`等。

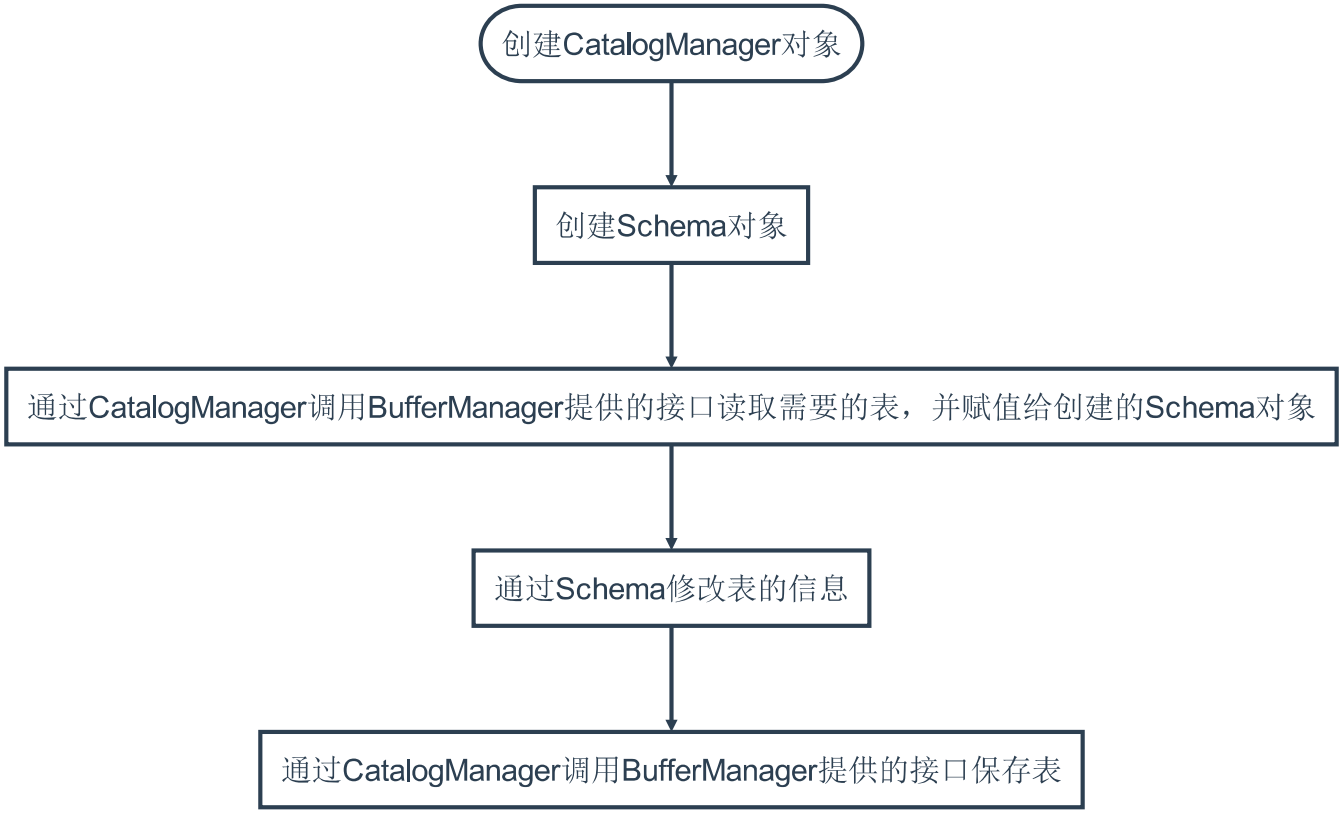
2.结构

为了便于对表的操作以及实现良好的封装性，我定义了`Attribute`、`Schema`和`Catalog Manager`三个类来管理数据库中的表的信息。其中，Attribute 定义了表中每个字段的信息，包括字段名、类型、长度、是否唯一等。Schema 定义了单个表的所有信息，包括表名、字段列表、主键、索引列表等，对单个表的修改操作都是在Schama类中实现的。而对整张表的创建、删除、读取、保存等操作则要通过CatalogManager类来完成。

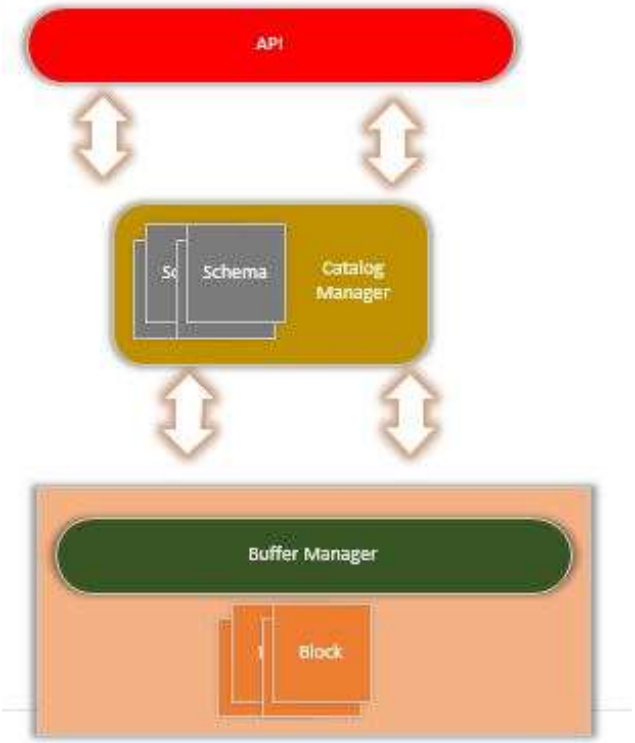
总得来说，创建一张新的表的过程如下：



修改一张表的过程如下：



所涉及到的模块有API、CatalogManager、BufferManager, 各个模块的关系如下图所示：



3.接口及实现

CatalogManager模块

（1） Class Attribute

成员变量：

Type	Name	Description
std::string	attrName	字段名
std::string	type	字段类型
int	length	字段长度
std::string	index	该字段上的索引
bool	unique	是否唯一

Atrribute作为数据结构使用，没有提供操作的接口

（2） Class Schema

成员变量：

Type	Name	Description

std::string	tableName	表名
std::vector `<std::string>`	attrNames	字段名字列表
std::map `<std::string, Attribute>`	attributes	字段列表
std::map `<std::string, std::string>`	indecies	索引列表
std::string	primaryKey	主键

成员函数：

Return	Method name	Describetion
static Schema	createSchema(std::string schString)	construct the schema from a string, it's `private`
std::string	getName()	get the table name
std::string	toString()	convert the schema to a string for the convenience of print and store
void	addAttribute(std::string attrName, std::string type)	add a new attribute to schema
bool	isAttrExists(std::string attrName)	return true if the schema has the attribute
bool	isIndexExists(std::string indexName)	return true if the schema has the index
std::string	getIndex(std::string attrName)	return the index on the attribute
void	addIndex(std::string indexName, std::string attrName)	add index to the attribute
void	deleteIndex(std::string indexName)	delete the index form the attribute
void	setPrimaryKey(std::string attrName)	set an attribute as the primary key of the table
std::string	getPrimaryKey()	return the primary key
void	setUnique(std::string attrName)	set the attribute to unique
bool	isUnique(std::string attrName)	return true if the attribute is unique
void	setType(std::string attrName, std::string type)	set the type of an attribute
std::string	getType(std::string attrName)	return the type of the attribute

void	setLength(std::string attrName, int length)	set the length of an attribute
int	getLength(std::string attrName)	return the length of the attribute
Attribute	getAttribute(std::string attrName)	find an attribute by it's name
std::vector	getAttributes()	return a list of attribute names
void	copyAttributes(std::vector& container)	copy the attributes to a vector
static std::string	intToString(int i)	used to convert an int to a string
static int	stringToInt(std::string str)	used to convert a string to an int
static std::vector	Split(std::string str,std::string pattern)	split a string and store the result into a vector

（3） Class CatalogManager

成员变量：

Type	Name	Description
std::map `<std::string, Schema>`	schemaQueue	表的列表，存储了所有的表
static CatalogManager*	instance	CatalogManager实例

成员函数：

Return	Method name	Describetion
static CatalogManager*	getInstance()	实例化CatalogManager对象
void	store(Schema sch)	保存新建的表
void	saveChange(Schema sch)	保存表的更改
void	drop(std::string tableName)	删除表
Schema	get(std::string name)	通过表的名字取得表
std::list	getTables()	获得所有表的名字列表
bool	isExist(std::string name)	判断表是否存在

CatalogManager采用单例模式，只能通过getInstance()函数来实例化，程序运行中只有一个实例化对象。CatalogManager构造时会读入所有的表，并以map的形式保存在内存中，析构时将所有的表写回到文件中。

API模块

API模块中，我实现的接口如下：

Return	Method name	Describetion
void	ShowTables()	输出所有表的名字
void	DescribeTable(std::string tname)	输出表的信息
void	createIndex(std::string indexName, std::string tableName, std::string attrName)	创建索引
void	dropIndex(std::string indexName, std::string tableName)	删除索引
void	CreateTable(Schema sch)	创建表
void	DropTable(std::string name)	删除表

简单介绍一下`createIndex`和`createTable`的实现：

- createTable:

```
首先，通过CatalogManager创建一个新的Schma并保存
然后，通过RecordManager创建一张空的表等待以后插入数据
```

- createIndex:

```
1.通过CatalogManager找到相应的Schema，然后在相应的字段上添加索引
2.通过RecordManager获取相应的表
3.遍历表上的所有记录，通过IndexManager将该字段的值插入到B+树中
4.通过IndexManager将建好的索引保存到文件中
```