Basic 基类模块设计报告

3120101973 陆洲

一、模块概述

在 MiniSQL 中，其他模块均有一部分功能要实现，而他们每个功能使用的数据单元在 C++中并没有可直接使用的定义。这可能造成混淆和麻烦。我们小组在开发过程中，感到首先需要对数据库常见的概念给予一个统一的定义，因此有了基类模块。

它主要包含属性、记录、表格属性等这些每个其他模块都会使用到的类。

二、主要功能

基类模块的主要功能是规范基本数据单元的使用。以下是指导书中的需求：

数据类型

只要求支持三种基本数据类型：int，char(n)，float，其中 char(n)满足 1 <= n <= 255 。

表定义

一个表最多可以定义 32 个属性，各属性可以指定是否为 unique；支持单属性的主键定义。

三、设计思路

1. 数据库由表格组成，表格再进行拆分是一条条的记录，因此设计 Record 类对表格内的数据起统管作用。

2. 记录里包含几个属性的值，因此设计 AttributeValue 类，还有集中管理的 AttributeInformation 类。

3. 出于进行检索的需要，因此还需要有一个对属性信息进行管理，同时包含表格信息的 TableInformation 类。

4. 在底层的 block 中，每个表格、每个索引都会有自己的地址，我们根据地址来调用它们，因此设计 Address 类。

5. 在数据库中，文件具有自己的格式，比如分成不同的 Block 来管理，同时在每个 block 开头保存着这个 block 专有的元数据，因此设计 DBfile 类规范数据库内部文件格式。

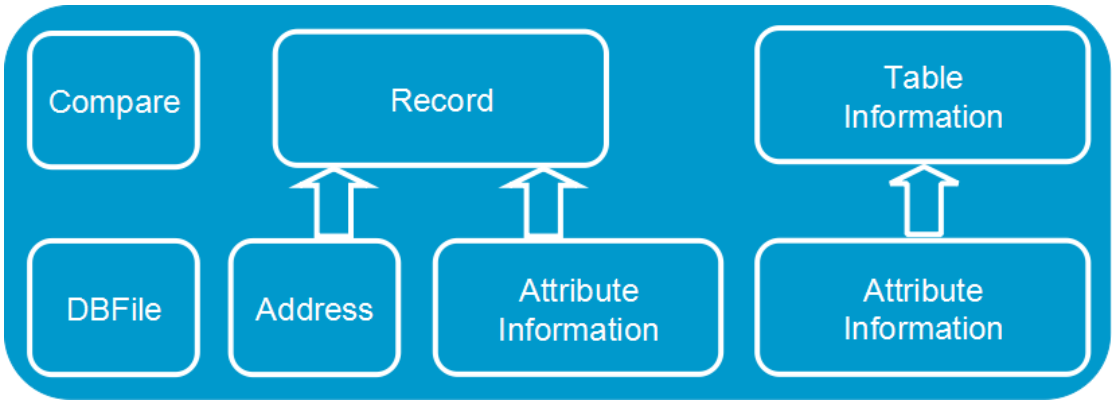
6. 除此之外，在功能设计中，选择语句的条件里可能包括两个属性值的比较，具体包括=、<>、<、>、<=、>=六个操作。这类比较语句出现频率较多，为支持多条件比较查询、避免代码过多重复，因此设计专用于比较的 Compare 类。

四、整体架构

这个模块主要是被其他模块使用，因此类中属性重要性大于方法，大多的类仅需要实现一下拷贝构造、重载‘=’运算符即可。

其中 Address、AttributeValue 类是最基本的，Record、TableInformation 和 AttributeInformation 类则用于管理 AttributeValue 类，其中 TableInformation 类则是通过调用 AttributeInformation 类来管理。

整体设计图如下：



五、关键函数及伪码

1. Record 类

```
class Record
{
public:
    AttributeValue value[32];//每个 record 最多有 32 个属性，之后也会进行检验
    int attributeCount;
    Address address;//record 的起始地址

    //拷贝构造及‘=’运算符重载函数
    ……
};
```

Record 类可以用于所有对 Record 的操作，在 RecordManager、BufferManager、CatalogManager 中尤为重要。

2. AttributeValue 类

```
class AttributeValue
{
public:
    String charValue;
    int charCount;
    float floatValue;
    int intValue;
```

```
    int typeNo; //内部定义了类型，0 为 char，1 为 float，2 为 int

    //拷贝构造及‘=’运算符重载函数
    ……
};
```

属性类是几乎每一个 Manager 都会使用的类，内部的类型若一直读字符串太不方便，因此使用了数字编号。

3. AttributeInformation 类

```
class AttributeInformation
{
public:
    String type;
    int charCount;
    bool unique; //属性是否 unique
    String attributeName;
    bool hasIndex; //在该属性上是否有 index

    //拷贝构造及‘=’运算符重载函数
    ……
    int getSize();//可被调用来确定属性的长度是否符合规范
};
```

该类对属性数据进行管理，对属性拥有的一些特性作了规定，比如属性是否唯一、是否建有索引等。

4. TableInformation 类

```
class TableInformation
{
public:
    AttributeInformation info[32];//最多 32 个属性，因此属性的管理最多也是 32 个
    int attributeCount;
    String tableName;
    String primaryKey;

    //拷贝构造及‘=’运算符重载函数
    ……
    int getTotalSize();//计算表格的大小
};
```

该类对表格数据进行管理，因此有 Primary Key 的定义，它将被定义为 Primary Key 的属性记录下来。

5. Address 及 Compare 类

```
class Address
{
public:
    int blockAddress;//块地址
    int indexAddress;//索引地址
```

```
};

class Compare
{
public:
	CString item;
	int operation; //操作符号内部编号：0=,1<,2>,3<=,4>=,5!=
	int typeNo; //内部定义了类型，0 为 char，1 为 float，2 为 int

	bool campareAll(const AttributeValue& a, int operation);//比较所有操作，返回所需操作
};
```

地址类在底层中应用频繁，是每一条属性的唯一标识。比较类则在属性比较中使用较多。

6.DBFile 类

```
class DBFile
{
public:
	CFile myfile;

	DBFile();
	~DBFile();
	int allocNewBlock();
	int findEnoughSpace(int，  int);//create a space whose size and start address offset are set

};

DBFile::DBFile()
{
	myfile.Open("Date.dat",CFile::modeCreate |
			CFile::modeNoTruncate| CFile::modeReadWrite);
	//创建新的 File，需要总的 Metadata
	if(myfile.GetLength() == 0){
		int content[4096/4];//divide into 1024 line, each line is a word wide
		content[0]=3;  //usage of every content[],??????
		content[1]=4096-3*8-1;
		content[2] = content[4] = content[6] = 8;
		content[3] = 4096-8;
		content[5] = 4096-16;
		content[7] = 4096-24;
		for(int i = 8; i < 4096/4; i++)
			content[i] = -1;//initialize the rest of byte
		myfile.Write((void *)content,4096);
	}
}
```

```
DBFile::~DBFile()
{
	//close the file
	……
}

int DBFile::allocNewBlock()
{
	unsigned int blockCount = myfile.GetLength()/4096;
	unsigned int base = blockCount * 4096;//从已经存了的地址处开始分配新格
	//初始化 Block
	……
	myfile.Seek(……);//磁盘定位
	myfile.Write(……);//写入新 Block，虽然内容是空白的，但相当于做标记
	return blockCount;
}

int DBFile::findEnoughSpace(int offset, int size)
{
	unsigned int blockCount = myfile.GetLength()/4096;
	int array[2];
	if(offset == 0)//不允许偏移为 0 的状况
		offset = 1;

	for(int i = offset; i < block_count; i++){
	    //如果有足够的空间就返回
		……
	}
	//没有足够的空间就新建
	return allocNewBlock();
}
```

该类主要规定了文件里 Metadata 的大小及格式，也规定了数据库中基本单元——Block，为上层的各 Manager 读取、新建、删除定义了统一的格式。包含的函数非常简单，只有新建总文件和查找空间是否充足两个，将更多的拓展和实现交给上层。