详细报告-Record Manager

万京杰　　3130000871

Record Manager负责处理表的数据的插入、查找、删除操作。Record Manager模块从API模块、Catalog模块处接收信息，并利用Buffer Manager所提供的方法对数据在buffer中进行筛选和操作。可以说，Record Manager模块是与各个模块联系都很紧密的一个模块。其具体实现如下四个类：

- Attribute类

```
class Attribute {
public:
    Attribute();
    Attribute(const Attribute &attr);
    ~Attribute();

    int compare (const Attribute &attr);
    int getKeyDataLength();
    void convertToRawData();
    void parseFromRawData();

    bool operator <  (const Attribute &attr);
    bool operator == (const Attribute &attr);
    bool operator >  (const Attribute &attr);
    bool operator <= (const Attribute &attr);
    bool operator >= (const Attribute &attr);

    AttributeType   type;
    int             length;
    int             intdata;
    float           floatdata;
    char            chardata[256];
    char            rawdata[256];
    string          attrName;
};
```

> **Attribute**这个类主要是用于存储一个传入的值的类型与值的一个类，方便后面做比较
> ConvertToRawData和ParseFromRawData两个方法主要是方便后面做拷贝和还原所创建的方法，主要将存在不同地方的数据拷贝到rawdata中去或者从rawdata拷回来。

- RecordPage类

```
class RecordPage: public Page
{
public:
    RecordPage(){pageType = PageType::RecordPage;}
    ~RecordPage(){};

    void writenext(PageIndexType);
    void writebefore(PageIndexType);
    PageIndexType readnext();
    PageIndexType readbefore();

};
```

> **RecordPage**是记录数据的一个类型，除了前八位之外，后面4088位都将用于记录纯数据。
> 前八位中，前四位用于记录下一个Page的PageIndex，后四位用于记录前一个Page的PageInex。
> 因此，分别用read和write两个指针的共四种方法，方便后面维护这个链表

- Tuple类

```
class Tuple{
public:
    Tuple()  {}
    ~Tuple() {}

    void createlist(string);
    void createPage(string);
    void convertToRawData();
    void ParseFromRawData();

    RecordPage page;
    vector<Attribute> list;
};
```

> **Tuple**是一个将前两个类联系在一起的一个重要的类，也是完成从内存数据到可用数据转换的一个重要的类，通过从CatalogManager或者BufferManager中读出两个成员变量中的其中一个，就可以用convertToRawData或者ParseFromRawData的方法得出另一个。

- Table类

```
class Table{
public:
    Table(string);
    ~Table();

    PageIndexType insertTuple(vector<Attribute>);
    void deleteTuple(PageIndexType);

    vector<PageIndexType> scanEqual(int , Attribute);
    vector<PageIndexType> scanNotEqual(int, Attribute);
    vector<PageIndexType> scanLess(int , Attribute);
    vector<PageIndexType> scanGreater(int , Attribute);
    vector<PageIndexType> scanLessEqual(int , Attribute);
    vector<PageIndexType> scanGreaterEqual(int , Attribute);

    vector<PageIndexType> getAll();
    vector<pair<Attribute, PageIndexType>> getAll(int);
    vector<Attribute> getTupleAtPage(PageIndexType);
    void printinfo(PageIndexType);

    PageIndexType head;
    string TableName;

};
```

Table这个类是RecordManager最终实现的体现，他用过接收Attribute类或者RecordPage类（页号），通过Tuple类的方法，实现了RecordManager所需要的一切功能。当然，创建这个Table类的时候需要知道他的Table的名字，方便后面的操作。

insertTuple 负责将一个Attribute所组成的vector插入到内存中，同时返回插入page的页号
deleteTuple得到一个page的页号将该页删除
scan的六个函数输入第几个属性的一个int参数和一个比较的Attribute参数，就将返回由页号组成的vector，方便读取
getAll可以直接返回这个Table的所有页号的vector或者输入第几个属性的一个int参数，来返回页号组成的vector
getTupleAtPage可以得到特定页的所有Attribute组成的vector
printfinfo 将搜索出来的结果输出出来

-对于数据的三大操作，RecordManager实际的实现分别如下：

INSERT：
Insert into (table) value (tuple)

> 用表名创建一个Table的类，然后通过**insertTuple**来插入这个属性值（并不判断属性正确）

SELECT：

1. select * / (attributes) from (table)
   用表名创建一个Table的类，然后通过**getAll**来读取这个属性
2. select * / (attributes) from (table) where (conditions)
   用表名创建一个Table的类，然后通过**scan**的方法来获取所有符合条件的页号
   其中，Conditions可以为多组，实现方法是将多组返回的vector用O（n）的时间merge在一起

DELETE：

1. delete from (table) where (conditions)
   具体查找操作同SELECT中的第2条实现
   其中，Conditions可以为多组，实现方法是将多组返回的vector用O（n）的时间merge在一起
   最后，调用**deteleTuple**来删除这些数据
2. delete from [table]
   用表名创建一个Table的类，然后通过**getAll**来读取所有的页号，再调用**deleteTuple**来删除所有的数据

- 可见，Record Manager在数据库结构中，扮演的是次级任务实现者的角色。之所以这么说，是因为Record Manager的实际工作，在大多数情况下并不是直接对底层的操作，而是借助Buffer Manager和Index Manager的功能来实现数据库的操作，其中buffer则是重中之重，因为所有数据都是通过它经行实际交互的。当然，Record Manager也会需要将得到的数据经行处理，以便向上传递。

测试样例

表中空的时候先测试所有的函数，无问题。

- PageIndexType insertTuple(vector< Attribute >);
  尝试连续插入1000条记录，插入成功

- void deleteTuple(PageIndexType);
  尝试删除其中200条记录，插入成功

- vector< PageIndexType > scan…(int , Attribute);
  尝试多次选出其中的记录，选取成功

- vector< PageIndexType > getAll();
  尝试拿出所有记录，选取成功

- vector< Attribute > getTupleAtPage(PageIndexType);
  尝试拿出一个属性所有值提供给索引，选取成功

- void printinfo(PageIndexType);
  每次选取后使用输出结果，输出成功