Catalog Manager 模块设计说明

作者：2016级软件工程专业

傅诤哲 程浩然 叶慕祈 石磊 钱程

一、Catalog Manager负责管理数据库的所有模式信息，包括：

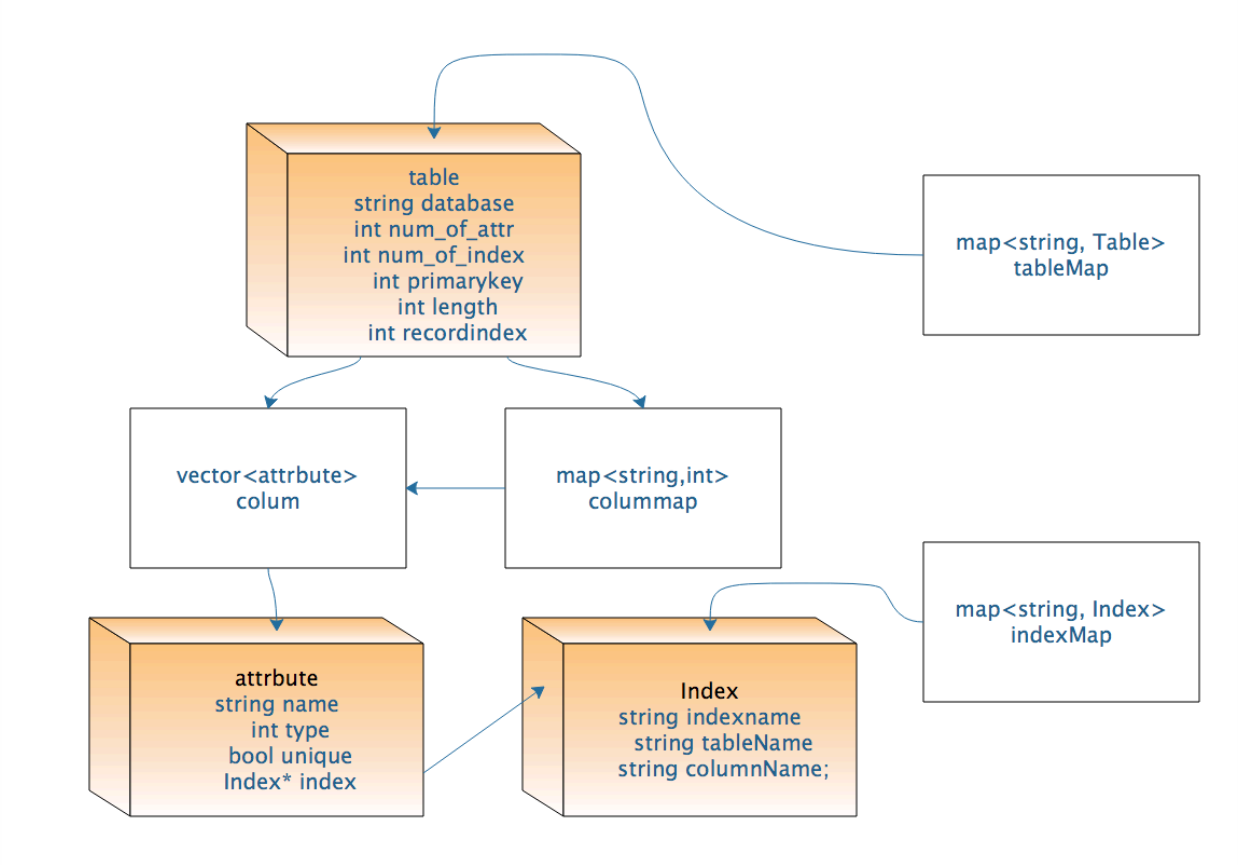
1. 数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。
2. 表中每个字段的定义信息，包括字段类型、是否唯一等。
3. 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。
4. 数据表中的记录条数及空记录串的头记录号。
5. 数据库内已建的表的数目。

Catalog Manager还必需提供访问及操作上述信息的接口，供Interpreter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。

二、模块总体设计思路

Catalog Manager 模块只提供在catalog上的数据记录和保持，通过API接口的功能，才能将信息传递给Record Manager模块和Index Manager模块，为Record Manager模块及Index Manager模块的实现提供了充足的信息。



由内存中的map关系协调所有在系统中的table和index信息，在每个table里有一个map来协调attrbute信息，每个index都是在attrubute上留有一个指针。

三、具体实现

1、声明了宏

```
#define CATA_PATH "***"                                    //catalog的文件储存位置
```

2、声明结构体

1）索引结构体（struct Index）：

定义了索引必须的索引名，建在哪张表的名字和建在表的哪个属性的名字。

2）属性信息结构体（struct attrbute）：

定义了属性名，属性类型，属性是否是唯一的，以及属性上是否有index。

3）表信息结构体（struct Table）：

定义了表所在数据库的名字，表中属性个数，索引个数，主键是第几个属性，属性的vector，属性名和属性在vector中的下标的对应关系，table元组的总长，以及已经记录的位置

```
struct Index{
    string indexname;                                      //索引必须的索引名
    string tableName;                                      //建在哪张表的名字
    string columnName;                                     //建在表的哪个属性的名字
};
struct attrbute{
    string name;                                           //属性名
    int type;                                              //属性类型
    bool unique;                                           //属性是否是唯一的
    Index* index;                                          //属性上是否有index
};
struct Table{
    string database;                                       //表所在数据库的名字
    int num_of_attr;                                       //属性个数
    int num_of_index;                                      //索引个数
    int primarykey;                                        //主键是第几个属性
    map<string,int> colummap;                              //属性名和属性在vector中的
下标的对应关系
    vector<attrbute> colum;                                //属性
    int length;                                            //table元组的总长
    int recordindex;                                       //已经记录的位置
};
```

3、BufferManager类参数声明：

```
string currentdatabase;                                    //当前数据库名
int num_of_table;                                          //当前数据库中的表的数目
map<string, Table> tableMap;                               //table名和Table Struct
的对应关系
map<string, Index> indexMap;                               //index名和Index Struct
的对应关系
```

1. currentdatabase 记录当前数据库名
2. num_of_table记录当前数据库中的表的数目
3. tableMap是table名和Table Struct的对应关系
4. indexMap是index名和Index Struct的对应关系

四、其主要函数的功能描述如下：

1. `bool tableExists(string table);`

   该函数用于检查表是否存在。

2. `bool Create_Table(string tablename);`

   用于建立新的表。

3. `bool Create_Attr(string tablename,string Attr_Name,int Attr_Type,bool unique);`

   用于添加表。

4. `int tableCount();`

用于返回表的总数。

5. `int attrCount(string table);`

   用于返回字段的总数。

6. `bool Drop_Table(string table);`

用于删除表。

7. `Table get_table_info(string table);`

用于获取表信息。

8. `attrbute get_attr_info(string table, string attr);`

用于获取列信息。

9. `bool isUnique(string table, string attr);`

用于检查某表某字段是否唯一（字段的属性是否是unique）

10. `bool hasIndex(string table, string attr);`

用于检查某表某字段是否有索引。

11. `bool attrExists(string table,string attr);`

用于属性是否存在。

12. `bool indexExists(string index);`

    用于检查索引是否存在。

13. `Index* indexName(string table, string attr);`

    用于返回某字段上所有的索引，返回类型尚待商榷。

14. `bool isPK(string table, string attr);`

    用于检查某表是否是主键。

15. `string pkOnTable(string table);`

    用于寻找表中主键 。

16. `Index get_index_info(string index);`

    用于获取某索引信息，返回类型尚待商榷。

17. `map<string,Index> get_all_index();`

    用于获取所有索引信息。

18. `int indexCount(string table);`

    用于返回索引的总数。

19. `string tableIndexOn(string index);`

    用于获取索引建立在哪个表。

20. `string attrIndexOn(string index);`

    用于获取索引建立在哪个字段。

21. `bool addIndex(string index,string table, string attr);`

    用于添加索引。

22. `bool deleteIndex(string index);`

    用于删除索引。

23. `bool save();`

    用于保存现有内存中的catalog信息

24. `bool setpk(string table,string attr);`

    用于设置primary key。

25. `void Insert_Data(string tablename);`

    用于增加对应record的记录。