API 模块设计报告 

> by 李易非

一、初步介绍

API是程序架构的中间层，API接受 Interpreter 解释完成、初步检测语法的命令，调用 CatalogManager 检验一致性、确定命令的具体执行规则；调用 RecordManager 来进行记录文件的更新；调用 IndexManager 进行索引文件的更新；在本次实验中，我们的API 类有如下几个函数。

```
class API
{
  public:
    bool create_table(表名, 属性);
    bool drop_table(表名);
    bool insert(表名, 插入数据, 插入数据类型);
    int Delete(表名, 属性名, 条件, 操作数);
    int select(表名, 属性名, 条件, 操作数);
    bool create_index(表名, 属性名, 索引名);
    bool drop_index(索引名, 表名);
};
```

二、具体实现

Create Table

- 调用 CatalogManager 的 Create Table 函数；
- 调用 RecordManager 的 Create Table 函数；
- 遍历 Interpreter 提供的属性，如果是 Primary / Unique，调用 IndexManager 建立索引；

Drop Table

- 调用 CatalogManager 获得所有的索引，通过获得的索引，调用 IndexManager 的 Drop Index 函数；
- 调用 CatalogManager 的 Drop Table 函数；
- 调用 RecordManager 的 Drop Table 函数；

Insert

- 调用 CatalogManager 获得表信息，通过表信息来判断插入数据的格式是否合法，同时把插入数据 (string) 为对应类型的 rawdata (char 数组) ；
- 判断各个属性的性质

    - 若为 Unique 且 Indexed —> 调用 IndexManager 的 Find 函数进行一致性检验；
    - 若为 Unique 但 UnIndexed —> 遍历数据文件进行一致性检验；
    - 若非 Unique —> 直接调用 RecordManager 进行插入；
- 一致性检验完毕后，分别调用 RecordManger / IndexManager 进行数据文件 / 索引文件的更新；

Delete

- 调用 CatalogManager 获得表信息，通过表信息来属性名是否存在，表是否存在；
- 对 `=` 条件检测是否存在索引，如果存在索引，调用 IndexManager 查找到记录位置，再调用 RecordManager 删除数据文件中的记录；
- 对于其它条件，直接利用 RecordManager 在文件中进行遍历搜索来删除记录；

Select

- 调用 CatalogManager 获得表信息，通过表信息来属性名是否存在，表是否存在；
- 对 `=` 条件检测是否存在索引，如果存在索引，调用 IndexManager 查找到记录位置，再调用 RecordManager 找到数据文件中的记录，在通过 CatalogManager 检验找到的记录是否符合所有的条件；
- 对于其它条件，直接利用 RecordManger 来查找记录，同时调用 CatalogManager 来检验找到的记录是否符合所有的条件；
- 找到符合条件的记录后，即时输出；

Create Index

- 调用 Catalog Manager 来判断表名是否存在，同时为索引提供对应属性的数据类型、以及在一条记录中的相对位置；
- 调用 Catalog Manager 的 Create Index 函数，判断属性是否 Unique，属性是否存在，更新 Metadata;
- 调用 Index Manager 的 Create Index 函数，建立一个新的 B+ 树；

Drop Index

- 调用 Catalog Manager 的 Drop Index 函数，判断是否存在这个索引，更新 Metadata;
- 调用 Index Manager 的 Drop Index 函数，删除对应索引文件；

三、 技术细节

错误管理

本次工程中，共建 `Error` 类来进行错误管理，所有底层函数都利用 **throw** 机制抛出错误，

在 API 层集中进行 **try catch** 。以 Drop Index 为例，API 要做的事很简单，调用对应的函数并处理错误即可。

```
bool API::drop_index(const string &index_name, const string &table_name)
{
    try
    {
```

```
        CatalogManager &catalogmanager = MiniSQL::get_catalog_manager();
        IndexManager &indexmanager = MiniSQL::get_index_manager();

        catalogmanager.drop_index(index_name, table_name);
        indexmanager.dropIndex(index_name);
        return true;
    }
    catch (Error err)
    {
        err.print_error();
        return false;
    }

    return true;
}
```