minisql excutor 个人报告

3200105515 颜晗

本次数据库project中，本人负责实现数据库执行器部分。基本思想即遍历得到的语法树，获取必要的信息和数据，根据信息数据获取对应数据库与表，再从底层到顶层依次构建变量并最终使用下面模块提供的函数进行插入删除与更新。下面将依次对各功能进行讲解，代码部分因为并没有什么特色，而且比较冗余庞大，此处不进行展示。

一、功能分析

1. **语法树**

语法树最高层用于判断操作类型；向下一层用于分析各种条件，例如标记使用数据库（表）的名字的节点，标记需要显示属性名的节点，标记判断条件；再向下一层为显示属性（全部的话在上一层即可判断）和条件（连接符和比较操作符）的具体值。整体按语序进行排列，我们按照形状遍历即可。

2. **构造函数与退出**

执行器其实就是多个数据库引擎的一个`map`容器，所以其构造就是再次调用数据库的构造函数，重点在于修改退出机制，使其在退出时将所有数据库的名称存储于一个文件中，然后就可以在构造中利用名字重新构造数据库，注意重新构造时数据库`init`参数为`false`，因为不能清空原数据。

3. **显示功能与使用数据库**

包括数据库、表、索引在内的所有展示功能都是遍历存容器并去除成员的名字。数据库即遍历执行器内`map`容器，表需要找到当前使用的数据库再遍历，索引由于语法树无法给参数，本人采用遍历并输出数据库所有表对应索引的形式进行输出。

执行器本身有标记当前数据库的字符串变量，使用数据库即遍历所有数据库名，如果有匹配的即数据库存在，将当前数据库字符串修改即可。另外，其余函数在最开始都有判断数据库字符串是否为空（是否有`use database`），防止非法访问出现内存错误，包括各种存在的判断，下面的介绍都略过不谈。

4. **创建与删除**

主要分为数据库、表、索引的创建与删除。

数据库的创建与删除已经有给定的构造与析构函数，只要给定名字即可，注意创建新的数据库与从已有文件恢复数据库引擎的`init`参数是不同的即可。`true`时会导致删除本有的数据库文件建立新的空数据库，`false`会在原有文件的基础上重新打开数据库。

表的创建相对复杂点。得到表的名字后，需要遍历类型节点及其所有分支得到属性的名字和类型，通过属性名与类型先构造`Column`变量存入`vector`容器，再通过该容器构造一个`Schema`变量，声明一个空的表信息指针（`TableInfo*`），即可调用下层的函数进行表的建立。最后有的表声明了主键，而且由于语序排布，主键的声明都在最后，我们需要先拿到所有的声明，还要根据主键来给一些表的列`unique`的属性才能开始属性的构建（因为沿用现成的函数，很少进行增加和修改，部分类的成员一旦构造无法修改），包括得到属性后还需要创建主键的索引，该操作在下面索引部分详细介绍步骤。表的删除只需要获取表的名称即可调用下层的函数进行删除了。

索引建立是表的简化版，拿到表的名字和索引名字，再拿到建立索引的列名，判断`unique`后构建索引的`Schema`(和表的`Schema`是同一类)，调用底层函数即可实现建立。删除同样是通过名字调用底层函数即可，只是由于没有表的名字，需要遍历数据库的所有表。

5. **插入**

插入主要是`Field`以及`Row`的数据获取和构造，调用下层函数即可将`Row`插入对应表中，至于写入硬盘是底层维护的事。另外，我们需要针对`primary key`以及`unique`进行查重，具体方法为通过构造值进行索引的查找，查到了即插入失败，不能有重复值，未查到即可进行插入。最后插入表成功后还需要将对应值插入索引进行维护。

6. **选择、删除、更新**

这三种操作的主要部分都是条件的获取与判断，当然，选择额外加入了使用索引的判断与搜索操作。

选择需要先获取显示的列，后面对表的`Schema`进行遍历对应输出。更新需要获取更新的属性和值，用于后面调用函数。

而对于条件的使用，我们构造了一个新类`cmpData`包括了`Field`类以及一个`string`成员存储对应的比较操作符还有一个用于标记对应值需要比较的列在表中所处位置的"索引"（此索引非上面的索引）成员，而且实现了额外函数对于一组条件三个节点（操作符、属性名、属性值）进行构造变量。另外对于连接符导致的多种条件，我们使用一个`vector`容器来存储一组条件（即使用`and`进行连接，需要同时满足），再使用`vector<vector>`来存储多组（即使用`or`连接）条件，遍历时满足一组条件即可。由于语法树按语序进行构造，多组条件的实现比较麻烦，目前只实现了单个条件或连接符的操作。

具体查找过程，删除和更新都是通过迭代器对表进行遍历一一比较。而查找除此之外还有索引的使用，而由于框架的限制，目前只支持单个条件的相等查找，经过验证可以看出索引的查找效率确实远高于遍历。而对于索引的查找需要构造索引包含的属性对应`Row`然后调用函数进行查找，如果传入的`vectror`中存了有效的`RowId`说明查到了，接下来再利用`RowId`对表进行查找即可大大提升速度。

7. **执行文件**

通过对主函数的过程进行复制，我们便实现了文件执行，只要使用C++的`stream`便可较好复刻过程。只是注意判断文件的结束。

二、个人感想

`minisql`的实现就此结束，从一开始无从下手到最终实现了基本功能，感觉自己还是收获了很多，虽然没有亲自实现底层，但是由于执行器的实现离不开底层各种类与函数，不得不将整个框架与代码看了几遍。另外，`minisql`的复杂对于代码能力的提升也非常明显，对C++的各种功能特性的使用能力都得到了锻炼。

当然目前整个框架仍旧存在一些问题，包括部分提供的代码注释不足，容易导致学生理解偏差进而实现与理想有差距导致出现难以理解的错误，另外，代码中也有一些错漏？比如比较让人难以理解的为什么`TypeChar`类型有`GetData()`函数而另外两个没有实现，而且作为继承却能调用基类的……这种框架本身的代码一般都是通过`.h`文件看功能调用，一旦出现问题比较难查出。总之还是希望未来框架的代码和注释文档能够逐渐完善，减少学生无意义`debug`的过程，毕竟数据库本身也不是主要锻炼代码能力的课程。