实验报告—实验2：执行引擎（上）

学号	姓名	邮箱	完成题目
221220067	刘思远	221220067@smail.nju.edu.cn	t1

1 完成情况

完成了`t1`，没有完成`f1`。

同实验1，在考察了了总评分数占比组成和实验2的各子题目占比组成后，发现 `f1` 即使不做，对总评的影响也可以忽略不计；做的话可能需要投入大量的时间，且功利性的回报和付出看起来并不成正比。综上，选择放弃`f1`。

2 实验详述

下面的叙述只考虑`t1`模块。总的实验结果截图如下：

```
[17:27:47]LOG <ClientHandler::154>: Client 5 connected
[17:27:47]LOG <ClientHandler::161>: Client 5 sent: open database db2024;
[17:27:47]LOG <ClientHandler::161>: Client 5 sent: drop table dbcourse;
[17:27:47]LOG <ClientHandler::161>: Client 5 sent: exit;
Passed: 05_cleanup
Passed [5/5] tests
```

2.1 前情提要与准备工作

首先，需要对火山模型的执行过程有一个详细的了解。所谓的火山调用，个人理解下，概括起来说，就是上层调用下层，下层返回上层结果。每层之间功能相互独立，只需下层为上层提供调用接口即可。

其次，需要搞清楚`execution/executor.cpp`中的`Execute`函数的执行逻辑。对于DDL语句或者DML语句，会在进入判断是否end的while循环前先调用一次next，则需要保证在next函数里面正确设置了是否结束的相关条件；对于其他的基本算子，由于在for循环开始会进行一步初始化操作，即`executor->Init()`，之后在循环体内部就直接`GetRecord`了，所以需要保证在`Init()`方法里面已经正确设置了`record_`变量的初始值。

2.2 insert

2.2.1 思路

insert操作要求需要同时在表格和索引中加入元组。则要遍历待加入的记录`inserts_`，对于每一条记录，都要既通过tablehandle插入，也要通过索引插入。索引插入时需要遍历`indexes_`，确保每个都插入。每插入一条记录，都要使`count`递增。

2.2.2 实验结果

```
45.95146840720358, 68.509997294541);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: insert into dbcourse values (763, 'Su', 16, 'Zhenjiang', 2.702
51, 52.15618957355295, 84.4692809013063);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: insert into dbcourse values (147, 'Fiona', 46, 'Langfang', 0.7
4711, 42.52482063110601, 88.11024395794989);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: insert into dbcourse values (835, 'Rex', 24, 'Taicang', 1.3545
7, 57.21361047950643, 69.74002556609759);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: insert into dbcourse values (282, 'Nina', 23, 'Qingyang', 3.89
521, 92.47702117430737, 96.436834252589);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: insert into dbcourse values (463, 'Yvonne', 12, 'Zhenjiang', 4
.99760, 40.74406573884635, 82.35316566604423);
[17:27:45]LOG <ClientHandler::161>: Client 5 sent: exit;
[17:27:45]LOG <Close::105>: close table
Passed: 01_prepare_table_dbcourse
```

2.3 seqscan_limit_projection

2.3.1 思路

Seqscan：init函数中，需要设置rid_为第一个rid的值，record_设置为对应的记录（调用tablehandle的GetRecord）函数来实现；Next函数中，每一次next都要获取下一条记录，则需要GetNextRid函数的支持，并把record_设置为对应记录；IsEnd函数中，通过rid_是否等于INVALID_RID来判断是否结束。

Limit：在next函数中，每调用一次child_的next函数，都需要递增count_。在判断是否结束时，首先判断是否到达limit_的数值，如果没有到，再看child_是否遍历结束。

Projection：init函数中，即需要使用`record_ = std::make_unique<Record>(out_schema_.get(), *child_record);`来生成符合投影要求的记录。

2.3.2 实验结果

```
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: select l2_score, address from dbcourse limit 1159;
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: select l1_score, l2_score, address, gpa, id from dbcourse ;
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: select id, gpa from dbcourse limit 502;
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: select id, l1_score, age from dbcourse ;
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: select gpa, address, id, name, l1_score from dbcourse limit 85
6;
[17:27:45]LOG <ClientHandler::161>: Client 4 sent: exit;
[17:27:45]LOG <Close::105>: close table
Passed: 02_seqscan_limit_projection
```

2.4 filter_update_delete

2.4.1 思路

Filter：init中，即需要按照next函数的主体部分得到第一条记录，即反复调用child的next函数，直到找到一条符合filter条件的记录，将其设置为record_。如果没有找到这样的记录，则说明在scan完成之后，也没有找到符合要求的记录，把record_设置为nullptr即可。

Update：需要实现的是next函数。首先遍历child的next_，对于每一个记录，首先得到它的recordschema，接着新开一个vector存放更新后的这条记录的值。遍历每一个field，如果发现在需要更新的field的list里面，则把update后的新值push到vector里面，否则把old_reord的旧值push进去。在完成new_values的构建之后，重新构建一个new_record，然后分别在表格和索引中进行更新。

Delete：和insert的思路一致，只不过需要把insertrecord函数的调用替换为deleterecord函数的调用。

2.4.2 实验结果

```
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: delete from dbcourse where address='Nanjing';
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: select * from dbcourse where  address='Nanjing';
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: select * from dbcourse where  address='Beijing';
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: delete from dbcourse where address='Beijing';
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: select * from dbcourse where  address='Beijing';
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: select * from dbcourse;
[17:27:46]LOG <ClientHandler::161>: Client 5 sent: exit;
[17:27:46]LOG <Close::105>: close table
Passed: 03_filter_update_delete
```

2.5 sort_final

2.5.1 思路

init中，需要先调用sortbuffer进行一个排序，之后调用next函数设置第一条获取的记录。

next中，需要分类讨论，如果`buf_idx_ < sort_buffer_.size()`，则需要把record设置为buffer中buf_idx指定位置的record；反之，则把record_设置为nullptr。

end函数中，判断是否结束的标志为`buf_idx_ > sort_buffer_.size() || record_ == nullptr`。

最为重要的是SortBuffer函数。首先开一个vector，来把child_能够获取到的record都填进去。接着调用标准库函数std::sort对vector中的函数进行排序。由于在init函数中会调用SortBuffer函数，则需要再SortBuffer中完成buf_idx的初始化操作，即把其设置为0。

2.5.2 实验结果

```
[17:27:46]LOG <ClientHandler::161>: Client 4 sent: select * from dbcourse where age > 20 order by desc age, name,
 address limit 10;
[17:27:46]LOG <ClientHandler::161>: Client 4 sent: select gpa, name, address from dbcourse where age > 20 order b
y age, name, gpa limit 10;
[17:27:47]LOG <ClientHandler::161>: Client 4 sent: select age, name, id from dbcourse where age > 20 and id <> ag
e order by desc age, name limit 10;
[17:27:47]LOG <ClientHandler::161>: Client 4 sent: exit;
[17:27:47]LOG <Close::105>: close table
Passed: 04_sort_final
```

3 实验感想

1. 心态曲曲折折起起伏伏。由于听同学说实验2的手册基本无提示，代码也无注释，最初对实验2有种畏难和抵触情绪。后来听隔壁班的同学说他们班实验2尽力而为就好，对最后总评的影响微乎其微，于是便不打算做了。再后来隔壁班突然说还是要写的，于是那就写吧！可惜呀，写完insert、projection和seqscan去进行select操作时，最开始还能出来1000条记录，两天之后，在印象中未曾更改一点儿代码的前提下，发现只能出来660条了。接着，便是漫长的debug之路。在尝试了能想到的所有方法之后，还是不行。在跨年夜晚上，决定把lab1的部分从教学立方下载下来，替换掉现在的lab1，万一中间不小心改了框架代码呢。山重水复疑无路，柳暗花明又一村。替换之后，在不改变lab2实现的前提下，顺利通过了！
2. 或许之后的实验手册可以在介绍背景知识外，适当的加一些思路上的提示，不然有的函数真的只能靠chatGPT或者deepseek等大神了（当然也可能是自己水平不够）。
3. 第一次在跨年夜坚持写完lab，虽然筋疲力尽，但也有所收获！
4. 希望10号的期末考试顺利！