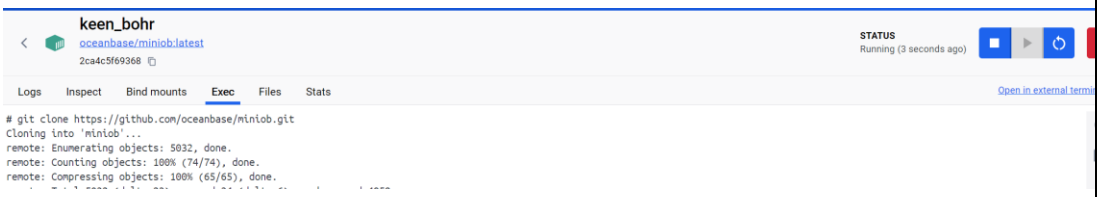
《数据库系统原理》实验报告（5）					
题目：miniOB 入门实验					
学号		姓名		日期	2024.4.23

实验环境：



实验步骤及结果截图：

1. 在 Docker 中建立 miniob 环境（参考第 0 章的教程 PPT 和第三节的内容）。

```
# git clone https://github.com/oceanbase/miniob.git
Cloning into 'miniob'...
remote: Enumerating objects: 5032, done.
remote: Counting objects: 100% (74/74), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 5032 (delta 23), reused 34 (delta 6), pack-reused 4958
Receiving objects: 100% (5032/5032), 27.42 MiB | 1.18 MiB/s, done.
Resolving deltas: 100% (3129/3129), done.
# cd miniob
# ls
benchmark  build.sh  cmake  CMakeLists.txt  CODE_OF_CONDUCT.md  CONTRIBUTING.m
# bash build.sh --make -j4
build.sh --make -j4
create soft link for build_debug, linked by directory named build
[ 99%] Built target ring_buffer_test
[100%] Linking CXX executable ../bin/thread_pool_executor_test
[100%] Built target simple_queue_test
[100%] Built target thread_pool_executor_test
# cd build
# ./bin/observer -s miniob.sock -f ../etc/observer.ini &
#
Welcome to the OceanBase database implementation course.

Copyright (c) 2021 OceanBase and/or its affiliates.

Learn more about OceanBase at https://github.com/oceanbase/oceanbase
Learn more about MiniOB at https://github.com/oceanbase/miniob

Successfully load ../etc/observer.ini

# ./bin/obclient -s miniob.sock

Welcome to the OceanBase database implementation course.
```

```
miniob > help
Commands
show tables;
desc `table name`;
create table `table name` (`column name` `column type`, ...);
create index `index name` on `table` (`column`);
insert into `table` values(`value1`,`value2`);
update `table` set column=value [where `column`=`value`];
delete from `table` [where `column`=`value`];
select [ * | `columns` ] from `table`;
miniob >
```

2. 创建一张表，包括学号，姓名，身高，体重。

表名：Student		
字段	**说明**	**类型**
No	学号	int
Name	姓名	char(10)
Height	身高	float
Weight	体重	float

```
miniob > create table student(no int,name char(10),height float,weight float)
SUCCESS
```

```
miniob > desc student;
Field | Type | Length
no | ints | 4
name | chars | 10
height | floats | 4
weight | floats | 4
```

3. 向该表插入下列数据。

No	Name	Height	Weight
2353123	王海峰	180.5	75.5
2331902	刘伟	174.9	63.3
2353074	孙国程	169.0	82.5
2233444	严磊	178.3	52.8

```
miniob > insert into student values (2353123,'王海峰',180.5,75.5);
SUCCESS
miniob > insert into student values (2331902,'刘伟',174.9,63.3);
SUCCESS
miniob > insert into student values (2353074,'孙国程',169.0,82.5);
SUCCESS
miniob > insert into student values (2233444,'严磊',178.3,52.8);
SUCCESS
```

```
miniob > select * from student;
no | name | height | weight
2353123 | 王海峰 | 180.5 | 75.5
2331902 | 刘伟 | 174.9 | 63.3
2353074 | 孙国程 | 169 | 82.5
2233444 | 严磊 | 178.3 | 52.8
```

4. 使用 select 语句展示学号，姓名，身高。

```
miniob > select no,name,height from student;
no | name | height
2353123 | 王海峰 | 180.5
2331902 | 刘伟 | 174.9
2353074 | 孙国程 | 169
2233444 | 严磊 | 178.3
```

5. 尝试修改指定行的体重如下表所示，能否成功？为什么？

No	Name	Height	Weight
2353123	王海峰	180.5	78.5
2331902	刘伟	174.9	60.0

```
miniob > update student set weight=78.5  where no=2353123;
SUCCESS
miniob > update student set weight=60.0  where no=2331902;
SUCCESS
miniob > select * from student;
no | name | height | weight
2353123 | 王海峰 | 180.5 | 75.5
2331902 | 刘伟 | 174.9 | 63.3
2353074 | 孙国程 | 169 | 82.5
2233444 | 严磊 | 178.3 | 52.8
```

没有成功

理由：因为 miniOB 数据库中并没有实现修改表中数据的功能。

6. 删除孙国程和严磊的记录。

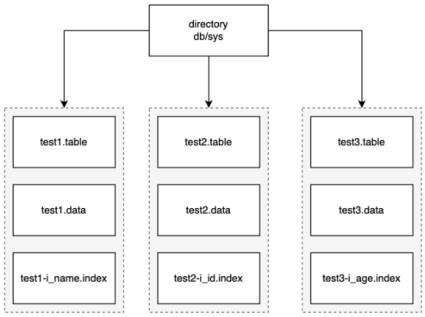
2353074	孙国程	169.0	82.5
2233444	严磊	178.3	52.8

```
miniob > delete from student where no=2353074;
SUCCESS
miniob > delete from student where no=2233444;
SUCCESS
miniob > select * from student;
no | name | height | weight
2353123 | 王海峰 | 180.5 | 75.5
2331902 | 刘伟 | 174.9 | 63.3
```

7. 对 miniob 源码进行阅读，主要选取一个功能（如 create table、 insert、delete 等）进行分析理解， 做简要报告（不超过两页）。

选取的功能是 create table

MiniOB 启动后会默认创建一个 sys 数据库，所有的操作都默认在 sys 中。



一个数据库下会有多张表。上图示例中只有三张表，接下来以 test1 表为例介绍一下表里都存放什么内容。

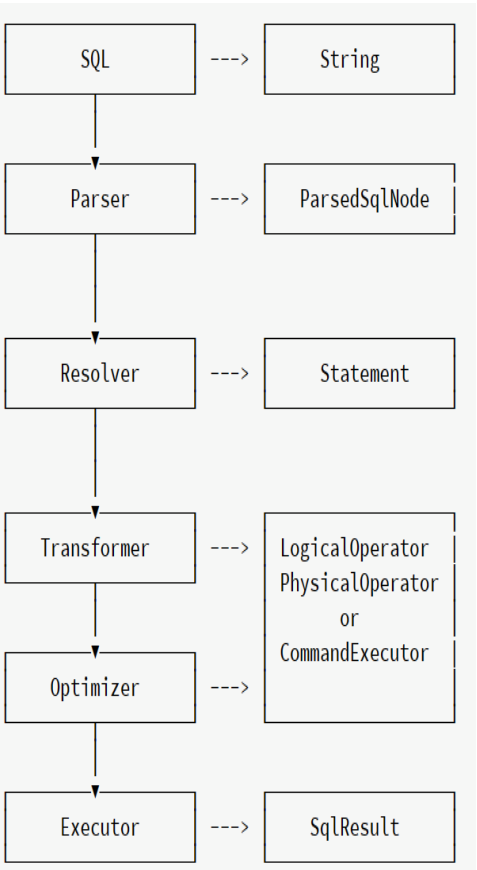
test1.table：元数据文件，这里面存放了一些元数据。如：表名、数据的索引、字段类型、类型长度等。

test1.data：数据文件，真正记录存放的文件。

test1-i_name.index：索引文件，索引文件有很多个，这里只展示一个示例。

MiniOB 的 SQL 语句执行流程如下图所示：

左侧是执行流程节点，右侧是各个执行节点输出的数据结构。



1. 我们收到了一个 SQL 请求，此请求以字符串形式存储;
2. 在 Parser 阶段将 SQL 字符串，通过词法解析(lex_sql.l)与语法解析(yacc_sql.y)解析成 ParsedSqlNode(parse_defs.h);
3. 在 Resolver 阶段，将 ParsedSqlNode 转换成 Stmt(全称 Statement， 参考 stmt.h);
4. 在 Transformer 和 Optimizer 阶段，将 Stmt 转换成 LogicalOperator，优化后输出 PhysicalOperator(参考 optimize_stage.cpp)。如果是命令执行类型的 SQL 请求，会创建对应的 CommandExecutor(参考 command_executor.cpp);
5. 最终执行阶段 Executor，工作量比较少，将 PhysicalOperator(物理执行计划)转换为 SqlResult(执行结果)，或者将 CommandExecutor 执行后通过 SqlResult 输出结果。

SQL 解析分为词法分析与语法分析。

（1）词法分析文件 lex_sql.l 中：

```
[\-]?{DIGIT}+  yylval->number=atoi(yytext); RETURN_TOKEN(NUMBER);
HELP           RETURN_TOKEN(HELP);
DESC           RETURN_TOKEN(DESC);
CREATE         RETURN_TOKEN(CREATE);
DROP           RETURN_TOKEN(DROP);
```

每一行都是一个模式，左边是模式，使用正则表达式编写，右边是我们返回的 token，这里的 token 是枚举类型，是我们在 yacc_sql.y 中定义的。

（2）语法分析文件 yacc_sql.y 中：

```
create_table_stmt:    /*create table 语句的语法解析树*/
    CREATE TABLE ID LBRACE attr_def attr_def_list RBRACE
    {
      $$ = new ParsedSqlNode(SCF_CREATE_TABLE);
      CreateTableSqlNode &create_table = $$->create_table;
      create_table.relation_name = $3;
      free($3);
      std::vector<AttrInfoSqlNode> *src_attrs = $6;
      if (src_attrs != nullptr) {
        create_table.attr_infos.swap(*src_attrs);
      }
      create_table.attr_infos.emplace_back(*$5);
      std::reverse(create_table.attr_infos.begin(), create_table.attr_infos.end());
      delete $5;
    }
    ;
```

每个规则描述，比如 create_table_stmt 都会生成一个结果，这个结果在.y 中以 "$$" 表示，某个语法中描述的各个 token，按照顺序可以使用 "$1 $2 $3" 来引用，比如 ID 就是 $3，attr_def 是 $5。"$n" 的类型都是 YYSTYPE。YYSTYPE 是 bison 根据.y 生成的类型，对应我们的规则文件就是 %union，YYSTYPE 也是一个 union 结构。比如 我们在.y 文件中说明 %type <sql_node> create_table_stmt，表示 create_table_stmt 的类型对应了 %union 中的成员变量 sql_node。我们在 %union 中定义了 ParsedSqlNode * sql_node;，那么 create_table_stmt 的类型就是 ParsedSqlNode *，对应了 YYSTYPE.sql_node。

%union 中定义的数据类型，除了简单类型，大部分是在 parse_defs.h 中定义的，表达式 Expression 是在 expression.h 中定义的，Value 是在 value.h 中定义的。

由于在定义语法规则时，这里都使用了左递归，用户输入的第一个元素会放到最前面，因此在计算得出最后的结果时，我们需要将列表（这里很多使用 vector 记录）中的元素逆转一下。

出现的问题：

1. 创建 **tabel** 时总是报错 SQL_SYNTAX > Failed to parse sql

解决方案：

1. 与 **oceanbase** 相比，去掉 **primary key** 约束就成功了