数据库大作业第二阶段实验报告

已完成的功能

`基础部分`

全部完成。已在OJ上提交通过。

`拓展部分`

实现了除存档功能外的全部功能

设计思路见说明文档，以下为所有功能的样例(包含了某些附加特色的展示)

1. 单函数

   字符串系列（4个）

   `int char_length(char)` 返回字符串长度

   `char lower(char)` 返回字符串的小写形式

   `char upper(char)` 返回字符串的大写形式

   `char space(int)` 返回整数个空白字符

   数字函数（5个）

   `num` 代指 `int/double`

   `num abs(num)` 返回数的绝对值

   `double sin(num)` 返回数的正弦值

   `double cos(num)` 返回数的余弦值

   `double tan(num)` 返回数的正切值

   `double exp(num)` 返回以e为底的幂，e为自然对数的底

   日期函数（2个）

   `date adddate(date,int)` 日期加上天数

   `time addtime(time,int)` 时间加上秒数

   测试文件：single_functions.sql

   测试结果：同名文件single_functions.out。下同。

   关键测试代码：

```sql
SELECT abs(grade) from oop_info;
SELECT char_length(stu_name) from oop_info;
SELECT lower(stu_name) from oop_info;
SELECT upper(stu_name) from oop_info;
SELECT space(1) from oop_info;
SELECT space(2) from oop_info;
SELECT abs(grade) from oop_info;
SELECT MAX(grade) from oop_info GROUP BY stu_name;
SELECT MIN(grade) from oop_info;
SELECT MIN(grade) from oop_info GROUP BY stu_name;
SELECT sin(angle) from triangle;
SELECT tan(angle) from triangle;
```

```
SELECT cos(angle) from triangle;
SELECT exp(val) from triangle;
```

2. 聚合函数

`double avg(num)` 返回字段的平均值

`int count(expr)` 返回不为空的记录条数

`auto max(expr)` 返回字段的最大值

`auto min(expr)` 返回字段的最小值

测试文件：aggregate_functions.sql

关键测试代码：

```
SELECT stu_name, COUNT(*) from oop_info GROUP BY stu_name ORDER BY COUNT(*);
SELECT stu_name, COUNT(*) from oop_info GROUP BY stu_name ORDER BY MAX(grade);
SELECT stu_name, COUNT(*) from oop_info GROUP BY stu_name ORDER BY MIN(grade);
SELECT stu_name, COUNT(*) from oop_info GROUP BY stu_name ORDER BY COUNT(grade);
SELECT stu_name, AVG(grade) from oop_info GROUP BY stu_name ORDER BY AVG(grade);
```

3. 日期类

引入两种新类型 `date` 和 `time`

`date` 的范围是 `1000-01-01` ~ `9999-12-31`

`time` 的范围是 `-838:59:59` ~ `838:59:59`

与之关联的为两个日期函数 `adddate` 和 `addtime`

测试文件：date.sql

关键测试代码：

```
create table date_info(id int not null, d date, primary key(id));
insert into date_info(id,d) values(1,"2000-10-19");
select * from date_info where id<2 xor d>"2000-11-19";
select * from date_info where id<2 xor d>"2000-10-18";
```

4. 复杂表达式

支持的运算符、逻辑符

`|| or xor`

`&& and`

`not !`（只表示逻辑非）

`= > < != LIKE`

`+ -`

`* / DIV % MOD`

所有的运算符、逻辑符、函数共同组成了复杂表达式，支持括号改变优先级、嵌套等表达式结构

复杂表达式可在select、order、group、where子句中使用

select子句中可用as语法结构指定别名

测试文件：complex_expressions.sql

关键测试代码：

```
   select (ABS(-1) - 5) /2 as AaA, (sin( 1) - cos (2))/exp(1) as
BBb,char_length("  aaa") + sin(Abs (-1)) as C, adddate("2000-10-19", 0), 3
Mod 0, 2 div 5;
   select char_length(stu_name) from oop_info where stu_id>
(ABS(-1)+ABS(grade)) and d != ADDDATE("2000-1-1", 100) or 3*2-5 > (3-2) /2;
   SELECT grade*grade,1 - COUNt(*), Avg(grade) as avggrade, min(stu_id) from
oop_info GROUP BY grade*grade ORDER BY COUNT(*),-min(stu_id) where
char_length(stu_name)+grade = 4;
```

5. 多表连接

   测试文件：join.sql

   关键测试代码：

```
SELECT * FROM test1 t1 LEFT JOIN  test2 t2 on t1.id=t2.id;
SELECT * FROM test1 t1 right JOIN  test2 t2 on t1.id=t2.id;
SELECT * FROM test1 t1  inner join  test2 t2 on t1.id=t2.id;
```

6. 模糊匹配

   测试文件：like.sql

   关键测试代码：

```
SELECT * from oop_info WHERE stu_name LIKE 'ja%';
SELECT * from oop_info WHERE stu_name LIKE '%ki%';
SELECT * from oop_info WHERE stu_name LIKE '%am';
SELECT * from oop_info WHERE stu_name LIKE 'jack%rose';
SELECT * from oop_info WHERE stu_name LIKE '%leave%';
```

7. 联合查询

   测试文件：multitables.sql

   关键测试代码：

```
select t1.name,t2.name from t1,t2 where t1.id = t2.id;
```

8. union语句

   测试文件：union.sql

   关键测试代码：

```
select * from test1 union select * from test2 order by name;
select * from test1 union all select * from test2 order by id;
```

9. 1 server + n clients

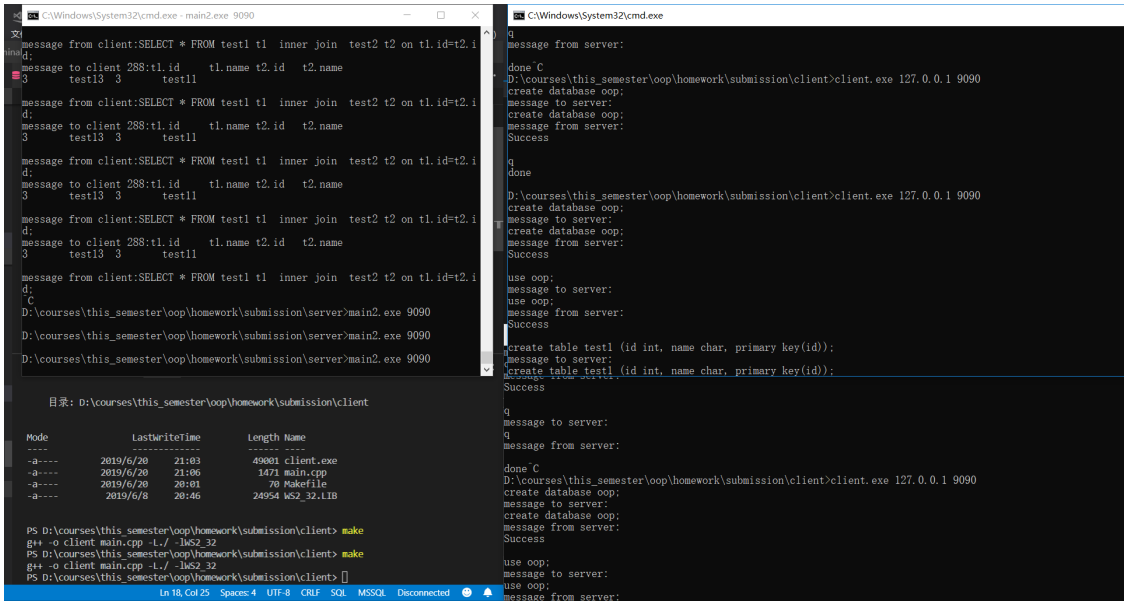
   服务端主函数：server/main.cpp

   启动方法：在server目录下make，然后main2.exe [port]

   客户端主函数：client/main.cpp

启动方法：在client目录下make，然后client.exe [服务端ip] [服务端port]。可以通过文件重定向或手动输入的方式向服务端发送请求。

测试结果：



测试方法

服务端只能接受来自客户端的输入，不能自定义测试。如果需要批量测试，可以在main文件夹下make生成main.exe，该可执行程序保留了除了充当服务器之外所有的数据库功能。

然后再进入test文件夹，运行

```
bash test.sh
```

即可执行以上全部测试文件。

优劣分析

亮点

1.使用智能指针对带有表中数据的类进行包装，在节省资源的同时避免了人工手动维护的代价过高

2.使用命名空间存储字符串操作库、运算符、单函数和聚合函数，划清类之间的界限，分类清晰便于拓展和调用

3.对select语句的处理兼具灵活性和对复杂表达式的支持，使用操作符栈、数据栈和虚拟表的思想建立统一的处理范式，使用较为简洁和系统的代码广泛实现，重用性高

不足

1.距真正的mysql仍缺少更加灵活的语法方式，如(select) union (select) 这样在代码块上加入括号的句法等

2.对某些语法现象仍需特判，如count(*)中为了避免*和运算符的混淆将其预替换为countall并在执行阶段与其余聚合函数剥离开单独处理，这样导致all成为保留关键字无法作为字段名和别名使用