OceanBase初赛题目介绍

背景

为了让更多的同学能够通过OceanBase数据库大赛学习到数据库实战知识，我们在MiniOB的基础上设计了一系列从浅入深的题目。这些题目的入门门槛较低，适合所有参赛选手。 面向的对象主要是在校学生，数据库爱好者，或者对基础技术有兴趣的爱好者。MiniOB 对诸多模块做了简化，比如不考虑并发操作，事务比较简单。 初赛的目标是让不熟悉数据库设计和实现的同学能够快速的了解与深入学习数据库内核，期望通过miniob相关训练之后，能够对各个数据库内核模块的功能与它们之间的关联有所了解，并能够在使用数据库时，设计出高效的SQL，并帮助降低学习 OceanBase 内核的门槛。

题目介绍

初赛题目按照难度排序，列表中靠后的题目可能依赖前面题目实现的功能，题目按照难度和工作量计分。

计分规则：每实现一个题并通过对应的测试，就获得对应的分数，一个题要么通过要么不通过。

作为练习，前面的题目在单个实现时，是比较简单的。而后面的题目与前面功能的互动会更多，希望你能够在实现某个功能时，考虑一下多个功能之间的关联。比如实现了多字段索引，可以考虑下多字段索引的唯一索引、根据索引查询数据等功能。另外，可以给自己提高一点难度。比如在实现表查询时，可以假设内存中无法容纳这么多数据，那么如何创建临时文件，以及如何分批次发送结果到客户端。

题目列表

名称	分值	描述	测试用例示例	前置题目依赖
查询元数据校验 select-meta	10	1. 查询语句中存在不存在的列名、表名等，需要返回失败。 2. 需要检查代码，需要返回错误的地方要返回错误。	create table t(id int, age int); select * from t where name='a'; select address from t where id=1; select * from t_1000; select * from t where not_exists_col=1; select *, address from t;	

删除表格
drop–table
	10	1. 删除表，清除表相关的资源。
		2. 要删除所有与表关联的数据，不仅包括磁盘中的文件，还包括内存中的索引等数据。
			create table t(id int, age int);
			create table t(id int, name char);
			drop table t;
			create table t(id int, name char);	

实现更新功能 update	10	1. update单个字段即可。 2. 可以参考insert_record和delete_record的实现。 3. 目前仅能支持单字段update的语法解析，但是不能执行。 4. 需要考虑带条件查询的更新，和不带条件的更新。	update t set age =100 where id=2; update t set age=20;	
基本类型的隐式转换 typecast	20	1. 字符串与整数，以及整数与浮点数的类型转换。 2. 字符串与整数比较时会转成整数类型，字符串/整数与浮点数比较时会转成浮点数类型，具体可查阅MySQL隐式转换规则以及常见问题说明。	create table t(id int, name char, year float); insert into t values(1,'a',12); select * from t where id='1a';	
查看索引 show–index	10	1. 查看某表的索引信息。 2. 输出固定为表名、是否唯一索引、索引名称、列名和列在索引	show index from t;	

		标、列名和列在索引中的序号（针对多列索引）： Table | Non_unique | Key_name | Seq_in_index | Column_name t | 1 | index_id | 1 | id t2 | 1 | index_id | 1 | id1 t2 | 1 | index_id | 2 | id2		
增加date字段 date	10	1. 要求实现日期类型字段。 2. 当前已经支持了int、char、float类型，在此基础上实现date类型的字段。 3. date测试可能超过2038年2月，也可能小于1970年1月1号。注意处理非法的date输入，需要返回FAILURE。 4. 这道题目需要从词法解析开始，一直调整代码到执行阶段，还需要考虑DATE类型数据的存储。 5. 需要考虑date字段作为索引时的处理，以及如何比较大小; 6. 这里没有限制日期的范围，需要处理溢出的情况。 – 需要考虑	create table t(id int, birthday date); insert into t values(1, '2020-09-10'); insert into t values(2, '2021-1-2'); select * from t;	

		闰年。		
字符串匹配 like	10	1. like 操作符用于在 WHERE 子句中搜索符合一定格式的字段。 2. '%'用于匹配零个到多个任意字符（英文单引号“'”除外），'_'用于匹配一个任意字符（英文单引号“'”除外）。 3. 只考虑char类型的字段即可。	select * from table where name like 'W%'; select * from table where name kike '____' and not like '%m_';	
多表查询 select-tables	10	1. 当前系统支持单表查询的功能，需要在此基础上支持多张表的笛卡尔积关联查询，并考虑基本类型转换。 2. 需要实现select * from t1,t2; select t1.*,t2.* from t1,t2; 以及select t1.id,t2.id from t1,t2;查询可能会带条件。 3. 查询结果展示格式参考单表查询。每一列必须带有表信息，比如: t1.id | t2.id 1 | 1	select * from t1,t2; select * from t1,t2 where t1.id=t2.id and t1.age > 10; select * from t1,t2,t3;	typecast

聚合运算
aggregation-func
	10	1. 实现聚合函数 max/min/count/avg/sum。
		2. 可能出现如select id
			select max(age) from t1;
			select count(*) from t1;
			select count(1) from t1;
				typecast

		2. 可能出现如select id, count(age) from t;这样的聚合和单个字段混合的测试语句，返回FAILURE。 3. 聚合函数中的参数不会是表达式，比如age +1。 4. 注意avg函数会考察字符串的隐式转换。	select count(id) from t1;	
多表连接操作 join-tables	20	1. INNER JOIN，需要支持join多张表。主要工作是语法扩展。 2. 注意带有多条on条件的join操作。	select * from t1 inner join t2 on t1.id=t2.id; select * from t1 inner join t2 on t1.id=t2.id inner join t3 on t1.id=t3.id; select * from t1 inner join t2 on t1.id=t2.id and t2.age>10 where t1.name >='a';	typecast
一次插入多条数据 insert	10	1. 单条插入语句插入多行数据，一次插入的数据要同时成功或失败。	insert into t1 values(1,1),(2,2),(3,3);	
实现复杂更新功能 update-select	10	1. update多个字段，并且简单支持与select语句结合。注意select结果为多行的情况。 2. 需要考虑聚合运算。	update t set age=20,num=10 where id=2; update t1 set col1=(select col from t2 where t2.id=1) where t1.id=2;	typecast、aggregation-func
多列索引 multi-index	20	1. 多个字段关联起来称为单个索引。 2. 需要支持查看索引。	create index i_id on t1(id, age);	show-index
唯一索引 unique	10	1. 实现唯一索引：create unique index。	create unique index i_id on t1(id); insert into t1 values(1,1);	show-index、multi-index

		2. 需要支持多列的唯一索引。 3. 需要支持查看索引。 4. 不考虑在已有重复数据的列上建立唯一索引的情况。	insert into t1 values(1,2); --failed	

支持NULL类型
null
	20	1. 字段支持NULL值。包括但不限于建表、查询和（多行）插入。
		2. 默认情况不允许为NULL，使用nullable关键字表示字段允许为NULL。
		3. Null不区分大小写。
		4. 注意NULL字段的对比规则是NULL与**任何**数据对比，都是FALSE。
		5. 如果实现了NULL，需要在update/date类型/聚合语句/多行数据插入/join/复杂update/唯一索引中提供对NULL的支持。
			create table t1 (id int, age int, address char nullable);
			insert into t1 values(1,1, null);
			update t1 set address=null where id=1;
				update、date、select-tables、aggregation-func、join-tables、insert、update-select、unique

简单子查询 simple-sub-query	10	1. 支持简单的IN(NOT IN)语句，不涉及基本类型转换。注意NOT IN语句面对NULL时的特殊性。 2. 支持简单的EXISTS(NOT EXISTS)语句。 3. 支持与子查询结果做比较运算。 注意子查询结果为多行的情	select * from t1 where name in(select name from t2); select * from t where not exists (select * from t2 where t2.id > t1.id); select * from t1 where t1.age >(select max(t2.age) from t2); select * from t1 where t1.age > (select avg(t2.age)	typecast、aggregation-func、null

		况。 4. 支持子查询中带聚合函数。 5. 子查询中不会与主查询做关联。 6. 表达式中可能存在不同类型值比较。	from t2) and t1.age > 20.0;	
别名 alias	20	1. 表和列可以临时取别名，在打印结果时表和字段都打印别名（如果有）。 2. 在查询时能够使用表的别名访问表的字段。 3. 两个表的别名在同一层查询中不能重复，子查询里面和外面的表的别名可以重复。 4. 列的别名只需要支持查询结果显示，不需要考虑使用别名进行运算和比较，也不考虑列的别名重复。	select column_name as col from table_name; select t.column_name from table_name as t;	select-tables、aggregation-func、simple-sub-query

函数
function
	10	1. 实现一些常见的函数：length()、round()、date_format()。
		2. 其中length只考虑char类型，round只考虑float类型，date_format只考虑date类型，遇到其他的数据类型返回FAILURE即可。
			select length('this is a string') as len;
			select length(name) from table;
			select round(score, 2) from table;
			select date_format(u_date, '%Y-%m-%d') from table;
				date、select-tables、alias

超长字段 text	20	1. 超长字段的长度可能超出一页，比如常见的text,blob等。这里仅要求实现text（text长度固定4096字节），可以当做字符串实现。 2. 当前的查询，只能支持一次返回少量数据，需要扩展 如果输入的字符串长度，超过4096，那么应该保存4096字节，剩余的数据截断。 3. 需要调整record_manager的实现，当前record_manager是按照定长长度来管理页面的。	create table t(id int, age int, info text); insert into t values(1,1, 'a very very long string'); select * from t where id=1;	
查询支持表达式 expression	20	1. 查询中支持运算表达式，这里的运算表达式包括 +-*/。运算只需要考虑整数和浮点数的转换。 2. 仅支持基本数据的运算即可，不对date字段做考察。 运算出现异常，按照NULL规则处理。 3. 只会出现在select语句中。 4. 需要考虑聚合运算。	select * from t1,t2 where t1.age +10 > t2.age 2 + 3-(t1.age +10)/3; select t1.col1+t2.col2 from t1,t2 where t1.age +10 > t2.age *2 + 3-(t1.age +10)/3; select min(col1)+avg(col2)*max(col3) from exp_table where id < 10;	select-tables、aggregation-func
复杂子查询	20	1. 子查询在where条件中，子查询语句支持多张表与and和or条件	select * from t1 where age in (select id from t2 where t2.name in (select name	aggregation-func、simple-sub-query

complex-sub-query		多张表与and和or条件表达式，查询条件支持max/min等。 2. 注意考虑子查询与父表相关联的情况。	t2.name in (select name from t3))	sub-query
排序 order-by	10	1. 支持order by功能。 2. 不指定排序顺序默认为升序(asc)。 3. 不需要支持order by字段为纯数字的情况，比如select * from t order by 1。 4. 需要考虑排序字段为null的情况。	select * from t,t1 where t.id=t1.id order by t.id asc,t1.score desc;	select-tables、null
分组 group-by	20	1. 支持group by功能，按照一个或多个字段对查询结果分组，group by中的聚合函数不要求支持表达式。 2. 支持having子句，因为聚合函数不能出现在where后面，所以增加having子句用于筛选分组后的数据。 3. having只和聚合函数一起出现。 4. 需要考虑分组字段为null的情况。	select t.id, t.name, avg(t.score),avg(t2.age) from t,t2 where t.id=t2.id group by t.id,t.name; select count(id) from table group by name having count(id)>2;	select-tables、null

redo log
clog
	10	1. 当前redo log仅为雏形，只支持包含insert和delete的事务，要求在此基础上实现对简单update语句的支持。	create table t(id int);
			// client1
			begin;
			insert into t values(1);
			insert into t values(2);	

		2. 不需要考虑ddl以及索引的相关问题。
		3. 当前事务仅为雏形，隔离级别为读未提交。
		4. redo log部分对其余赛题实现没有影响，除此题外测试用例中不会使用begin;与commit;语句。
			// client2
			begin;
			insert into t values(3);
			update t set id = 4 where id = 2;
			
			// client1
			select * from t; （此时可以读到client2的未提交事务）
			commit;
			
			// close connection, restart server
			// client1
			select * from t; （此时无法读到client2的未提交事务）	

测试常见问题

测试Case

basic 测试

基础测试是隐藏的测试case，是代码本身就有的功能，比如创建表、插入数据等。如果选手把原生仓库代码提交上去，就能够测试通过basic。做其它题目时，可能会影响到basic测试用例，需要注意。

浮点数展示问题

按照输出要求，浮点数最多保留两位小数，并且去掉多余的0。目前没有直接的接口能够输出这种格式。比如 printf("%.2f", f); 会输出 1.00，printf("%g", f); 虽然会删除多余的0，但是数据比较大或者小数位比较多时展示结果也不符合要求。

1. select-meta 测试

这个测试对应了“元数据校验”。选手们应该先做这个case。

常见测试失败场景有一个是 where 条件校验时 server core了。

需要支持 `select *,col1,... from t;`，不支持 `select col1,* from t;`

2. drop-table case测试

目前遇到最多的失败情况是没有校验元数据，比如表删除后，再执行select，按照“元数据校验”规则，应该返回"FAILURE"。

3. update 测试

update 也要考虑元数据校验，比如更新不存在的表、更新不存在的字段等。

对于基本数据类型间的转换，将放在复杂update（update-select）进行考察。

更新需要考虑的几个场景，如果这个case没有过，可以对比一下：

假设存在这个表：

create table t (id int, name char, col1 int, col2 int);

--表上有个索引

create index i_id on t (id);

-- 单行更新

update t set name='abc' where id=1;

-- 多行更新

update t set name='a' where col1>2; -- 假设where条件能查出来多条数据

-- 更新索引列

update t set id=4 where name='c';

-- 全表更新

update t set col1=100;

-- where 条件有多个

update t set name='abc' where col1=0 and col2=0;

一些异常场景：

- 更新不存在的表
- 更新不存在的字段
- 查询条件中包含不合法的字段
- 查询条件查出来的数据集合是空（应该什么都不做，返回成功）
- 使用无法转换的类型更新某个字段，比如使用字符串更新整型字段

4. 基本类型转换

字符串转数字

– 如果字符串刚好是一个数字，则转换为对应的数字（如'1'转换为1，'2.1'转换为2.1）；

– 如果字符串的前缀是一个数字，则转换前缀数字部分为数字（如'1a1'转换为1，'2.1a'转换为2.1）；

– 如果字符串前缀不是任何合法的数字，则转换为0（不需要考虑前导符号 '+' '–'）；

问题：

– 如果转换数字溢出怎么处理？（不考虑）

– 是否考虑十六进制/八进制？（不考虑）

数字转字符串

问题：

– 转换是否带符号？（不带）

– 数字转换为字符串，长度溢出时如何处理？（不考虑溢出）

– 浮点数的表示方法是什么？（举例：1.5转换为'1.5'）

整数转浮点数

– 直接转换为浮点数

问题：

– 整数转换为浮点数溢出时怎么处理？（不考虑溢出）

浮点数转整数

问题：

– 四舍五入还是只保留整数部分？（四舍五入，如1.3转换为1，1.5转换为2）

涉及浮点数的比较

整数转换为浮点数，字符串转换为浮点数，再比较。

问题：

– 是否需要考虑日期与其它类型的转换？（不需要）

其他参考

MySQL 的类型转换可参考 https://dev.mysql.com/doc/refman/8.0/en/type-conversion.html

5. date 测试

date测试需要注意校验日期有效性。比如输入"2021-2-31"，一个非法的日期，应该返回"FAILURE"。

date不需要考虑和string(char)做对比。比如 select * from t where d > '123'; select * from t where d < 'abc'; 不会测试这种场景。但是需要考虑日期与日期的比较，比如select * from t where d > '2021-01-21';。

date也不会用来计算平均值。

select * form t where d=’2021-02-30‘； 这种场景在mysql下面是返回空数据集，但是我们现在约定都返回 FAILURE。

温馨提示：date 可以使用整数存储，简化处理。

6. 多表查询

多表查询的输入SQL，只要是字段，都会带表名。比如不会存在 select id from t1,t2;

不带字段名称的场景（会测试）： select * from t1,t2;

带字段：select t1.id, t1.age, t2.name from t1,t2 where t1.id=t2.id;

或者：select t1.* , t2.name from t1,t2 where t1.id=t2.id;

多表查询，查询出来单个字段时，也需要加上表名字。原始代码中，会把表名给删除掉。比如 select t1.id from t1,t2; 应该输出列名： t1.id。这里需要调整原始代码。输出列名的规则是：单表查询不带表名，多表查询带表名。

不要仅仅使用最简单的笛卡尔积运算，否则可能会内存不足。

7. 聚合运算

本题中的聚合运算只需要考虑单表的情况，后续的题目中用到聚合运算的可能涉及多表。

字符串可以不考虑AVG运算。

最少需要考虑的场景：

假设有一张表 create table t(id int, name char, price float);

select count(*) from t;

select count(id) from t;

select min(id) from t;

select min(name) from t; -- 字符串

select max(id) from t;

select max(name) from t;

select avg(id) from t; -- 整数做AVG运算，输出可能是浮点数，所以要注意浮点数输出格式

select avg(price) from t;

select avg(price), max(id), max(name) from t;

还需要考虑一些异常场景：

select count(*,id) from t;

select count() from t;

select count(not_exists_col) from t;

8. inner-join

inner-join 与 多表查询类似，很多同学做完多表查询就开始做inner-join了。 inner-join出现非常多的一个问题就是下面的语句，返回了空数据，或者没有任何返回，可能是测试时程序coredump，或者长时间没有返回结果，比如死循环。测试语句是：

select * from join_table_large_1 inner join join_table_large_2 on join_table_large_1.id=join_table_large_2.id inner join join_table_large_3 on join_table_large_1.id=join_table_large_3.id inner join join_table_large_4 on join_table_large_3.id=join_table_large_4.id and join_table_large_4.num4 <= 5 inner join join_table_large_5 on 1=1 inner join join_table_large_6 on join_table_large_5.id=join_table_large_6.id where join_table_large_3.num3 <10 and join_table_large_5.num5>90;

9. 支持NULL类型

NULL的测试case描述的太过简单，这里做一下补充说明。 NULL的功能在设计时，参考了mariadb的做法。包括NULL的比较规则：任何值与NULL做对比，结果都是**FALSE**。

因为miniob的特殊性，字段默认都是不能作为NULL的，所以在该题的测试用例中，要求增加关键字nullable，表示字段可以是NULL。

需要考虑的场景

- 建表

create table t(id int, num int nullable, birthday date nullable);

-- 表示创建一个表t,字段num和birthday可以是NULL, 而id不能是NULL。

- 建索引

create index i_num on t(num); -- 支持在可以为NULL的字段上建索引

- 支持增(包括多条插入)删改(包括复杂更新)查

insert into t values(1, 2, '2020-01-01');

insert into t values(1, null, null);

insert into t values(1, null, '2020-02-02'); -- 同学们自己多考虑几种场景

insert into t values(null, 1, '2020-01-02'),(2,null,null); -- 应该返回FAILURE，因为ID不能是NULL

update t set id=null, num=null where id =1;-- 应该返回FAILURE，因为ID不能是NULL

select * from t; -- 全表遍历

- null 条件查询 -- 同学们自己多测试几种场景

select * from t where id is null;

select * from t where id is not null;

select * from t where num is null;

select * from t where num > null;

select * from t where num <> null;

select * from t where 1=null;

select * from t where 'a'=null;

select * from t where null = null;

select * from t where null is null; -- 注意 = 与 is 的区别

select * from t where '2020-01-31' is null;

- 多表查询
- 多表连接
- 聚合

select count(*) from t;

select count(num) from t; -- 字段值是NULL时，比较特殊，不需要统计在内。

select avg(num) from t;-- 如果是AVG，字段值为NULL的行不进入统计。

-- 所有行的字段为NULL时则min/max/avg/sum结果为NULL

- 唯一索引

当唯一索引中某列允许为NULL将破坏该列的唯一性，如

create table t(id1 nullable, id2 nullable);

create unique index id1_id2 on t(id1,id2);

insert into t values(1,null);

insert into t values(1,null);-- 允许

insert into t values(2,2);

insert into t values(2,2);-- 不允许

10. 函数

- length()

length()函数用于计算字段的长度，对于字符串来说就是字符的个数，对于整数和浮点数来说就是有效数字的位数，这里简化了函数功能，只需要考虑字符串的长度，其他数据类型返回FAILURE即可。

length既可以出现在查询结果中，也可以出现在查询条件中，还可以出现在无表查询语句中：

select length('this is a string');

select length(name) from table where length(name)>5;

- round()

round()函数用于把数值字段舍入为指定的小数位数，这里只考虑浮点数。

round()函数可以只有一个参数，表示数据取整，它也可以有两个参数，第二个参数表示保留的小数位数。

select round(235.415);

select round(num, 2) from table;

- date_format()

date_format()函数用于格式化日期，这里只考虑日/月/天（d/m/y，D/M/Y），区分大小写。

字母含义对照表：

字母	含义
%Y	年，4 位，如2022
%y	年，2 位，如22
%M	月名，英文，如January
%m	月号，数字，如06、12
%D	月的一天，带英文后缀，如21st、28th
%d	月的一天，数字，如09、30

11. 表达式

表达式主要考虑加、减、乘、除四种运算，需要注意运算优先级。

测试的表达式中可能涉及到不同表的字段之间的运算，比如t1.id * t2.num，需要考虑整数和浮点数的比较，比如 t.id > 1.1 或者 5/4 = 1等。

对于除数为零的表达式，计算结果为null，根据null的规则进行处理，不要返回FAILURE。

12. Redo log

当前miniob实现了破产版的redo log，每个事务由begin日志开始，commit日志提交（其中单语句事务会自动开始/提交）。数据会首先写入log buffer，并在发现commit日志后持久化到log文件。

由于redo log没有实现checkpoint机制，miniob修改了缓存页的落盘机制，防止未持久化的日志所对应的事务操作提前下刷至外存数据页。可以理解当前的miniob是以日志保证持久性的数据库。

测试考察的核心是对于未commit的事务（主要是当前未实现的update语句），在observer未重启前能够被访问，而在重启后不再可见。

有余力的同学可以实现checkpoint机制，保证持久化的日志所对应的事务能够在一定程度上下刷至外存数据页，进而完善redo log的功能。