

# 测试说明文档

---

## 测试说明文档

### 题目二 查询执行

- 测试点1：尝试建表
- 测试点2：单表插入与条件查询
- 测试点3：单表更新与条件查询
- 测试点4：单表删除与条件查询
- 测试点5：连接查询
- 测试文件说明

### 题目三 BIGINT类型

- 测试文件说明

### 题目四 时间类型

- 测试点1：建表时创建时间类型的属性，并在该字段上进行增删改查
- 测试点2：对输入的合法性进行判断
- 测试文件说明

### 题目五 唯一索引

- 测试点1：创建、删除、展示索引
- 测试点2：索引查询
- 测试点3：索引维护
- 测试文件说明

### 题目六 聚合函数

- 测试点1：SUM
- 测试点2：MAX,MIN
- 测试点3：COUNT(),COUNT(\*)
- 测试文件说明

### 题目七 order by操作符

- 测试点
- 测试文件说明

### 题目八 块嵌套循环连接算法

- 测试点
- 测试文件说明

### 题目九 事务控制语句

- 测试点
- 测试文件说明

### 题目十 基于死锁预防的可串行化隔离级别

- 测试点：判断系统是否会出现五种数据异常
- 测试文件说明

### 题目十一 系统故障恢复

- 测试点
- 测试文件说明

## 题目二 查询执行

---

# 测试点1：尝试建表

测试示例：

```
create table t1(id int,name char(4));
show tables;
create table t2(id int);
show tables;
drop table t1;
show tables;
drop table t2;
show tables;
```

期待输出：

```
| Tables |
| t1 |
| Tables |
| t1 |
| t2 |
| Tables |
| t2 |
| Tables |
```

# 测试点2：单表插入与条件查询

测试示例：

```
create table grade (name char(20),id int,score float);
insert into grade values ('Data Structure', 1, 90.5);
insert into grade values ('Data Structure', 2, 95.0);
insert into grade values ('Calculus', 2, 92.0);
insert into grade values ('Calculus', 1, 88.5);
select * from grade;
select score,name,id from grade where score > 90;
select id from grade where name = 'Data Structure';
select name from grade where id = 2 and score > 90;
```

期待输出：

```
| name | id | score |
| Data Structure | 1 | 90.500000 |
| Data Structure | 2 | 95.000000 |
| Calculus | 2 | 92.000000 |
| Calculus | 1 | 88.500000 |
| score | name | id |
| 90.500000 | Data Structure | 1 |
```

95.000000	Data Structure	2
92.000000	Calculus	2
id		
1		
2		
name		
Data Structure		
Calculus		

## 测试点3：单表更新与条件查询

测试示例：

```
create table grade (name char(20),id int,score float);
insert into grade values ('Data Structure', 1, 90.5);
insert into grade values ('Data Structure', 2, 95.0);
insert into grade values ('Calculus', 2, 92.0);
insert into grade values ('Calculus', 1, 88.5);
select * from grade;
update grade set score = score + 5 where name = 'Calculus' ;
select * from grade;
update grade set name = 'Error name' where name > 'A';
select * from grade;
update grade set name = 'Error' ,id = -1,score = 0 where name = 'Error name' and score > 90;
select * from grade;
```

期待输出：

name	id	score
Data Structure	1	90.500000
Data Structure	2	95.000000
Calculus	2	92.000000
Calculus	1	88.500000
name	id	score
Data Structure	1	90.500000
Data Structure	2	95.000000
Calculus	2	97.000000
Calculus	1	93.500000
name	id	score
Error name	1	90.500000
Error name	2	95.000000
Error name	2	97.000000
Error name	1	93.500000
name	id	score
Error	-1	0.000000
Error	-1	0.000000
Error	-1	0.000000

```
| Error | -1 | 0.000000 |
```

## 测试点4：单表删除与条件查询

测试示例：

```
create table grade (name char(20),id int,score float);
insert into grade values ('Data Structure', 1, 90.5);
select * from grade;
delete from grade where score > 90;
select * from grade;
```

期待输出：

```
| name | id | score|
| Data Structure | 1 | 90.500000 |
| name | id | score|
```

## 测试点5：连接查询

测试示例：

```
create table t ( id int , t_name char (3));
create table d (d_name char(5),id int);
insert into t values (1,'aaa');
insert into t values (2,'baa');
insert into t values (3,'bba');
insert into d values ('12345',1);
insert into d values ('23456',2);
select * from t, d;
select t.id,t_name,d_name from t,d where t.id = d.id;
select t.id,t_name,d_name from t join d where t.id = d.id;
```

期待输出：

id	t_name	d_name	id
1	aaa	23456	2
1	aaa	12345	1
2	baa	23456	2
2	baa	12345	1
3	bba	23456	2
3	bba	12345	1
id	t_name	d_name	
1	aaa	12345	
2	baa	23456	
id	t_name	d_name	
1	aaa	12345	
2	baa	23456	

## 测试文件说明

- basic\_query\_test1: DDL语句，包含create table、drop table、show tables语句的测试和语义的检查。
- basic\_query\_test2: 单表插入与条件查询和语义检查。
- basic\_query\_test3: 单表更新与条件查询
- basic\_query\_test4: 单表删除与条件查询
- basic\_query\_test5: 连接查询与浮点数精度测试

## 题目三 BIGINT类型

测试示例：

```
CREATE TABLE t (bid bigint, sid int);
INSERT INTO t VALUES (372036854775807, 233421);
INSERT INTO t VALUES (-922337203685477580, 124332);
SELECT * FROM t;
INSERT INTO t VALUES (9223372036854775809, 12345);
SELECT * FROM t;
```

期待输出：

```
| bid | sid |
| 372036854775807 | 233421 |
| -922337203685477580 | 124332 |
failure
| bid | sid |
| 372036854775807 | 233421 |
| -922337203685477580 | 124332 |
```

# 测试文件说明

- 单个测试点：bigint字段的增删改查以及合法性检查

## 题目四 时间类型

### 测试点1：建表时创建时间类型的属性，并在该字段上进行增删改查

测试示例：

```
create table t(id int , time datetime);
insert into t values(1, '2023-05-18 09:12:19');
insert into t values(2, '2023-05-31 12:34:32');
select * from t;
delete from t where time = '2023-05-31 12:34:32';
update t set id = 2023 where time = '2023-05-18 09:12:19';
select * from t;
```

期待输出：

	id		time	
	1		2023-05-18 09:12:19	
	2		2023-05-31 12:34:32	
	id		time	
	2023		2023-05-18 09:12:19	

### 测试点2：对输入的合法性进行判断

测试示例：

```
create table t(time datetime, temperature float)
insert into t values('1999-07-07 12:30:00' , 36.0);
select * from t;
insert into t values('1999-13-07 12:30:00' , 36.0);
insert into t values('1999-1-07 12:30:00' , 36.0);
insert into t values('1999-00-07 12:30:00' , 36.0);
insert into t values('1999-07-00 12:30:00' , 36.0);
insert into t values('0001-07-10 12:30:00' , 36.0);
insert into t values('1999-02-30 12:30:00' , 36.0);
insert into t values('1999-02-28 12:30:61' , 36.0);
select * from t;
```

期待输出：

```
| time | temperature |
| 1999-07-07 12:30:00 | 36.000000 |
failure
failure
failure
failure
failure
failure
failure
failure
| time | temperature |
| 1999-07-07 12:30:00 | 36.000000 |
```

## 测试文件说明

- storage\_test1: 建表时创建时间类型的属性, 并在该字段上进行增删改查。
- storage\_test2: 对时间类型的属性的输入值的合法性进行判断。

## 题目五 唯一索引

### 测试点1: 创建、删除、展示索引

测试示例:

```
create table warehouse (id int, name char(8));
create index warehouse (id);
show index from warehouse;
create index warehouse (id,name);
show index from warehouse;
drop index warehouse (id);
drop index warehouse (id,name);
show index from warehouse;
```

期待输出:

```
| warehouse | unique | (id) |
| warehouse | unique | (id) |
| warehouse | unique | (id,name) |
```

### 测试点2: 索引查询

测试示例:

```
create table warehouse (w_id int, name char(8));
insert into warehouse values (10 , 'qweruiop');
```

```

insert into warehouse values (534, 'asdfhjkl');
insert into warehouse values (100, 'qwerghjk');
insert into warehouse values (500, 'bgtyhnmj');
create index warehouse(w_id);
select * from warehouse where w_id = 10;
select * from warehouse where w_id < 534 and w_id > 100;
drop index warehouse(w_id);
create index warehouse(name);
select * from warehouse where name = 'qweruiop';
select * from warehouse where name > 'qwerghjk';
select * from warehouse where name > 'aszdefgh' and name < 'qweraaaa';
drop index warehouse(name);
create index warehouse(w_id,name);
select * from warehouse where w_id = 100 and name = 'qwerghjk';
select * from warehouse where w_id < 600 and name > 'bztyhnmj';

```

期待输出：

```

| w_id | name |
| 10 | qweruiop |
| w_id | name |
| 500 | bgtyhnmj |
| w_id | name |
| 10 | qweruiop |
| w_id | name |
| 10 | qweruiop |
| w_id | name |
| 500 | bgtyhnmj |
| w_id | name |
| 100 | qwerghjk |
| w_id | name |
| 10 | qweruiop |
| 100 | qwerghjk |

```

## 测试点3：索引维护

测试示例：

```

create table warehouse (w_id int, name char(8));
insert into warehouse values (10, 'qweruiop');
insert into warehouse values (534, 'asdfhjkl');
select * from warehouse where w_id = 10;
select * from warehouse where w_id < 534 and w_id > 100;
create index warehouse(w_id);
insert into warehouse values (500, 'lastdanc');
insert into warehouse values (10, 'uiopqwer');
update warehouse set w_id = 507 where w_id = 534;
select * from warehouse where w_id = 10;

```



```
select * from warehouse where w_id < 534 and w_id > 100;
drop index warehouse(w_id);
create index warehouse(w_id,name);
insert into warehouse values(10,'qqqqoooo');
insert into warehouse values(500,'lastdanc');
update warehouse set w_id = 10, name = 'qqqqoooo' where w_id = 507 and name =
'asdfhjkl';
select * from warehouse;
```

期待输出：

```
| w_id | name |
| 10 | qweruiop |
| w_id | name |
failure
| w_id | name |
| 10 | qweruiop |
| w_id | name |
| 500 | lastdanc |
| 507 | asdfhjkl |
failure
failure
| w_id | name |
| 10 | qqqqoooo |
| 10 | qweruiop |
| 500 | lastdanc |
| 507 | asdfhjkl |
```

## 测试文件说明

- storage\_test3: 创建、删除、展示索引。
- storage\_test4: 索引查询。
- storage\_test5: 索引维护。需要保证建有索引的表中，不存在任意两个元组具有相同的索引属性值，当要插入或者修改的元组违背了唯一索引的要求时向output.txt输入failure。在测试文件中，不存在以下情况：在某张表上建立索引前，表中已经存在两个元组具有相同的索引属性值。
- judge\_whether\_use\_index\_on\_single\_attribute: 创建表并插入数据后，进行大量单列查询，记录耗时time\_a，在某列创建索引，再次进行大量单列查询，记录耗时time\_b。若 $\text{time\_b} / \text{time\_a} * 100\% \leq 70\%$ ，视为在单列查询时使用了索引。若判断为没有使用索引，则测试点2和测试点3零分。
- judge\_whether\_use\_index\_on\_multiple\_attributes: 创建表并插入数据后，进行大量多列查询，记录耗时time\_a，创建多列索引，再次进行大量多列查询，记录耗时time\_b。若 $\text{time\_b} / \text{time\_a} * 100\% \leq 70\%$ ，视为在多列查询时使用了索引。若判断为没有使用索引，则测试点2和测试点3零分。

## 题目六 聚合函数

## 测试点1: SUM

测试示例:

```
create table aggregate (id int, val float);
insert into aggregate values(1,5.5);
insert into aggregate values(3,4.5);
insert into aggregate values(5,10.0);
select SUM(id) as sum_id from aggregate;
select SUM(val) as sum_val from aggregate;
```

期待输出:

```
| sum_id |
| 9      |
| sum_val |
| 20.000000 |
```

## 测试点2: MAX,MIN

测试示例:

```
create table aggregate (id int, val float);
insert into aggregate values(1,5.5);
insert into aggregate values(3,4.5);
insert into aggregate values(5,10.0);
select MAX(id) as max_id from aggregate;
select MIN(val) as min_val from aggregate;
```

期待输出:

```
| max_id |
| 5      |
| min_val |
| 4.500000 |
```

## 测试点3: COUNT(),COUNT(\*)

测试示例:

```
create table aggregate (id int,name char(8),val float);
insert into aggregate values (1,'qwerasdf',1.0);
insert into aggregate values (2,'qwerasdf',2.0);
insert into aggregate values (3,'uiophjkl',2.0);
select COUNT(*) as count_row from aggregate;
select COUNT(id) as count_id from aggregate;
select COUNT(name) as count_name from aggregate where val = 2.0;
```

期待输出:

```
| count_row |
| 3 |
| count_id |
| 3 |
| count_name |
| 2 |
```

## 测试文件说明

- aggregate\_test1: sum函数测试, 浮点数保留6位小数, 整数不显示小数, as别名要求与SQL一致。
- aggregate\_test2: max、min函数测试。
- aggregate\_test3: count函数测试。

## 题目七 order by操作符

### 测试点

测试示例:

```
create table orders (company char(10), order_number int);
insert into orders values ('AAA',12);
insert into orders values ('ABB',13);
insert into orders values ('ABC',19);
insert into orders values ('ACA',1);
SELECT company, order_number FROM orders ORDER BY order_number;
SELECT company, order_number FROM orders ORDER BY company, order_number;
SELECT company, order_number FROM orders ORDER BY company DESC, order_number ASC;
SELECT company, order_number FROM orders ORDER BY order_number ASC LIMIT 2;
```

期待输出:

```
| company | order_number |
| ACA | 1 |
| AAA | 12 |
| ABB | 13 |
```

```
| ABC | 19 |  
| company | order_number |  
| AAA | 12 |  
| ABB | 13 |  
| ABC | 19 |  
| ACA | 1 |  
| company | order_number |  
| ACA | 1 |  
| ABC | 19 |  
| ABB | 13 |  
| AAA | 12 |  
| company | order_number |  
| ACA | 1 |  
| AAA | 12 |
```

## 测试文件说明

- 单个测试点（order\_by\_test）：包含单字段orderby、多字段orderby、limit字段、升序和降序测试

## 题目八 块嵌套循环连接算法

### 测试点

测试示例：

```
select * from t1, t2 where t1.id = t2.id;  
select * from t1, t2 where t1.id < t2.id and t2.id < 1000;
```

## 测试文件说明

- join\_test\_1：等值连接测试
- join\_test\_2：不等值连接测试

## 题目九 事务控制语句

### 测试点

测试示例：

```
create table student (id int, name char(8), score float);
insert into student values (1, 'xiaohong', 90.0);
begin;
insert into student values (2, 'xiaoming', 99.0);
delete from student where id = 2;
abort;
select * from student;
```

期待输出：

```
| id | name | score |
| 1 | xiaohong | 90.000000 |
```

## 测试文件说明

- commit\_test: 事务提交测试, 不包含索引
- abort\_test: 事务回滚测试, 不包含索引
- commit\_index\_test: 事务提交测试, 包含索引
- abort\_index\_test: 事务回滚测试, 包含索引

## 题目十 基于死锁预防的可串行化隔离级别

### 测试点：判断系统是否会出现五种数据异常

测试示例：

```
-- 对脏读数据异常进行测试：
create table concurrency_test (id int, name char(8), score float);
insert into concurrency_test values (1, 'xiaohong', 90.0);
insert into concurrency_test values (2, 'xiaoming', 95.0);
insert into concurrency_test values (3, 'zhanghua', 88.5);

-- 事务1的测试语句：
t1a begin;
t1b update concurrency_test set score = 100.0 where id = 2;
t1c abort;
t1d select * from concurrency_test where id = 2;

--事务2的测试语句：
t2a begin;
t2b select * from concurrency_test where id = 2;
t2c commit;
```

期待操作序列：

t1a t2a t1b t2b t1c t1d

## 测试文件说明

- concurrency\_read\_test: 并发读测试
- dirty\_write\_test: 脏写测试
- dirty\_read\_test: 脏读测试
- lost\_update\_test: 丢失更新测试
- unrepeatable\_read\_test: 不可重复读测试
- unrepeatable\_read\_test\_hard: 不可重复读测试
- phantom\_read\_test\_1: 幻读测试
- phantom\_read\_test\_2: 幻读测试
- phantom\_read\_test\_3: 幻读测试
- phantom\_read\_test\_4: 幻读测试

## 题目十一 系统故障恢复

### 测试点

测试示例:

```
create table t1 (id int, num int);
begin;
insert into t1 values(1, 1);
commit;
begin;
insert into t1 values(2, 2);
crash      // 系统接收到终止信号
...
重启系统
select * from t1;
```

## 测试文件说明

- crash\_recovery\_single\_thread\_test: 单个客户端连接, 系统故障恢复测试, 不包含索引
- crash\_recovery\_single\_thread\_test\_2: 单个客户端连接, 系统故障恢复测试, 包含索引
- crash\_recovery\_index\_test: 单个客户端连接, 包含索引, 对系统故障恢复过程中索引的一致性状态进行检查
- crash\_recovery\_multi\_thread\_test: 多个并发事务运行过程中故障, 系统故障恢复测试, 不包含索引
- crash\_recovery\_large\_data\_test: 多个并发事务运行过程中故障, 系统故障恢复测试, 包含索引

