数据库开发技术

151250216 庄宇州

1. 某服装店的后台管理系统

（1）请创建表Order（要求：oid为主键，其余字段不能为空）

```
CREATE TABLE `order` (
`oid` int(11) NOT NULL AUTO_INCREMENT,
`cuid` int(11) NOT NULL,
`cid` int(11) NOT NULL,
`quantity` int(11) NOT NULL,
`totalprice` double NOT NULL,
`ordertime` datetime NOT NULL,
PRIMARY KEY (`oid`)
)
```

（2）查询价格在100元到180元之间（包括100元和180元）的所有衬衫，列出它们的名字和单价，并按价格递减

```
SELECT c.name, c.price FROM clothes c WHERE c.type = '衬衫' AND c.price BETWEEN 120
 AND 180 ORDER BY c.price ASC
```

（3）查询【nike】2015年新上市的所有【裤子】的至今为止的各自的销量。

```
SELECT c.cid, ifnull(sum(o.quantity), 0) FROM clothes c LEFT JOIN `order` o ON c.c
id = o.cid WHERE c.type = '裤子' AND c.brand = 'nike' AND c.launchYear = 2015 GROUP
 BY c.cid
```

（4）查询顾客“jacky”在2014年11月这个月内购买服装所花的总费用。

```
SELECT sum(o.totalprice) FROM customer c, `order` o WHERE c.cname = 'jacky' AND c.
cuid = o.cuid AND year(o.ordertime) = 2014 AND month(o.ordertime) = 11
```

（5）查询同时购买了【nike】品牌【2015】年新上市的最贵的【外套】和【裤子】的顾客的【姓名】。

```
SELECT c.cname FROM customer c WHERE c.cuid IN (SELECT o1.cuid FROM `order` o1 WHE
RE o1.cid IN (SELECT c1.cid FROM clothes c1 WHERE c1.price = (SELECT max(c11.price
) FROM clothes c11 WHERE c11.brand = 'nike' AND c11.type = '外套' AND c11.launchYea
r = 2015) AND c1.type = '外套' AND c1.launchYear = 2015 AND c1.brand = 'nike')) AND
 c.cuid IN (SELECT o2.cuid FROM `order` o2 WHERE o2.cid IN (SELECT c2.cid FROM clo
thes c2 WHERE c2.price = (SELECT max(c12.price) FROM clothes c12 WHERE c12.brand =
 'nike' AND c12.type = '裤子' AND c12.launchYear = 2015) AND c2.type = '裤子' AND c2
.launchYear = 2015 AND c2.brand = 'nike'))
```

（6）查询2014.12.12这天销量排名【前三】的服装的【名称】，【销量】以及它们对应的【品牌】。

```
SELECT c.name, sum(o.quantity), c.brand FROM clothes c, `order` o WHERE c.cid = o.
cid AND o.ordertime BETWEEN '2014-12-12' AND '2014-12-13' GROUP BY c.cid ORDER BY
sum(o.quantity) DESC LIMIT 3
```

(7)查询2014.11.11，在所有购买了nike品牌服装的顾客中，消费金额最大的顾客的姓名和联系电话。

```
SELECT cu.cname, cu.phone FROM customer cu WHERE cu.cuid IN (SELECT o.cuid FROM cl
othes c, `order` o WHERE c.cid = o.cid AND c.brand = 'nike' AND o.ordertime BETWEE
N '2014-11-11' AND '2014-11-12' GROUP BY o.cuid ORDER BY sum(o.totalprice) DESC) L
IMIT 1
```

(8)查询2014.12.12这天，每个订单消费金额都在800元及以上的顾客的信息.

```
SELECT * FROM customer cu WHERE cu.cuid IN (SELECT o.cuid FROM `order` o WHERE o.o
rdertime BETWEEN '2014-12-12' AND '2014-12-13') AND cu.cuid NOT IN (SELECT o.cuid
FROM `order` o WHERE o.ordertime BETWEEN '2014-12-12' AND '2014-12-13' AND o.total
price < 800)
```

(9)删除2015年9月1日之前过去一年内没有消费过的顾客的信息。

```
DELETE FROM customer WHERE cuid NOT IN (SELECT o.cuid FROM `order` o WHERE o.order
time BETWEEN '2014-9-1' AND '2015-9-1')
```

(10)授予销售经理的账号Mike对表customer的更新、插入和查询权限，但不给删除权限。

```
GRANT UPDATE, INSERT, SELECT ON customer TO 'Mike';
REVOKE DELETE ON customer FROM 'Mike'
```

2. 某文件提交记录系统

(1)删除提交记录表中增加代码行数大于5000行，删除代码行数小于100行的提交记录。

```
DELETE FROM commit WHERE total_add > 5000 AND total_delete < 100
```

(2)查询项目中每一个迭代每一个学生的代码提交数量，显示迭代id，学生姓名，代码行数。

```
SELECT d.id, c.author, count(*) FROM commit c, deadline d WHERE c.datetime BETWEEN
 d.start_day AND d.end_day GROUP BY c.author, d.id
```

(3)查询项目中所有的java文件占总文件数量的比例，显示java文件的数量，总文件的数量。

```
SELECT (SELECT count(DISTINCT f1.filename) FROM file f1 WHERE f1.filename LIKE '%.
java') java_file, count(DISTINCT f2.filename) all_file FROM file f2
```

(4)查询项目过程中每个迭代中提交代码次数最多的日期，显示迭代号，提交日期，对应日期提交的次数。

```
SELECT *  (SELECT d.id id, DATE(c.datetime) commitDate, count(*) count FROM commit
 c, deadline d WHERE c.datetime BETWEEN d.start_day AND d.end_day GROUP BY d.id, c
ommitDate) count_iteration1 WHERE count_iteration1.count = (SELECT max(count_itera
tion2.count) FROM (SELECT d.id id, DATE(c.datetime) commitDate, count(*) count FRO
M commit c, deadline d WHERE c.datetime BETWEEN d.start_day AND d.end_day GROUP BY
 d.id, commitDate) count_iteration2 WHERE count_iteration2.id = count_iteration1.i
d)
```

(5)查询所有的文件行数超过200行的java文件（假设每个文件的初始行数为0行），并按照降序排列，显示文件名，文件的代码行数。

```
SELECT f.filename, sum(f.add_line) - sum(f.delete_line) line FROM file f GROUP BY
f.filename HAVING sum(f.add_line) - sum(f.delete_line) > 200 ORDER BY sum(f.add_li
ne) - sum(f.delete_line) DESC
```

(6)更新迭代表中迭代三的结束日期为原来结束日期的一周。

```
UPDATE deadline SET end_day = date_add(end_day,INTERVAL 1 WEEK) WHERE id = 3
```

3. 某付费文章阅读平台

(1)为数据库来自IP120.55.91.83的用户writer, 密码为writer， 设置文章作者表的增改查权限(该数据库的schema名称为platform)。

```
GRANT INSERT, UPDATE, SELECT ON platform_writer TO 'writer'@'120.55.91.83' IDENTIF
IED BY 'writer'
```

(2)查询姓名为zoe的读者最近付费的3篇文章的名称，内容和作者姓名。

```
SELECT pa.article_title, pa.content, pw.writer_name FROM platform_article pa, plat
form_deal pd, platform_reader pr, platform_writer pw WHERE pw.writer_id = pa.write
r_id AND pd.article_id = pa.article_id AND pd.reader_id = pr.reader_id AND pr.read
er_name = 'fabian' ORDER BY pd.create_time LIMIT 3
```

(3)查询所有文章中付费人数最多的前3篇文章的名字，付费人数及总付费金额。

```
SELECT pa.article_title, sum(pd.deal_payment) total_pay, count(*) total_people FRO
M  platform_deal pd, platform_article pa WHERE pa.article_id = pd.article_id GROUP
 BY pd.article_id ORDER BY count(*) DESC LIMIT 3
```

(4)平台所有的作者姓名(platform*writer表的write*name字段)需要添加“w*”前缀，如“Joe”需要修改为“w*Joe”。

```
UPDATE platform_writer SET writer_name = concat('w_', writer_name)
```

(5)新创建的作者姓名仍是不带“w*”前缀的，因此需要在插入数据时自动为其添加“w*”前缀(用触发器解决，触发器的名称定义为“modifywritername”)。

```
CREATE TRIGGER modifywritername
BEFORE INSERT ON platform_writer
FOR EACH ROW
BEGIN
SET NEW.writer_name = concat('w_', NEW.writer_name);
END;
```

(6)查询每位作者的名称，该作者发表的文章总数，该作者的所有文章付费用户总数，按付费用户总数倒序排序。

```
SELECT table1.writer_name writer_name, table2.article_count, table1.user_count FRO
M (SELECT pw.writer_name, pw.writer_id, count(DISTINCT pr.reader_id) user_count FR
OM platform_writer pw, platform_article pa, platform_deal pd, platform_reader pr W
HERE pw.writer_id = pa.writer_id AND pd.article_id = pa.article_id AND pr.reader_i
d = pd.reader_id GROUP BY pw.writer_id ORDER BY count(DISTINCT pr.reader_id) DESC)
 table1, (SELECT pw.writer_name, pw.writer_id, count(pa.article_id) article_count
FROM platform_writer pw, platform_article pa WHERE pw.writer_id = pa.writer_id GRO
UP BY pw.writer_id) table2 WHERE table1.writer_id = table2.writer_id
```

(7)创建一个试图article_writer， 包含文章的所有字段，文章的付费总额，文章作者的姓名和邮箱。

```
CREATE VIEW article_writer AS SELECT pa.content, pw.writer_name, pw.writer_email, ifnull(sum(pd.deal_payment), 0) payment FROM platform_writer pw, platform_article pa, platform_deal pd WHERE pw.writer_id = pa.writer_id AND pd.article_id = pa.article_id GROUP BY pa.article_id
```

(8)由于create_time是datetime格式，现在需要将其中的日期提取出来，查询每位读者每日的付费阅读总数和付费金额，结果集中包含读者ID，姓名，交易日期，当日付费阅读量，当日付费金额，并按照日期降序排序。

```
SELECT pr.reader_id, DATE(pd.create_time) date_time, count(*) read_count, sum(pd.deal_payment) payment_count FROM platform_reader pr, platform_deal pd WHERE pr.reader_id = pd.reader_id GROUP BY pr.reader_id, date_time ORDER BY date_time DESC
```