

## 作业7-10185102142-李泽浩

1.先在account表中执行更新语句，将最后活跃日期和当前时间相距超过三年的所有账户的Status属性设置为“不活跃”。然后，在acc\_transaction上定义一个AFTER INSERT触发器，触发器名为“d\_newTransaction”，当对acc\_transaction表执行任何插入操作后被启动，将该账户在account表中的Status属性设置为“正常”。（根据需要可能需要删除该表已有触发器）

### #更新操作

```
UPDATE account
SET STATUS = '不活跃'
WHERE CURDATE() > ADDDATE(LAST_ACTIVITY_DATE, INTERVAL 3 YEAR);
```

### #触发器

```
DROP TRIGGER IF EXISTS d_newTransaction;

CREATE TRIGGER d_newTransaction AFTER INSERT
ON acc_transaction FOR EACH ROW
BEGIN
    UPDATE account SET STATUS='正常'
    WHERE account.ACCOUNT_ID = NEW.ACCOUNT_ID AND account.STATUS='不活跃';
END;
```

2.在员工表中添加一项字段SIMPLE\_KPI,字段类型int(11),不可空，初始值为0。在acc\_transaction上定义AFTER INSERT触发器，触发器名为“e\_newTransaction” 当对acc\_transaction表执行任何插入操作后被启动，根据TELLER\_EMP\_ID，员工表中的SIMPLE\_KPI+=1，如果有上级领导SUPERIOR\_EMP\_ID，则他的SIMPLE\_KPI也+=1，迭代直到上级领导为空。

### #添加字段

```
ALTER TABLE employee ADD COLUMN SIMPLE_KPI INT(11) NOT NULL DEFAULT(0);

DROP TRIGGER IF EXISTS e_newTransaction;

CREATE TRIGGER e_newTransaction
AFTER INSERT
ON acc_transaction FOR EACH ROW
BEGIN
    DECLARE temp_id INT;
    SET temp_id = new.TELLER_EMP_ID;

    WHILE temp_id IS NOT NULL DO
        UPDATE employee SET employee.SIMPLE_KPI = employee.SUPERIOR_EMP_ID + 1
        WHERE employee.EMP_ID = temp_id;

        SELECT employee.SUPERIOR_EMP_ID INTO temp_id
    END WHILE;
END;
```

```

FROM employee
WHERE employee.EMP_ID = temp_id;
END WHILE;

END;

```

3.请创建一个新表instructor(ID int, name varchar(20), dept\_name varchar(10), salary int), 并插入一条教师信息 (12121, 'Wu', 'Comp. Sci.', 5000) , 进行MySQL隔离级别验证实验。完成实验后删除instructor表。

```


DROP TABLE IF EXISTS instructor;

CREATE TABLE instructor (
    ID int,
    name varchar(20),
    dept_name varchar(10),
    salary int,
    PRIMARY KEY (ID)
);

INSERT INTO instructor VALUES(12121, 'Wu', 'Comp. Sci.', 5000);

DROP TABLE IF EXISTS instructor;

```




ID	name	dept_name	salary
12121	Wu	Comp. Sci.	5000

(1) 将instructor表中编号12121的名为Wu的教工的salary设置为10000元

```

UPDATE instructor SET salary = 10000
WHERE ID = 12121;

```



ID	name	dept_name	salary
12121	Wu	Comp. Sci.	10000

以下实验在命令行中实现

(2) 同时开两个命令行界面，称为A和B，在A、B两个窗口都执行下面的语句，设定隔离级别为“读未提交”。  
 set tx\_isolation='read-uncommitted';  
 可以执行select @@session.tx\_isolation;验证设定是否已经成功。

```

set transaction_isolation='read-uncommitted';
select @@session.transaction_isolation;

#show variables like '%isolation%';

```

```
mysql> set transaction_isolation='read-uncommitted';
Query OK, 0 rows affected (0.00 sec)

mysql> select @@session.transaction_isolation;
+-----+
| @@session.transaction_isolation |
+-----+
| READ-UNCOMMITTED                 |
+-----+
1 row in set (0.00 sec)
```

(3) 在A窗口执行如下语句：

```
begin;
```

```
select * from instructor;
```

可以看到所有教工的信息，编号12121的名为Wu的教工的salary当前值应该为10000

A窗口结果：

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+----+-----+-----+-----+
| ID  | name | dept_name | salary |
+----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 10000 |
+----+-----+-----+-----+
1 row in set (0.00 sec)
```

(4) 在B窗口执行如下语句：

```
begin;
```

```
update instructor set salary = salary - 500 where id= 12121;
```

```
select * from instructor;
```

B 窗口结果：

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> update instructor set salary = salary - 500 where id= 12121;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from instructor;
+----+-----+-----+-----+
| ID  | name | dept_name | salary |
+----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500 |
+----+-----+-----+-----+
1 row in set (0.00 sec)
```

(5) 在A窗口执行如下语句：

```
select * from instructor;
```

请回答：

Q1： 是否可以看到编号12121的名为Wu的教工的salary值被B窗口修改但未提交的结果， 回答是/否。

是

Q2： 这种现象的名称是？

脏读

(6) 在B窗口执行如下语句：

rollback;

```
mysql> rollback;  
Query OK, 0 rows affected (0.00 sec)
```

(7) 在A窗口执行如下语句：

update instructor set salary = salary - 500 where id= 12121;

select \* from instructor;

Q：当前编号12121的名为Wu的教工的salary值为多少？

**9500**

```
mysql> update instructor set salary = salary - 500 where id= 12121;  
Query OK, 1 row affected (0.00 sec)  
Rows matched: 1  Changed: 1  Warnings: 0  
  
mysql> select * from instructor;  
+-----+-----+-----+-----+  
| ID    | name | dept_name | salary |  
+-----+-----+-----+-----+  
| 12121 | Wu   | Comp. Sci. | 9500   |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

(8) 在A窗口执行如下语句：

rollback;

select \* from instructor;

```
mysql> rollback;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> select * from instructor;  
+-----+-----+-----+-----+  
| ID    | name | dept_name | salary |  
+-----+-----+-----+-----+  
| 12121 | Wu   | Comp. Sci. | 10000  |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

(9) 在A、B两个窗口都执行下面的语句， 设定隔离级别为“读已提交”。

set tx\_isolation='read-committed';

可以执行select @@session.tx\_isolation;验证设定是否已经成功。

```
mysql> set transaction_isolation='read-committed';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> select @@session.transaction_isolation;  
+-----+  
| @@session.transaction_isolation |  
+-----+  
| READ-COMMITTED                  |  
+-----+  
1 row in set (0.00 sec)
```

(10) 在A窗口执行下面语句：

begin;

select \* from instructor;

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 10000  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(11) 在B窗口执行下面语句：

```
begin;
select * from instructor;
update instructor set salary = salary - 500 where id= 12121;
```

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 10000  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> update instructor set salary = salary - 500 where id= 12121;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

(12) 在A窗口执行如下语句：

```
select * from instructor;
```

请回答：

Q1：当前编号12121的名为Wu的教工的salary值为多少？

**10000**

```
mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 10000  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Q2：是否可以看到编号12121的名为Wu的教工的salary值被B窗口修改但未提交的结果，回答是/否

**否**

(13) 在B窗口执行下面语句：

```
commit;
select * from instructor;
```

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(14) 在A窗口执行如下语句：

```
select * from instructor;
```

```
commit;
```

请回答：

Q1：当前编号12121的名为Wu的教工的salary值为多少？

**9500**

```
mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

Q2：是否可以看到编号12121的名为Wu的教工的salary值被B窗口修改并已提交的结果，回答是/否

**是**

Q3：A窗口当前两次的select \* from instructor;查询结果是否一致？

**不一致**

Q4：这种现象的名称是？

**不可重复读**

(15) 在A、B两个窗口都执行下面的语句，设定隔离级别为“可重复读”。

```
set tx_isolation='repeatable-read';
```

可以执行select @@session.tx\_isolation;验证设定是否已经成功。

```
mysql> set transaction_isolation='repeatable-read';
Query OK, 0 rows affected (0.00 sec)

mysql> select @@session.transaction_isolation;
+-----+
| @@session.transaction_isolation |
+-----+
| REPEATABLE-READ                  |
+-----+
1 row in set (0.00 sec)
```

(16-1) 在A窗口执行下面语句：

```
begin;
```

```
select * from instructor;
```



```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(16-2) 在B窗口执行下面语句：

```
begin;
select * from instructor;
update instructor set salary = salary - 500 where id= 12121;
commit;
select * from instructor;
```

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> update instructor set salary = salary - 500 where id= 12121;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9000   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(17) 在A窗口执行如下语句：

```
select * from instructor;
```

```
mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 9500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Q1：当前编号12121的名为Wu的教工的salary值为多少？

**9500**

Q2：是否可以看到编号12121的名为Wu的教工的salary值被B窗口修改并已提交的结果，回答是/否

**否**

Q3：A窗口当前两次的select \* from instructor;查询结果是否一致？

一致

(18) 在A窗口执行如下语句：

```
update instructor set salary = salary - 500 where id= 12121;
commit;
select * from instructor;
```

Q1：当前编号12121的名为Wu的教工的salary值为多少？

**8500**

```
mysql> update instructor set salary = salary - 500 where id= 12121;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 8500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(19) 在A窗口执行如下语句：

```
begin;
select * from instructor;
```

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 8500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(20) 在B窗口执行下面语句：

```
begin;
insert into instructor values(99999,'test','Comp. Sci.',10000);
commit;
select * from instructor;
```

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into instructor values(99999,'test','Comp. Sci.',10000);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 8500   |
| 99999 | test | Comp. Sci. | 10000  |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```



(21) 在A窗口执行如下语句：

```
select * from instructor;
```

Q1：是否可以看到B窗口新插入并已提交的结果，回答是/否

否

```
mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 8500   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

(22) 在A窗口执行如下语句：

```
insert into instructor values(99999,'test','Comp. Sci.',10000);
```

Q1：A窗口是否能插入此条数据，回答是/否

否

```
mysql> insert into instructor values(99999,'test','Comp. Sci.',10000);
ERROR 1062 (23000): Duplicate entry '99999' for key 'instructor.PRIMARY'
```

Q2：这种现象的名称是？

幻读

(23) 在A窗口执行如下语句：

```
update instructor set salary = 6666 where id=99999;
```

```
select * from instructor;
```

```
commit;
```

Q1：A窗口是否更新了一条自己会话中查不到的数据，回答是/否

是

```
mysql> update instructor set salary = 6666 where id=99999;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from instructor;
+-----+-----+-----+-----+
| ID    | name | dept_name | salary |
+-----+-----+-----+-----+
| 12121 | Wu   | Comp. Sci. | 8500   |
| 99999 | test | Comp. Sci. | 6666   |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> commit;
Query OK, 0 rows affected (0.01 sec)
```

## 4.加载JDBC驱动程序，利用JDBC连接数据库

(1) 通过PreparedStatement对象实现往客户表customer添加一条数据的操作：'1'，'上海市南京西路11号1021室'，'上海市'，'I'，'上海市'，'027027'，'1000'

(2) 用Statement对象实现修改操作，将联系地址为'上海市南京西路11号1021室'的客户类型编号修改为'B'。

```

import java.sql.*;

public class Main {

    public static void main(String[] args) {
        try{
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e){
            System.out.println("抛出异常");
            e.printStackTrace();
        }
        try {
            String url = "jdbc:mysql://localhost:3306/bank";
            String user = "root";
            String pass = "mysql";
            Connection connection = DriverManager.getConnection(url, user, pass);
            System.out.println("连接成功! ");

            String sql1="insert into customer values(?,?,?,?,?,?)";
            PreparedStatement preparedStatement=connection.prepareStatement(sql1);
            preparedStatement.setInt(1,1);
            preparedStatement.setString(2,"上海市南京西路11号1021室");
            preparedStatement.setString(3,"上海市");
            preparedStatement.setString(4,"I");
            preparedStatement.setString(5,"上海市");
            preparedStatement.setString(6,"027027");

            int temp=preparedStatement.executeUpdate();
            if(temp!=0){
                System.out.println("记录添加成功");
            }
            else{
                System.out.println("记录添加失败");
            }

            Statement statement=connection.createStatement();
            String sql2="update customer set cust_type_cd='B' where address='上海市南京西
路11号1021室'";
            temp=statement.executeUpdate(sql2);
            if(temp!=0){
                System.out.println("记录更新成功");
            }
            else{
                System.out.println("记录更新失败");
            }
        } catch (SQLException e) {
            System.out.println("连接数据库失败");
            e.printStackTrace();
        }
    }
}

```

