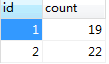
首先，先介绍下phonecount表，phonecount表只有两个字段，同时也只有两行数据。

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 备注 |
| Id | Int | Id |
| Count | Int | 计数值 |

该表中只有两行数据，具体数据是干什么的，我会在后面介绍程序思路的时候详细说明，现在只是初步说明。Id=1的时候count值总小于等于Id=2的时候的count值。



Sign表

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 备注 |
| Id | Int | Id |
| phoneSign | Int | 标志（unique） |

Promotion表

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 备注 |
| Id | Int | Id |
| Phone | Bigint | 号码（unique） |
| Price | Int | 价格 |

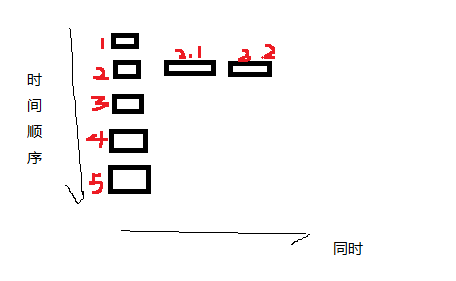
活动的大概流程是：

参与者输入号码和价格🡪系统判断号码是否参与过，该价格是否是当前活动🡪给出相应反馈

那么我们怎么给参与者正确的反馈呢？

现在我就介绍下我的思路。

首先，我们按时间顺序排一列队，同时间请求的排一横队。如下图所示



我们按顺序给1号一个标志0，先将标志插入sign表的phoneSign字段里，成功后再将1号的数据插入到promotion表中。（这就是我们生活中的取号办事。）

现在来说说标志是什么：

在这里我们可以把标志类比为号码牌，从0～49999中的唯一的一个值，每个时间点只能给当前时间点的号码。因为现在给第一个时间点标志，所以此时的标志值为0。

在第二个时间点的时候，有三个人发来请求，但是只能有一个人能获得标志。这就会不公平啊，明明还有好多号码，为什么我这么靠前但是却不给我标志呢。因此就在之后未发的号码里随机拿一个给他们。然后顺序处理每个时间点的请求。

这个时候就会出现两个数据，一个是当前时间点的号码（标志），另外一个是已经给了多少个号码。这就是phonecount表里的两行数据，id=1时，当前时间点的号码，id=2时，总共已经给的号码。

好了，写了这么多。铺垫基本已经完成了。现在来写UserPhoneImpl.java的详细思路。

**public** **class** UserPhoneImpl **implements** UserPhoneDao {

@Override

**public** **int** addPhone(UserPhone phone) {

**int** flag = 0; //是否成功

DataConnection dConn = **new** DataConnection();

Connection conn = dConn.getConn();

PreparedStatement pStat = **null**;

**try** {

conn.setAutoCommit(**false**);

String sql = "select count from phoneCount where id = 1";//首先获取当前时间点的号码

Statement stat = conn.createStatement();

ResultSet rs = stat.executeQuery(sql);

**int** count = 0;//用count存取出的值

**while**(rs.next()) {

count = rs.getInt(1);

}

//这个代码块是判断当前时间点的号码count是否已经被用过了，因为并发会用到后面未到的时间点的号码。如果已经用过了就将count+1，然后继续判断，直到count是个没有用过的时间点的号码。

sql = "select id from sign where phoneSign = " + count + "";

ResultSet re = stat.executeQuery(sql);

**int** contained = 0;

**while**(re.next()) {

contained = re.getInt(1);

}

**while**(contained > 0) {

count ++;

contained = 0;

sql = "select id from sign where phoneSign = " + count + "";

re = stat.executeQuery(sql);

**while**(re.next()) {

contained = re.getInt(1);

}

}

//这个代码块是标志插入的算法。先将标志插入sign表。如果插入失败，则为count赋在count～本级最大值之间随机一个count，再插入一次如果这次还失败，则说明现在是并发量高峰期，已无更多资源给该用户了。如果插入成功，则当前号码（标志）已经用过，数据库中当前时间点号码+1.

sql = "insert into sign (phoneSign) values (" + count + ") "; **try** {

stat.executeUpdate(sql);

sql = "update phoneCount set count = count + 1 where id = 1";//插入成功，数据库中当前时间点号码+1.（id=1）

stat.executeUpdate(sql);

} **catch** (SQLException e) {

e.printStackTrace();

**int** count2 = (**int**)(Math.*random*() \* (phone.getPrice() \* 10000 - count)) + count; //这句代码就是实现在count～本级最大值之间随机一个count的

sql = "insert into sign (phoneSign) values (" + count2 + ") ";

stat.executeUpdate(sql);

}

//这个代码块是最后一个阶段，将号码和价格插入promotion表。如果成功，该用户基本已经购买成功了，到了这个阶段，肯定消耗了一个标志（不论是不是当前时间点的标志），因此消耗总数+1（id=2）。最后还会再读取消耗标志的总数，看是不是超过当前阶段的最大值，如果超过了，就回滚，否则就成功了。返回flag

sql = "insert into promotion (phone,price) values (?,?) "; //将号码和价格插入promotion表

pStat = conn.prepareStatement(sql);

pStat.setLong(1, phone.getPhone());

pStat.setInt(2, phone.getPrice());

flag = pStat.executeUpdate();

sql = "update phoneCount set count = count + 1 where id = 2";//消耗总数+1（id=2）

stat.executeUpdate(sql);

sql = "select count from phoneCount where id = 2"; //读取当前消耗标志的总数

ResultSet res = stat.executeQuery(sql);

**while**(res.next()) {

count = res.getInt(1);

}

**if**(stat != **null**){

stat.close();

}

**if**(count - phone.getPrice() \* 10000 <= 0) {//看是否越界了。

conn.commit();

} **else** {

flag = 0;

conn.rollback();

}

} **catch** (Exception e) {

e.printStackTrace();

**try** {

conn.rollback();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

} **finally**{

**try** {

pStat.close();

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**return** flag;

}

}