###### 7　并发事务问题之丢失更新

　　丢失更新：一个事务的更新被另一个事务的更新覆盖了；

|  |  |  |
| --- | --- | --- |
| 时间点 | 事务1 | 事务2 |
| t1 | 开始事务 |  |
| t2 |  | 开始事务 |
| t3 | 查询pid=p1的记录结果为[pid=p1,pname=zhangSan,age=23,sex=male] |  |
| t4 |  | 查询pid=p1的记录结果为[pid=p1,pname=zhangSan,age=23,sex=male] |
| t5 | 修改age=24，其它保留原值，即：  update person set pname=’zhangSan’,  age=24,sex=’male’ where pid=’p1’; |  |
| t6 | 提交事务 |  |
| t7 |  | 修改sex=female，其它保留原值  update person set pname=’zhangSan’,  age=23,sex=’female’ where pid=’p1’; |
| t8 |  | 提交事务 |

事务2覆盖了事务1的更新操作。结果为：[pid=p1,pname=zhangSan,age=23,sex=female]。因为事务2没有在事务1的基础上进行更新，而是在自己的查询基础上进行更新。

|  |
| --- |
| **public** **class** Demo1 {  **private** **static** Connection getConnection() **throws** Exception {  String driverClassName = "com.mysql.jdbc.Driver";  String url = "jdbc:mysql://localhost:3306/day12?useUnicode=true&characterEncoding=utf8";  String username = "root";  String password = "123";  Class.*forName*(driverClassName);  **return** DriverManager.*getConnection*(url, username, password);  }  **public** Person load(Connection con, String pid) **throws** Exception {  String sql = "select \* from t\_person where pid=?";  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, pid);  ResultSet rs = pstmt.executeQuery();  **if** (rs.next()) {  **return** **new** Person(rs.getString(1), rs.getString(2), rs.getInt(3),  rs.getString(4));  }  **return** **null**;  }    **public** **void** update(Connection con, Person p) **throws** Exception {  String sql = "update t\_person set pname=?, age=?, gender=? where pid=?";  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, p.getPname());  pstmt.setInt(2, p.getAge());  pstmt.setString(3, p.getGender());  pstmt.setString(4, p.getPid());    pstmt.executeUpdate();  }    @Test  **public** **void** fun1() **throws** Exception {  Connection con = *getConnection*();  con.setAutoCommit(**false**);    //[pid=p1,pname=zs,age=24,gender=male]  Person p = load(con, "p1");  p.setAge(42);//断点  update(con, p);    con.commit();  }  @Test  **public** **void** fun2() **throws** Exception {  Connection con = *getConnection*();  con.setAutoCommit(**false**);  //[pid=p1,pname=zs,age=24,gender=male]  Person p = load(con, "p1");  p.setGender("female");//断点  update(con, p);    con.commit();  }  } |

处理丢失更新：

* 悲观锁：在查询时给事务上排他锁，这可以让另一个事务在查询时等待前一个事务解锁后才能执行；
* 乐观锁：给表添加一个字段，表示版本，例如添加version字段，比较查询到的version与当前vesion是否相同；

7.1　悲观锁解决丢失更新

只需要修改上面代码的load()方法中select语句即可：

select \* from t\_person where pid=? for update

|  |
| --- |
| **public** Person load(Connection con, String pid) **throws** Exception {  String sql = "select \* from t\_person where pid=? for update";  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, pid);  ResultSet rs = pstmt.executeQuery();  **if** (rs.next()) {  **return** **new** Person(rs.getString(1), rs.getString(2), rs.getInt(3),  rs.getString(4));  }  **return** **null**;  } |

悲观锁：悲观的思想，认为丢失更新问题总会出现，在select语句中添加for update为事务添加排他锁，这会让其他事务等待当前事务结束后才能访问。当然，其他事物的select语句中也要加上for update语句才会等待；

悲观锁的性能低！

7.2　乐观锁

乐观锁与数据库锁机制无关；

我们需要修改t\_person表，为其添加一个字段表示当前记录的版本。例如给t\_person表添加version字段，默认值为1。

当事务查询记录时得到version=1，再执行update时需要比较当前version的值是否与查询到的version相同，决定update是否执行成功。如果update成功，还要把version的值加1。

|  |
| --- |
| **public** **void** update(Connection con, Person p) **throws** Exception {  String sql = "update t\_person set pname=?, age=?, gender=?, version=version+1 where pid=? and version=?";  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, p.getPname());  pstmt.setInt(2, p.getAge());  pstmt.setString(3, p.getGender());  pstmt.setString(4, p.getPid());  pstmt.setInt(5, p.getVersion());    pstmt.executeUpdate();  } |

* 事务1：查询时得到version=1；
* 事务2：查询时得到version=1；
* 事务1：执行update时因为version没有改变，所以update执行成功，update不只修改了age=42，还修改了version=2；
* 事务2：执行update语句时version已经为2，而查询时的version为1，所以update执行失败；

乐观锁：与数据库锁机制无关，乐观的思想，认为丢失更新不是总出现；通过给表添加版本字段来决定update操作是否成功。即查询时和更新时的版本必须一致！