**任课教师：张涵翠、霍戌文**

**《数据库系统实训》**

**（2021-2022学年第2学期）**

**实**

**验**

**报**

**告**

**学号：2019329621004**

**姓名：梅雨欣**

**班级：19计算机科学与技术（1）班**

**实验十：事务**

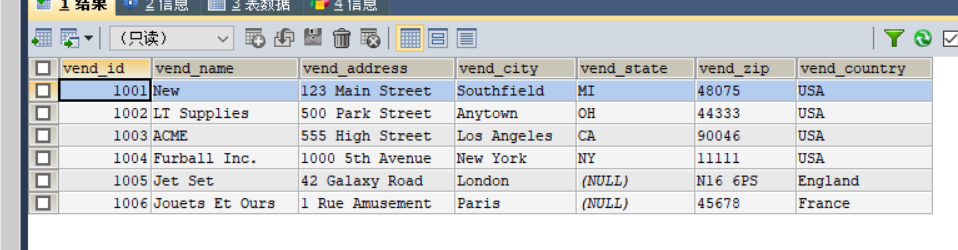
**事务的应用**

**第一部分：样例库的应用**

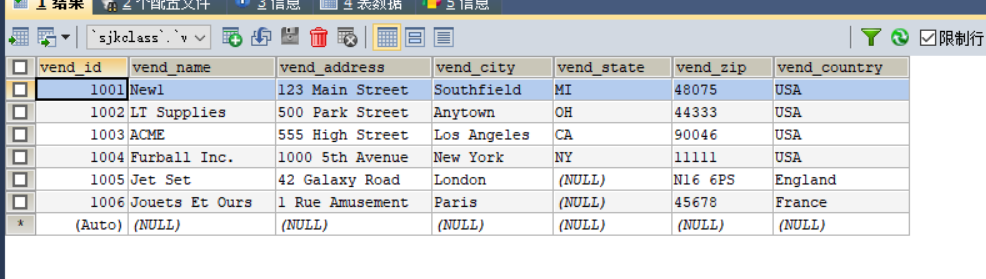
**1. 关闭自动提交功能**

1. **自动提交功能关闭前：**

|  |
| --- |
| update vendors set vend\_name=’New’ where vend\_id=1001;  select \* from vendors; |



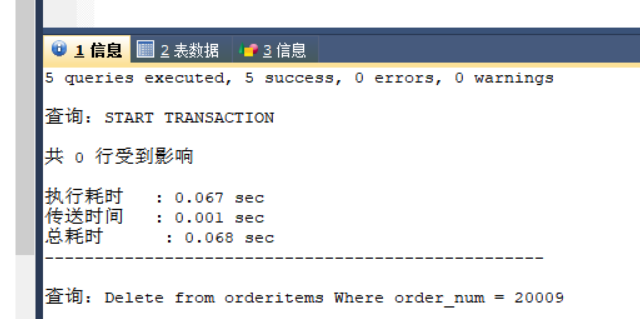
**2）自动提交功能关闭后：**



**2. 开始事务**

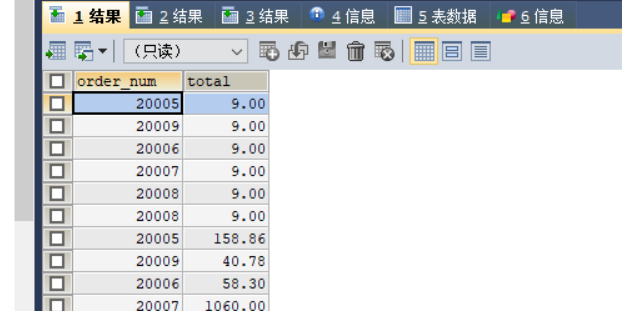
**显示开启：**

|  |
| --- |
| START TRANSACTION;  Delete from orderitems Where order\_num = 20009;  Delete from orders Where order\_num = 20009;  COMMIT; |



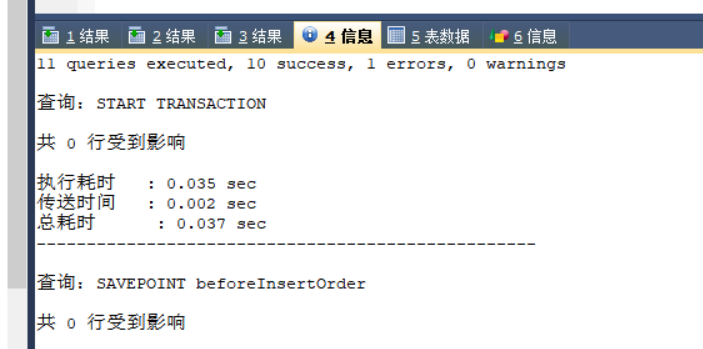
**3.撤销事务**

|  |
| --- |
| SELECT \* FROM ordertotals;  START TRANSACTION;  DELETE FROM ordertotals;  SELECT \* FROM ordertotals;  ROLLBACK;  SELECT \* FROM ordertotals; |



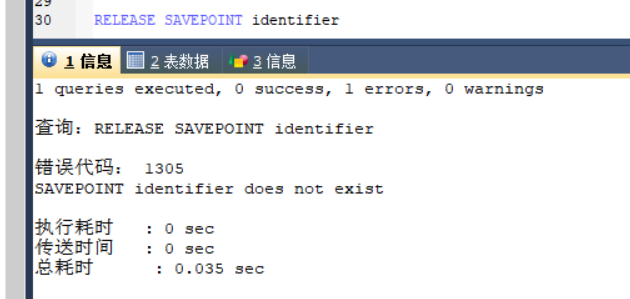
**4. 使用保留点回滚**

|  |
| --- |
| START TRANSACTION;  SAVEPOINT beforeInsertOrder;  Insert into orders values(20009, now(), 10001);  Select \* from orders;  SAVEPOINT beforeInsertOrderItem;  Insert into orderitems values(20009, 1, FB, 1, 10);  Select \* from orderitems;  ROLLBACK to beforeInsertOrderItem;  Select \* from orderitems;  Insert into orderitems values(20009, 1, FC, 1, 2.5);  COMMIT; |



**5.删除保留点**

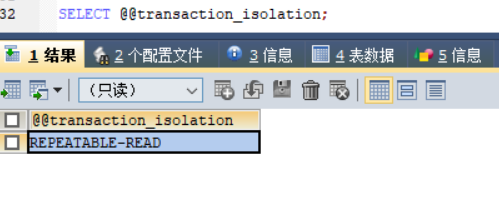
|  |
| --- |
| **RELEASE SAVEPOINT identifie** |



**6.事务隔离机制**

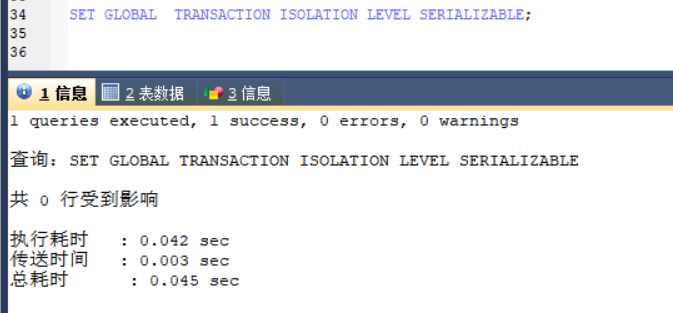
### 1）查看隔离级别

|  |
| --- |
| SELECT @@transaction\_isolation; |



### 2）修改隔离级别

|  |
| --- |
| SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE; |



### 3）隔离机制实验区分

建立两个用户连接，模拟事务A、事务B，对比4种隔离机制。

（1）读未提交——脏读

|  |  |
| --- | --- |
| # 设置隔离机制为 读未提交  SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED ; | |
| # 事务A：读取  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  #读到了事务B未提交的数据，事务B执行回滚后，数据无效  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num; | #事务B：修改订单详情表内容后回滚  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  #对数据做修改，但未提交  update orderitems set quantity = 1 where order\_num=20005 and order\_item=1;  # 事务回滚  ROLLBACK; |

（2）读已提交——不可重复读

|  |  |
| --- | --- |
| # 设置隔离机制为 读已提交  SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED ; | |
| # 事务A：读取  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A再执行查询，没有脏读  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务B提交后，A读到B提交的数据，前后两次读取结果不一致，出现不可重复读现象  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num; | #事务B  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  #对数据做修改，但未提交  update orderitems set quantity = 1 where order\_num=20005 and order\_item=1;  # 事务提交  commit; |

（3）可重复读——幻读

|  |  |
| --- | --- |
| # 设置隔离机制为 可重复读  SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ; | |
| # 事务A：  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A再执行查询，没有脏读  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A修改order\_item=2的记录数据并查询，可修改  update orderitems set quantity = 10 where order\_num=20005 and order\_item=2;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A在事务B提交后查询，数据没有变更，解决不可重复读  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A提交后，执行查询，出现幻读  Commit;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num; | #事务B  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  #对数据先做修改，但未提交  update orderitems set quantity = 1 where order\_num=20005 and order\_item=1;  # 事务B在事务A修改后，查询，没有脏读  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务B提交  commit; |

（4）序列化

|  |  |
| --- | --- |
| # 设置隔离机制为 序列化  SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE; | |
| # 事务A：  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A执行查询，没有脏读  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A修改order\_item=2的记录数据并查询，不能修改，等待  update orderitems set quantity = 10 where order\_num=20005 and order\_item=2;  # 事务A的修改在事务B提交后才执行，强制串行  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务A提交  Commit; | #事务B  Start transaction;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  #事务B对数据先做修改，但未提交  update orderitems set quantity = 3 where order\_num=20005 and order\_item=1;  select order\_num, sum(item\_price\*quantity) from orderitems group by order\_num;  # 事务B提交  commit; |

**7．测试使用事务**

**测试1：创建存储过程实现用户订单的新增，在存储过程中创建事务。**

|  |
| --- |
| BEGIN  DECLARE foundnum INT DEFAULT 0;  DECLARE insertid INT DEFAULT 0;  SELECT \*FROM customers WHERE cust\_id=cust\_id1;  SET foundnum = FOUND\_ROwS();  if foundnum = 1  then  INSERT INTO customers(cust\_id)VALUES (cust\_id1);  END if;  START TRANSACTION;  INSERT into orders(order\_date,cust\_id) VALUES (order\_date,cust\_id);  SET insertid = LAST\_INSERT\_ID();  CALL insertorderitem(insertid,order\_item,prod\_id,quantity);  COMMIT  END |

**测试2：创建存储过程实现销量管理，在存储过程中创建事务。**

|  |
| --- |
| BEGIN  DECLARE productnum INT DEFAULT -1;  START TRANSACTION;  INSERT into ordersitem(order\_num, order\_item, prod\_id,quantity, item\_price)  SELECT order\_num1, order\_item1, prod\_id1, quantity1, prod\_price FROMproducts WHERE products.prod\_id = prod\_id1;  UPDATE products SET stock = stock-quantity1;  SELECT stock INTO productnum FROM products WHERE prod\_id = prod\_id1;  if productnum >= 0  then  COMMIT;  ELSE  ROLLBACK;  END if;  END |

**第二部分：所选课题数据库的应用**

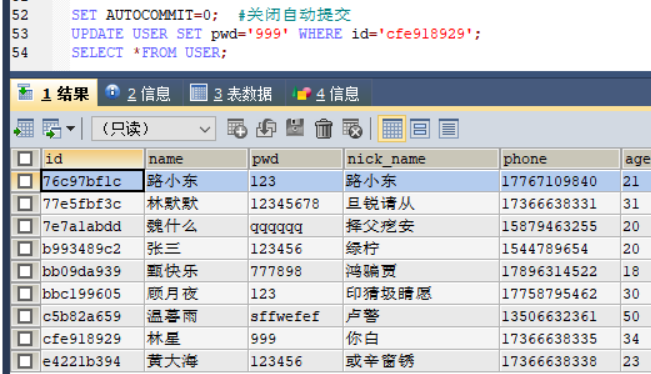
**1．关闭自动提交功能**

**语句：**

|  |
| --- |
| SET AUTOCOMMIT=0; #关闭自动提交  update user set pwd='999' where id='cfe918929';  select \*from user; |

**作用：**自动提交关闭后，从执行结果中发现对应行的数据已经被更新，但由于没有自动提交，结果数据并没有被持久化处理，会出现其他终端查询时数据并没有被修改。

**结果：**



**终端查询：**



**2. 开始事务、结束事务**

**语句：**

|  |
| --- |
| START TRANSACTION; # 事务开始  UPDATE user SET pwd = '12345' WHERE phone = '17366638335';  UPDATE user SET pwd = '12345' WHERE price = '17366638338';  COMMIT; #提交并关闭事务 |

**作用：**显示开启事务，当COMMIT|ROOLBACK语句执行后，表示事务的关闭。

**结果：**



**3. 撤销事务**

**语句：**

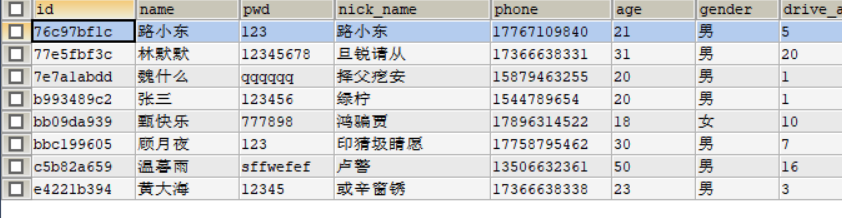
|  |
| --- |
| SELECT \* FROM hotel; #查询显示所有表记录  START TRANSACTION;  DELETE FROM hotel where id=4;  SELECT \* FROM hotel;  ROLLBACK; # 回滚事务  SELECT \* FROM hotel; |

**作用：**使用rollback撤销hotel表删除记录的操作。

**删除前：**



**删除后：**



**回滚后：**



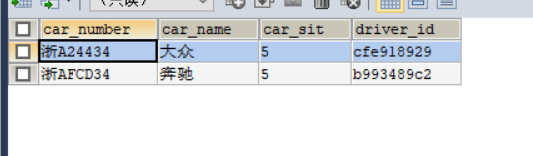
**4. 使用保留点回滚**

**语句：**

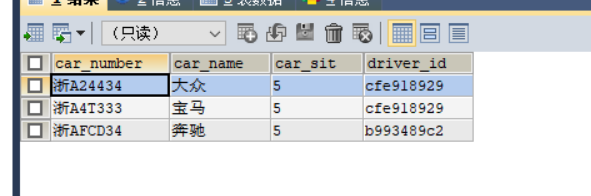
|  |
| --- |
| START TRANSACTION;  Select \* from cars;  Insert into cars values('浙A4T333','宝马','5','cfe918929');  Select \* from cars;  SAVEPOINT beforeInsertcars;  INSERT INTO cars VALUES('浙A4T311','宝马','5','cfe918929');  Select \* from cars;  ROLLBACK to beforeInsertcars;  Select \* from cars;  COMMIT; |

**作用：**使用ROLLBACK TO语句使事务回滚到某个点，在这之前使用SAVEPOINT语句设置保存点。保存点之前的擦插入不会被撤销。

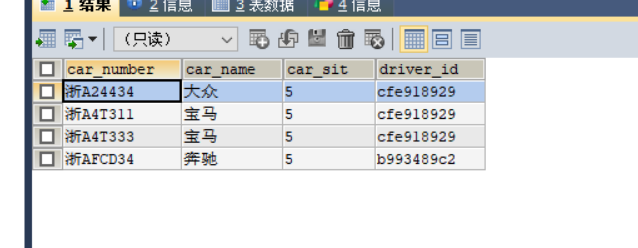
**插入第一条记录前：**



**插入第一条记录后：**



**插入第二条记录后：**



**回滚到第一个检查点：**

