Hands-on Lab: Committing and Rolling Back a Transaction

Estimated time needed: 20 minutes

A transaction is simply a sequence of operations performed using one or more SQL statements as a single logical unit of work. A database transaction must be ACID (Atomic, Consistent, Isolated and Durable). The effects of all the SQL statements in a transaction can either be applied to the database using the COMMIT command or undone from the database using the ROLLBACK command.

In this lab, you will learn some commonly used TCL (Transaction Control Language) commands of SQL through the creation of a stored procedure routine. You will learn about COMMIT, which is used to permanently save the changes done in the transactions in a table, and about ROLLBACK, which is used to undo the transactions that have not been saved in a table. ROLLBACK can only be used to undo the changes in the current unit of work.

Objectives

After completing this lab, you will be able to:

- Permanently save the changes done in a transaction
- Undo the transaction that has not been saved

Software Used in this Lab

MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.

To complete this lab you will utilize MySQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

Mysql_learners database has been used in this lab.

Data Used in this Lab

The data used in this lab is internal data. You will be working on the **BankAccounts** and **ShoeShop** tables.

ACCOUNTNUMBER
B001
B002
B003
B004

This lab requires you to have the **BankAccounts** and **ShoeShop** tables populated with sample data. Download the `BankAccounts-CREATE.sql` and `ShoeShop-CREATE.sql` scripts below, and load them to the phpMyAdmin console. The scripts will create new tables called `BankAccounts` and `ShoeShop` and populate them with the sample data required for this lab.

- BankAccounts-CREATE.sql
- ShoeShop-CREATE.sql

Sample Exercise

Example of committing and rolling back a transaction.

1. **Scenario:** Rose is buying a pair of boots from `ShoeShop`. So we have to update Rose's balance as well as the `ShoeShop` balance in the `BankAccounts` table. Then we also have to update Boots stock in the `ShoeShop` table. After Boots, let's also attempt to buy Rose a pair of Trainers.

- Once the tables are ready, create a stored procedure routine named **TRANSACTION_ROSE** that includes TCL commands like COMMIT and ROLLBACK.
- Now develop the routine based on the given scenario to execute a transaction.
- To create the stored procedure routine on MySQL, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
DELIMITER //
CREATE PROCEDURE TRANSACTION_ROSE()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;
    START TRANSACTION;
    UPDATE BankAccounts
    SET Balance = Balance-200
    WHERE AccountName = 'Rose';
    UPDATE BankAccounts
    SET Balance = Balance+200
    WHERE AccountName = 'Shoe Shop';
    UPDATE ShoeShop
    SET Stock = Stock-1
    WHERE Product = 'Boots';
    UPDATE BankAccounts
    SET Balance = Balance-300
    WHERE AccountName = 'Rose';
    COMMIT;
END //
DELIMITER ;
```

3. Let's now check if the transaction can successfully be committed or not. Copy the code below in a **new blank script** and paste it to the textarea of the **SQL** page. Click **Go**

```
CALL TRANSACTION_ROSE;
SELECT * FROM BankAccounts;
SELECT * FROM ShoeShop;
```

4. Observe that the transaction has been executed. But when we observe the tables, no changes have permanently been saved through COMMIT. All the possible changes happened might have been undone through ROLLBACK since the whole transaction fails due to the failure of a SQL statement or more. Let's go through the possible reason behind the failure of the transaction and how COMMIT - ROLLBACK works on a stored procedure:

    - The first three UPDATEs should run successfully. Both the balance of Rose and ShoeShop should have been updated in the BankAccounts table. The current balance of Rose should stand at 300 - 200 (price of a pair of Boots) = 100. The current balance of ShoeShop should stand at 124,200 + 200 = 124,400. The stock of Boots should also be updated in the ShoeShop table after the successful purchase for Rose, 11 - 1 = 10.

    - The last UPDATE statement tries to buy Rose a pair of Trainers, but her balance becomes insufficient (Current balance of Rose: 100 < Price of Trainers: 300) after buying a pair of Boots. So, the last UPDATE statement fails. Since the whole transaction fails if any of the SQL statements fail, the transaction won't be committed.

Practice exercise

Now let's practice an exercise on committing and rolling back a transaction.

Create a stored procedure **TRANSACTION_JAMES** to execute a transaction based on the following scenario: First buy James 4 pairs of Trainers from ShoeShop. Update his balance as well as the balance of ShoeShop. Also, update the stock of Trainers at ShoeShop. Then attempt to buy James a pair of Brogues from ShoeShop. If any of the UPDATE statements fail, the whole transaction fails. You will roll back the transaction. Commit the transaction only if the whole transaction is successful.

▼ Hint

Use the previous code from Task A Step 2 and modify it. Take 1200 (4 x 300) from James's balance and add 1200 to the ShoeShop balance. Take 4 trainers out of the stock in ShoeShop. Then take 150 from James's balance.

▼ Solution

```
DELIMITER //
CREATE PROCEDURE TRANSACTION_JAMES()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;
    START TRANSACTION;
    UPDATE BankAccounts
    SET Balance = Balance-1200
    WHERE AccountName = 'James';
    UPDATE BankAccounts
    SET Balance = Balance+1200
    WHERE AccountName = 'Shoe Shop';
    UPDATE ShoeShop
    SET Stock = Stock-4
    WHERE Product = 'Trainers';
    UPDATE BankAccounts
    SET Balance = Balance-150
    WHERE AccountName = 'James';
    COMMIT;
END //
DELIMITER ;
```

Conclusion

Congratulations! You have completed this lab, and you are ready for the next topic.

You are now able to:

- Write a stored procedure to record a transaction in multiple tables
- Understand the difference between permanent and rollable transactions
- Perform a transaction
- Perform a rollback

Author(s)

Abhishek Gagneja

Lakshmi Holla

Malika Singla