Hands-on Lab: Committing and Rolling back a Transaction using a Stored Procedure in MySQL using phpMyAdmin

Estimated time needed: 20 minutes

In this lab, you will learn how to create tables and load data in the MySQL database service using the phpMyAdmin graphical user interface (GUI) tool.

Software Used in this Lab

In this lab, you will use MySQL. 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.

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.

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 on Db2. Download the `BankAccounts-CREATE.sql` and `ShoeShop-CREATE.sql`scripts below, upload them to the Db2 console and run them. The scripts will create new tables called BankAccounts and ShoeShop while dropping any previous BankAccounts and ShoeShop tables if they exist, and will populate them with the sample data required for this lab.

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

Please go through the lab below to learn how to upload and run a script on mysal phpadmin console (for this case, you need don't need to know anything else other than how to upload and run a script):

- Hands-on Lab : Create tables using SQL scripts and load data into tables

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

Task A: Example exercise

Let us go through an example on committing and rolling back a transaction

1. Make sure you have created and populated the **BankAccounts** and **ShoeShop** tables by following the **"Data Used in this Lab"** section of this lab.

2.
   - You will create a stored procedure routine named **TRANSACTION_ROSE** which will include TCL commands like COMMIT and ROLLBACK.
   - Now develop the routine based on the given scenario to execute a transaction.
   - **Scenario:** Let's buy Rose a pair of Boots from ShoeShop. So we have to update the Rose 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.
   - To create the stored procedure routine on Db2, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
```

```
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 ;
```

Copied!

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**

```
1
2
3
4
5
```

```
    CALL TRANSACTION_ROSE;

    SELECT * FROM BankAccounts;

    SELECT * FROM ShoeShop;
```

Copied!

4. We can 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 124200 + 200 = 124400. 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.

Task B: Practice exercise

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

1. Problem:

    *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

▶ Solution

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
```

```
 1. DELIMITER //
 2.
 3. CREATE PROCEDURE TRANSACTION_JAMES()
 4.
 5. BEGIN
 6.
 7.
 8.
 9.     DECLARE EXIT HANDLER FOR SQLEXCEPTION
10.     BEGIN
11.         ROLLBACK;
12.         RESIGNAL;
13.     END;
14.         START TRANSACTION;
15.         UPDATE BankAccounts
16.         SET Balance = Balance-1200
17.         WHERE AccountName = 'James';
18.
19.         UPDATE BankAccounts
20.         SET Balance = Balance+1200
21.         WHERE AccountName = 'Shoe Shop';
22.
23.         UPDATE ShoeShop
24.         SET Stock = Stock-4
25.         WHERE Product = 'Trainers';
26.
27.         UPDATE BankAccounts
28.         SET Balance = Balance-150
29.         WHERE AccountName = 'James';
30.
31.
32.
33.             COMMIT;
34.
35.
36.
37. END //
38.
39. DELIMITER ;
```

Copied!

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

Author(s)

Lakshmi Holla

Malika Singla

Changelog

Date	Version	Changed by	Change Description
2023-05-10	0.3	Eric Hao & Vladislav Boyko	Updated Page Frames
2023-05-04	0.2	Rahul Jaideep	Updated Markdown file
2021-11-01	0.1	Lakshmi Holla, Malika Singla	Initial Version