Committing and Rolling back a Transaction using a Stored Procedure

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.

We 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.

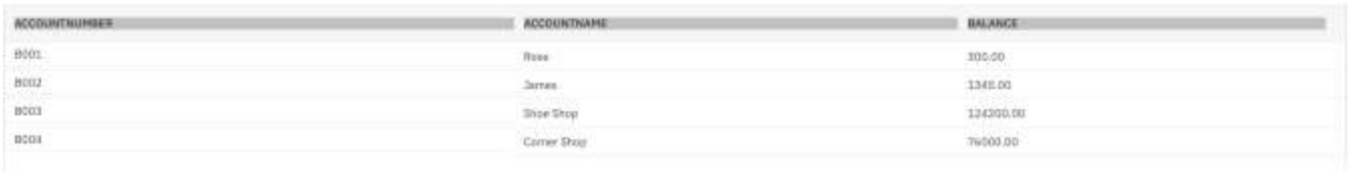
Software Used in this Lab

In this lab, you will use an [IBM Db2 Database](). Db2 is a Relational Database Management System (RDBMS) from IBM, designed to store, analyze and retrieve data efficiently.

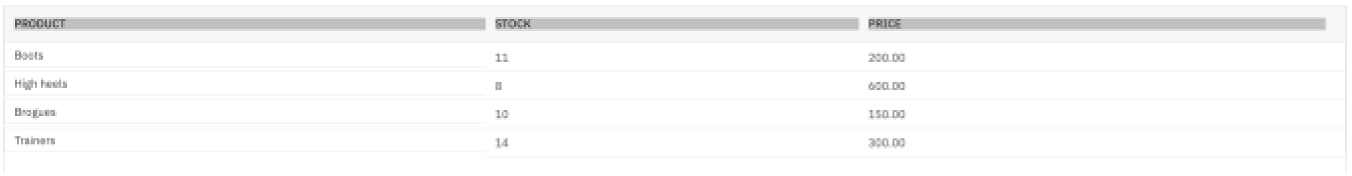
To complete this lab you will utilize a Db2 database service on IBM Cloud.

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	ACCOUNTNAME	BALANCE
B001	Rose	300.00
B002	James	1345.00
B003	Shoe Shop	124200.00
B004	Corner Shop	76000.00



PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

This lab requires you to have the **BankAccounts** and **ShoeShop** tables populated with sample data on Db2. Download the `bank_accounts_create.sql` and `shoe_hop_create.sql` provided in this module, 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.

Objectives

- Permanently saving the changes done in a transaction
- Undoing the transaction that has not been saved

Instructions

When you approach the exercises in this lab, follow the instructions to run the queries on Db2:

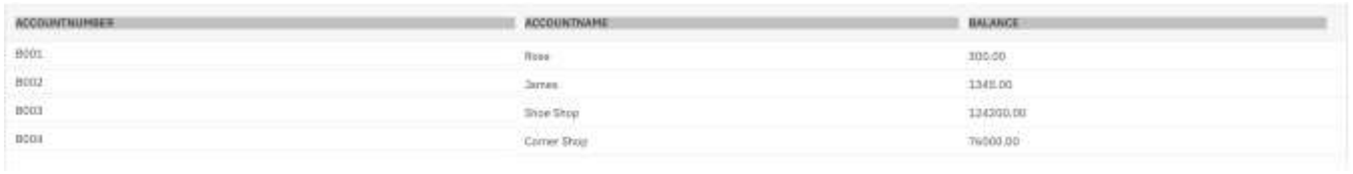
- Go to the Resource List of IBM Cloud by logging in where you can find the Db2 service instance that you created in a previous lab under **Services** section. Click on the **Db2-xx service**. Next, open the Db2 Console by clicking on **Open Console** button. Click on the 3-bar menu icon in the top left corner and go to the **Run SQL** page. The Run SQL tool enables you to run SQL statements.

Exercise

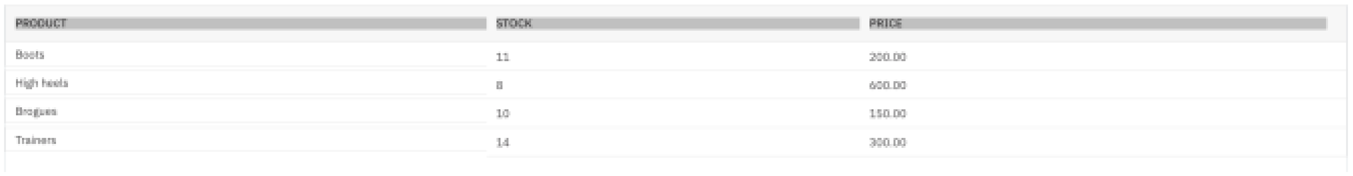
Task A: Example exercise

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.



ACCOUNTNUMBER	ACCOUNTNAME	BALANCE
B001	Rose	300.00
B002	James	1345.00
B003	Shoe Shop	124200.00
B004	Corner Shop	76000.00



PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

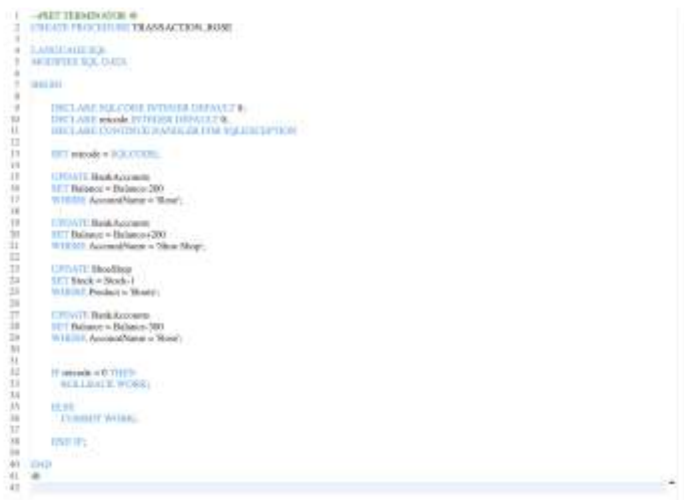
2. You will create a stored procedure routine named **TRANSACTION_ROSE** which will include TCL commands like COMMIT and ROLLBACK.

   Now develop the routine based on the given scenario to execute a transaction.

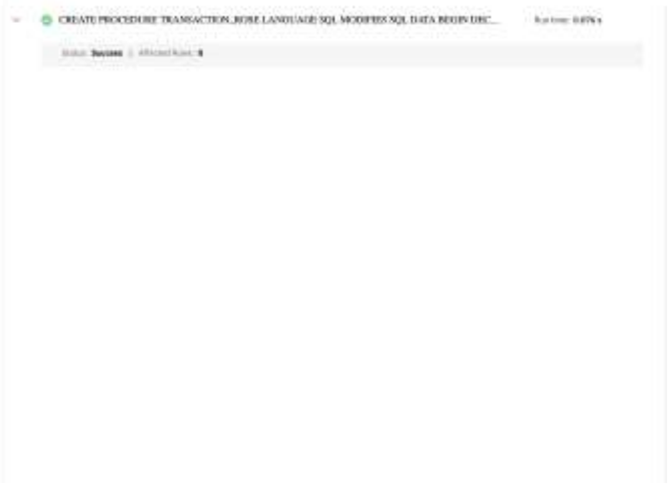
   Scenario: Let's buy Rose a pair of Boots from ShoeShop. So we have to update the Rose balance as well as the ShoeShop balance in the BankAccounts table. Then we also have to update Boots stock in the ShoeShop table. After Boots, let's also attempt to buy Rose a pair of Trainers.

   To create the stored procedure routine on Db2, copy the code below and paste it to the textbox of the **Run SQL** page. Click **Run all**.

```
--#SET TERMINATOR @
CREATE PROCEDURE TRANSACTION_ROSE                          -- Name of this
stored procedure routine
LANGUAGE SQL                                               -- Language used
in this routine
MODIFIES SQL DATA                                          -- This routine
will only write/modify data in the table
BEGIN
        DECLARE SQLCODE INTEGER DEFAULT 0;                 -- Host variable
SQLCODE declared and assigned 0
        DECLARE retcode INTEGER DEFAULT 0;                 -- Local variable
retcode with declared and assigned 0
        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION          -- Handler tell
the routine what to do when an error or warning occurs
        SET retcode = SQLCODE;                             -- Value of
SQLCODE assigned to local variable retcode
        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';
        IF retcode < 0 THEN                                 --  SQLCODE
returns negative value for error, zero for success, positive value for warning
            ROLLBACK WORK;
        ELSE
            COMMIT WORK;
        END IF;
END
@                                                           -- Routine
termination character
```



```
--#SET TERMINATOR @
CREATE PROCEDURE TRANSACTION_ROSE

LANGUAGE SQL
MODIFIES SQL DATA

BEGIN

    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE retcode INTEGER DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

    SET retcode = SQLCODE;

    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';


    IF retcode < 0 THEN
      ROLLBACK WORK;

    ELSE
      COMMIT WORK;

    END IF;

END
@
```

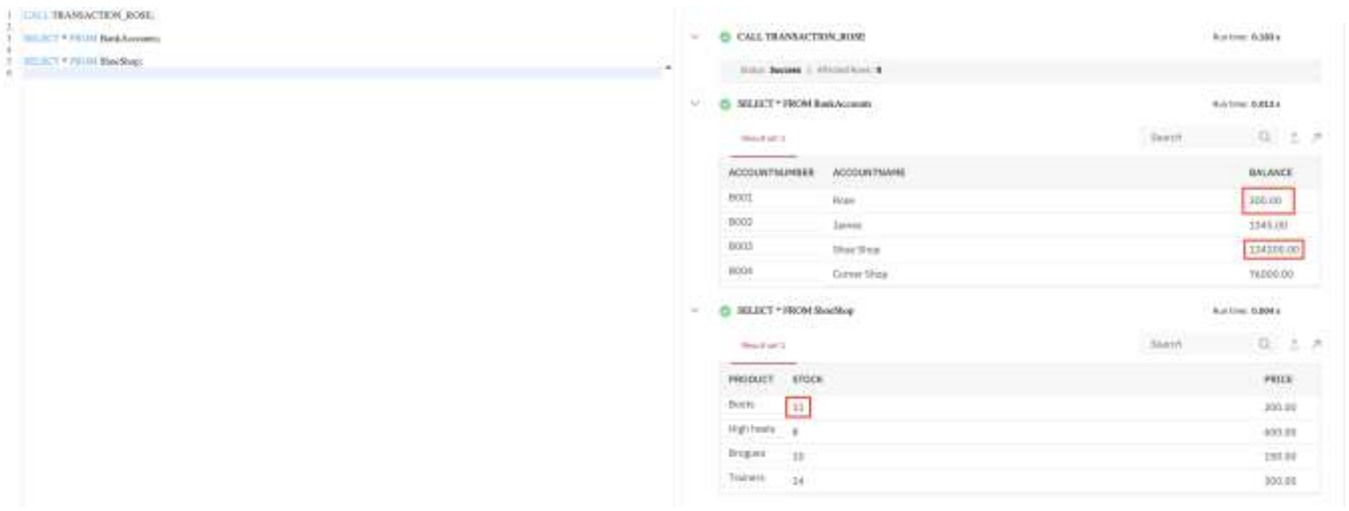


```
CREATE PROCEDURE TRANSACTION_ROSE LANGUAGE SQL MODIFIES SQL DATA BEGIN DEC...    Run time: 0.076 s
Status: Success | Affected Rows: 0
```

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 textbox of the **Run SQL** page. Click **Run all**.

```
CALL TRANSACTION_ROSE;  -- Caller query
SELECT * FROM BankAccounts;
SELECT * FROM ShoeShop;
```

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.
    - The **SQLCODE** which is a stand-alone host variable contains success/failure/warning information of each SQL statement execution. Now since **SQLCODE** variable gets reset back as the next SQL statement runs, **retcode** is our local variable to catch the return value of this **SQLCODE**. **SQLCODE** returns negative value for each SQL statement if not executed successfully. So, on any error occurrence, all the changes are rolled back. Commit only takes place after the transaction gets executed successfully without any error.



```
CALL TRANSACTION_ROSE;

SELECT * FROM BankAccounts;

SELECT * FROM ShoeShop;
```

CALL TRANSACTION_ROSE Run time: 0.380 s

Status: Success | Affected Rows: 1

SELECT * FROM BankAccounts Run time: 0.013 s

ACCOUNTNUMBER	ACCOUNTNAME	BALANCE
B001	Rose	100.00
B002	James	1345.00
B003	Shoe Shop	124200.00
B004	Corner Shop	76000.00

SELECT * FROM ShoeShop Run time: 0.004 s

PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

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.*

Solution:

```
--#SET TERMINATOR @
CREATE PROCEDURE TRANSACTION_JAMES                          -- Name of this
stored procedure routine
LANGUAGE SQL                                                -- Language used
in this routine
MODIFIES SQL DATA                                           -- This routine
will only write/modify data in the table
BEGIN
        DECLARE SQLCODE INTEGER DEFAULT 0;                  -- Host variable
SQLCODE declared and assigned 0
        DECLARE retcode INTEGER DEFAULT 0;                  -- Local variable
retcode with declared and assigned 0
        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION           -- Handler tell
the routine what to do when an error or warning occurs
        SET retcode = SQLCODE;                              -- Value of
SQLCODE assigned to local variable retcode
        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';
        IF retcode < 0 THEN                                  --  SQLCODE
returns negative value for error, zero for success, positive value for warning
            ROLLBACK WORK;
        ELSE
            COMMIT WORK;
        END IF;
END
@                                                            -- Routine
termination character
```