**TRANSACTIONS**

Transaction is a sequence of SQL statements processed / executed as an atomic unit. All statements in the transaction are either committed (saved to the database) or rolled back (cancelled).

**COMMIT**: TO SAVE TRANSACTIONS FROM DATABASE LOG FILE TO DATA FILE.

**ROLLBACK**: TO UNDO or CANCEL A TRANSACTION.

EVERY TRANSACTION SHOULD BE EITHER COMMITTED OR ROLLED BACK.

These transactions are used enforce **ACID** properties:

**Atomicity** - Every Transaction is a Single Unit. Complete operation Or Nothing at all.

**Consistency** - Success Or Failure of Transaction will not impact Database State.

**Isolation** - One Transaction can be dependent on other Transactions.

**Durability** - Transactions once Committed, the data is permanent.

**Types of Transactions in SQL Server:**

1. Auto Commit Transactions : Auto Start, Auto End

2. Explicit Transactions : Manual Start, Manual End

3. Implicit Transactions : Auto Start, Manual End

**COMPARING TYPES OF TRANSACTIONS**:

**AUTO COMMIT EXPLICIT COMMIT IMPLICIT COMMIT**

----------------------------------------------------------------------------------------------------------------------------

BEGIN TRANSACTION SET IMPLICIT\_TRANSACTIONS ON

STATEMENT 1 STATEMENT 1 STATEMENT 1

STATEMENT 2 STATEMENT 2 STATEMENT 2

COMMIT | ROLLBACK COMMIT | ROLLBACK

**PRACTICE EXAMPLES**

-- ITEM 1: EXAMPLES FOR AUTOCOMMIT TRANSACTIONS : MEANS AUTO SAVE.

CREATE DATABASE bankdb

use bankdb

create table tblAccounts -- Either complete table is created or nothing at all.

(

AccountID int,

Accbal float,

AccStatus int

)

INSERT INTO tblAccounts VALUES ('10001', 6969621, 1) -- THIS IS ONE TRANSACTION

INSERT INTO tblAccounts VALUES ('10002', 2346372, 1) -- THIS IS ONE TRANSACTION

-- ITEM 2: EXAMPLES FOR EXPLICIT TRANSACTIONS: MANUAL START, MANUAL END

BEGIN TRANSACTION T1 -- THIS STATEMENT IS USED TO START THE TRANSACTION

INSERT INTO tblAccounts VALUES ('10003', 6969621, 1)

INSERT INTO tblAccounts VALUES ('10004', 2346372, 1)

DECLARE @RWCOUNT\_VARIABLE INT -- VARIABLE : A TEMPORARY PLACEHOLDER FOR DATA IN MEMORY.

SELECT @RWCOUNT\_VARIABLE = COUNT(\*) FROM tblAccounts -- THIS IS TO ASSIGN THE ROW COUNT TO A VARIABLE

IF @RWCOUNT\_VARIABLE <= 100

COMMIT -- TO END THE TRANSACTION: SAVE THE DATA CHANGES

ELSE

ROLLBACK -- TO END THE TRANSACTION: UNDO THE DATA CHANGES.

-- ITEM #3: OPEN TRANSACTIONS : SUCH TRANSACTIONS THAT ARE STARTED BUT NOT COMMTTED OR ROLLEDBACK

BEGIN TRANSACTION T2

INSERT INTO tblAccounts VALUES ('20001', 6969621, 1)

INSERT INTO tblAccounts VALUES ('20002', 2346372, 1)

/\*

TO TEST THE OPEN TRANSACTION IMPACT:

FROM SSMS : GO TO FILE > NEW > QUERY WITH CURRENT CONNECTION > RUN BELOW QUERY:

SELECT \* FROM tblAccounts -- THIS QUERY RUNS FORVER. MEANS, QUERY IS BLOCKED. REASON : OPEN TRANSACTIONS.

-- GO TO TOP : QUERY MENU > CANCEL.

-- TO AVOID QUERY BLOCKING: USE "LOCK HINTS"

SELECT \* FROM tblAccounts WITH (NOLOCK) -- NO QUERY BLOCKING BUT UNCOMITTED DATA / BAD DATA / DIRTY READS ARE SHOWN

SELECT \* FROM tblAccounts WITH (READPAST) -- NO QUERY BLOCKING, NO DIRTY READS

\*/

-- TO RESOLVE QUERY BLOCKING:

COMMIT

/\*

REAL-WORLD USE OF TRANSACTIONS: CONSIDER A BANKING ENVIRONMENT. DEBITS FROM ONE ACCOUNT, CREDITS TO ANOTHER ACCOUNT.

IF CREDIT FAILS, THEN DEBIT SHOULD BE ROLLEDBACK. THIS IS POSSIBLE ONLY BY IMPLEMNTING EXPLICIT TRANSACTIONS MECHANISM.

\*/

/\*

WHEN TO USE AUTOCOMMIT TRANSACTIONS:

1. FOR DATA RETREIVAL (SELECT)

2. FOR DDL OPERATIONS (CREATE, ALTER, DROP)

3. FOR UNCONDITIONAL DML OPERATIONS [EX: DATA IMPORT]

WHEN TO USE EXPLICIT / IMPLICIT TRANSACTIONS:

1. CONDITIONAL DML OPERATIONS (INSERT, UPDATE, DELETE)

2. COMPLEX CODING SCENERIOS : PAYMENT GATEWAYS, ONLINE RESERVATIONS...

\*/