**Transaction**

Transaction like a Recipe (khana pkana wala). Recipe has a series of steps that he needs to follow in order to make a dish.

**Example :** A transaction is like a bundle of actions that need to be done together to make sure everything is accurate and consistent.

For example, imagine you're playing a game with your friend and you both need to make a move at the same time to complete the Run.

This is similar to a transaction, where multiple actions need to be completed together to make sure the result is accurate.

**Transaction Types**

1. **Implicit Transactions (isharatan) Inser,update,delete**

These are transactions that are automatically started and committed by the database system without any explicit command from the user.

🡪 Wo kaam jo bagaar Commit, rollback hon -🡪 Implicit Transaction

create table customer(

id int primary key identity,

name varchar(25),

payment int

)

---\_\_\_\_\_\_ 1. Implicit Transaction (Insert, Update, Delete) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

--1. insert --

insert into customer values ('sajid',4000)

--2. update ---

update customer set payment = 50000 where id = 1

--3. delete--

delete customer where id = 1

UPDATE Products SET Price = 10.99 WHERE ProductID = 1

1. **Explicit Transactions (btana prtaa ha)**

These are transactions that are explicitly started and committed by the user using the

BEGIN Transaction,

COMMIT Transaction,

ROLLBACK Transaction.

**Its very Important in Any language.**

***Let* : inserting 10 Rows in Table In 1 time , if at 9th row Errors Occurs . So 9th 10th record will missed out.**

***What We Do :*** put our Code inside **(Begin Trans)** (**commit Trans**) **(rollback Trans)**

Steps:

1. Transaction Begin
2. If no Error occurs -🡪 Commit transactio Applies
3. Else Error Occurs 🡪 RollBack Tans Applies

***Example 1 :* commit , rollback**

create table customer(

id int primary key identity,

name varchar(25),

payment int

)

insert into customer values ('sajid',9000),('sajid',8000),('sajid',10000)

select \* from customer

---\_\_\_\_\_\_\_\_ 1. If No Error \_\_ we use commit ----

Begin Transaction

update customer set payment = (payment+1000) where id = 3

Commit Transaction

print 'payment updatedSuccessfuly'

---\_\_\_\_\_\_\_\_ 2. If Error Occur \_\_ we use rollback ---

Begin Transaction

update customer set payment = (payment+1000) where id = 3 -- 5 id not excit in table

--select 1/0

ROLLBACK Transaction;

print 'Transaction RollBack : System hevy load please try latter'

select \* from customer

---\_\_\_\_\_\_\_\_ 3. ---

BEGIN TRANSACTION

UPDATE Customer SET Payment = Payment + 1000 WHERE Id = 2

--\_\_\_\_\_ will give Update Record \_\_\_

select \* from customer

WAITFOR DELAY '00:00:03' -- wait for 1 minute

ROLLBACK TRANSACTION

--\_\_\_\_ after rollback \_\_ old data\_\_\_\_

select \* from customer

***Example 2 :* Try,Catch**

---\_\_\_\_\_\_ 3. TryCatch (commit , rollback) \_\_\_\_\_\_ ----

-- commit Rollback are used with tryCatch

Begin Try

Begin Transaction

update customer set payment = (payment+1000) where id = 3

-- Error --

select 1/0;

commit Transaction;

end try

begin catch

ROLLBACK Transaction;

print 'Transaction RollBack : System hevy load please try latter'

end catch

\_\_\_ check Results \_\_

select \* from customer

1. **Distributed Transactions**

Transactions that involve multiple databases.

**For example:** if you need to transfer money from one bank to another, this involves multiple databases, and a distributed transaction is needed to ensure that the transaction is completed accurately and consistently.

**This Transaction work with** 🡪 transaction coordinator, such as Microsoft Distributed Transaction Coordinator (MSDTC)

**Simple Words :** transaction b/w 2, more database is known as Distributed Transaction.

***Example 11 : (database to database ||server to server )* FQDN**

***We Don’t have 2 Server so 🡪 We will Do (database to database)***

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3. Distributed Transaction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-- don't have 2 Server so ---> do with (database to database) fqdn

insert into practice1.xyz.customer values ('sajid')

insert into practice1.xyz.customer values ('Akram')

insert into practice1.xyz.customer values ('Noman')

use master

select \* from practice1.xyz.customer

select \* from practice2.dbo.orders

--\_\_\_\_\_\_\_\_\_\_\_\_ Distributed Transaction \_\_\_\_\_\_\_\_\_\_\_\_\_

BEGIN DISTRIBUTED TRANSACTION;

Begin try

BEGIN TRANSACTION

update practice1.xyz.customer set name = 'Pakistan' where id = 1;

--select \* from practice1.xyz.customer

update practice2.dbo.Orders set OrderAmount = 11111 where OrderID = 1;

select 1/0;

--select \* from practice2.dbo.Employees

commit transaction;

end try

begin catch

Rollback Transaction;

--- Raised Message ---

print Error\_Message();

end catch

**Example 2**

begin try

Begin Transaction

update customer set payment = (payment+1000) where id = 3

select 1/0;

commit transaction;

end try

begin catch

rollback transaction;

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = 'Error Number : ' + cast( Error\_number() as varchar ) + CHAR(13) + CHAR(10) +

'Error Message : ' +cast( Error\_message() as varchar ) + CHAR(13) + CHAR(10) +

'Error Line : ' + cast( Error\_Line() as varchar ) + CHAR(13) + CHAR(10) +

'Error state : ' + cast( Error\_state() as varchar ),

@ErrorSeverity = 16,

@ErrorState = 1;

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

end catch