Stored Procedures and Transactions in SQL Server

1. Stored Procedures

A **Stored Procedure** is a precompiled set of SQL statements stored within a database that can be executed multiple times. It acts as a **modular program inside the database**, allowing users to encapsulate complex logic and reuse it across different applications or tasks.

Key Features and Benefits

1. **Reusability:**
   a. Stored Procedures allow developers to write SQL logic once and execute it multiple times with different parameters, reducing repetition.
2. **Performance Improvement:**
   a. SQL Server caches the execution plan of a procedure, which speeds up repeated executions compared to dynamic queries.
3. **Security:**
   a. Users can execute procedures without direct access to underlying tables, protecting sensitive data.
4. **Complex Logic Management:**
   a. Procedures can include conditions (IF...ELSE), loops (WHILE), and multiple SQL statements, making them ideal for handling complex business logic.

Example

```
CREATE PROCEDURE GetCustomerAccounts

   @CustomerID INT

AS

BEGIN

   SELECT AccountID, AccountType, Balance, Status

   FROM Account

   WHERE CustomerID = @CustomerID;
```

END;

Usage

EXEC GetCustomerAccounts @CustomerID = 1;

Explanation:

The parameter @CustomerID allows the procedure to be dynamic and reusable for any customer.

This approach reduces redundancy, improves maintainability, and centralizes business logic.

2. Transactions

A Transaction in SQL Server is a sequence of one or more SQL statements executed as a single unit of work. Transactions ensure data consistency and integrity by enforcing the principle of atomicity: either all operations succeed (COMMIT) or none take effect (ROLLBACK).

Key Benefits

Data Integrity:

Ensures that related operations either complete fully or not at all.

Error Recovery:

If an error occurs during execution, changes can be rolled back to maintain a consistent database state.

Concurrent Access Management:

Helps manage multiple users accessing the same data simultaneously without causing inconsistencies.

Basic Transaction Commands

BEGIN TRANSACTION → starts a new transaction.

COMMIT → saves all changes if successful.

ROLLBACK → cancels all changes if an error occurs.

Example: Funds Transfer Between Accounts

sql

```
BEGIN TRANSACTION;

BEGIN TRY
   -- Deduct amount from source account
   UPDATE Account
   SET Balance = Balance - 1000
   WHERE AccountID = 101;

   -- Add amount to target account
   UPDATE Account
   SET Balance = Balance + 1000
   WHERE AccountID = 102;
```

```
    COMMIT; – Save changes if all updates succeed
END TRY
BEGIN CATCH
    ROLLBACK; – Undo changes if an error occurs
    PRINT 'Transaction failed. Changes rolled back.';
END CATCH;
```

Explanation:

Both updates are treated as a single unit of work.

If any operation fails, the database rolls back all changes, ensuring that accounts remain consistent.

Conclusion

- Stored Procedures and Transactions are essential tools in SQL Server for:
- Simplifying complex operations (Procedures)
- Ensuring data consistency and integrity (Transactions)
- In real-life applications like banking systems, they are used to:
- Encapsulate business logic (e.g., fetching customer accounts, processing payments).
- Handle critical operations safely (e.g., fund transfers) without risking data corruption.