

- **TRANSACTION**

-- SQL Server Transactions --

Definition

- A **Transaction** is a sequence of one or more SQL operations (INSERT, UPDATE, DELETE) executed as a **single unit**.
- Ensures that either **all operations succeed** or **none are applied**.

2. ACID Properties

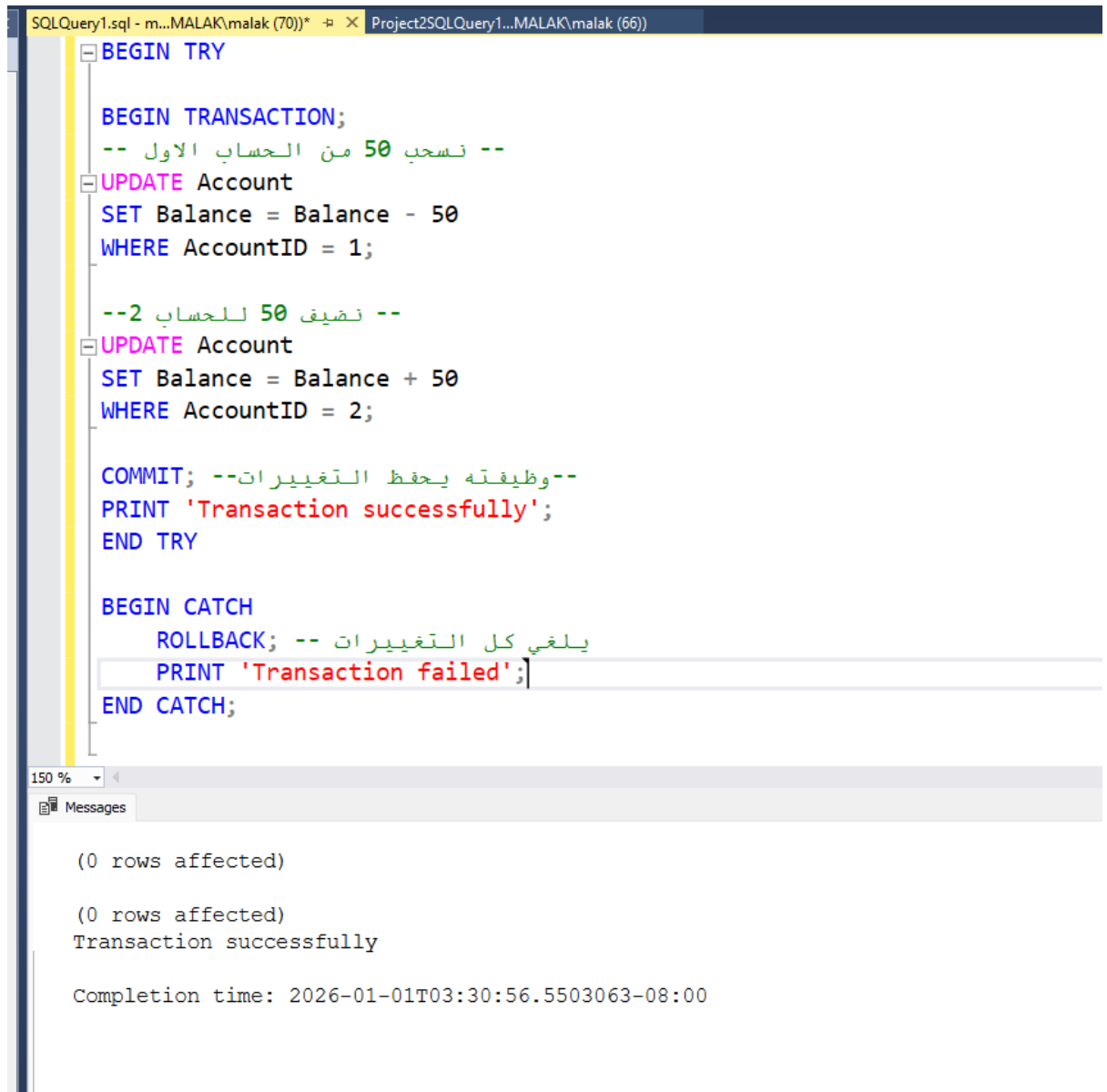
1. **Atomicity** – All operations in the transaction are completed, or none are applied.
2. **Consistency** – Database remains in a valid and consistent state.
3. **Isolation** – Concurrent transactions do not affect each other.
4. **Durability** – Once committed, changes are permanent even if the system fails.

. Benefits

- Maintains **data integrity** during complex operations.
- Prevents **partial updates** that could corrupt data.
- Essential in critical systems like **banking, e-commerce, and payroll**.

. Example Use Case (Banking)

- **Scenario:** نريد نأخذ من حساب الشخص الأول ونعطيها الحساب 2



The screenshot displays a SQL Server Enterprise Manager window with a T-SQL script editor and a Messages pane. The script is a transaction that updates the balance of two accounts. The Messages pane shows the execution results, including the number of rows affected and the completion time.

```
SQLQuery1.sql - m...MALAK\malak (70)) * X Project2SQLQuery1...MALAK\malak (66)

BEGIN TRY

    BEGIN TRANSACTION;

    -- نسحب 50 من الحساب الاول --
    UPDATE Account
    SET Balance = Balance - 50
    WHERE AccountID = 1;

    -- نضيف 50 للحساب 2 --
    UPDATE Account
    SET Balance = Balance + 50
    WHERE AccountID = 2;

    COMMIT; -- وظيفته يحفظ التغييرات --
    PRINT 'Transaction successfully';
END TRY

BEGIN CATCH
    ROLLBACK; -- يلغي كل التغييرات --
    PRINT 'Transaction failed';
END CATCH;
```

150 %
Messages

(0 rows affected)

(0 rows affected)
Transaction successfully

Completion time: 2026-01-01T03:30:56.5503063-08:00

- # If an error occurs, you can **ROLLBACK** to cancel all changes.

. Notes

- Transactions **do not speed up queries**, but **ensure safe and reliable updates**.
- Can be used with **updatable Views** to modify underlying tables safely.
- Often used in operations that must be **all-or-nothing**, like fund transfers, loan processing, or inventory updates.

- **PROCEDURE**

-- Stored Procedures in SQL Server--

What is a Stored Procedure?

A **Stored Procedure** is a group of SQL statements that is saved inside the database and executed as a single unit.

Instead of writing the same SQL queries repeatedly, we can store them once and reuse them by calling the procedure.

Why Do We Use Stored Procedures?

Stored Procedures are used to:

- Simplify complex SQL logic
- Reduce repeated SQL code
- Improve performance
- Increase security
- Ensure data consistency

Stored Procedures and Transactions

Stored Procedures are often used with **Transactions** to ensure data integrity.

A transaction guarantees that:

- All operations are completed successfully (**COMMIT**), or
- None of them are applied if an error occurs (**ROLLBACK**)

This is very important in systems like banking, where money transfers must be accurate.

Real-Life Example (Banking System)

In a banking system, a stored procedure can be used to:

- Transfer money between two accounts
- Deduct an amount from one account
- Add the same amount to another account

If any step fails, the transaction is rolled back to prevent data inconsistency.

Advantages of Stored Procedures

- **Better Performance:** Stored procedures are precompiled, so they execute faster.
- **Security:** Users can be given permission to execute a procedure without accessing tables directly.
- **Maintainability:** Business logic is stored in one place and easy to update.
- **Reusability:** The same procedure can be reused by different applications.

Limitations of Stored Procedures

- Harder to debug compared to normal SQL queries
- Changes require altering the procedure
- Logic is tied to the database (less flexibility in some applications)

```

Project2SQLQuery1...MALAK\malak (51)
--16. sp_IssueBook--
CREATE PROCEDURE sp_IssueBook
@MemberID INT,
@BookID INT,
@DueDate DATE
AS
BEGIN
SET NOCOUNT ON;
BEGIN TRY
--بنتحقق اذا الكتاب موجود--
IF NOT EXISTS (SELECT 1 FROM Book WHERE Book_ID = @BookID AND IsAvailable = 1)
BEGIN
PRINT 'Error: This book is not available.';
RETURN;
END
--بنتحقق اذا العضو عنده اي قروض متاخرة--
IF EXISTS (SELECT 1 FROM Loan WHERE Member_ID = @MemberID AND Status = 'Overdue')
BEGIN
PRINT 'Error: Member has overdue loans. Cannot issue new book.';
RETURN;
END
--اذا تحققت الشرطين سويلي سجل قرض جديد--
INSERT INTO Loan (Loan_Date, Due_Date, Status, Member_ID, Book_ID)
VALUES (GETDATE(), @DueDate, 'Issued', @MemberID, @BookID);
--نحدث حالة الكتاب الجين--
UPDATE Book
SET IsAvailable = 0
WHERE Book_ID = @BookID;
--اذا نجح اعرض الرسالة ذي --
PRINT 'Success: Book has been issued to the member.';
END TRY
BEGIN CATCH
--التعامل مع أي خطأ--
PRINT 'Error: Something went wrong.';
PRINT ERROR_MESSAGE();
END CATCH
END;
EXEC sp_IssueBook @MemberID = 1, @BookID = 2, @DueDate = '2026-01-15';

```

```

EXEC sp_IssueBook @MemberID = 1, @BookID = 2, @DueDate = '2026-01-15';
150 %
Messages
Error: This book is not available.

Completion time: 2026-01-01T03:34:20.3416531-08:00

```

Conclusion

Stored Procedures are a powerful feature in SQL Server.

They help manage complex operations, improve performance, and protect data integrity, especially in critical systems such as banking and financial applications.