Day 9

*** Stored Procedure & Triggers :**

- فى تنفيذ كل Query بيمر بمجموعة من الـ Steps

Query → Parsing → Optimizing → Query Tree → Execution Plan

- الـ Parsing : بيتأكد من صلاحية الـ Syntax

- الـ Optimizing : بيتأكد من الـ Metadata والـ Db objects

- الـ Query Tree: بتوضح الـ execution order لـ Query

- الـ Execution Plan: Execute query + Transfer data to Memory

- لو عملت run تانى لنفس الـ query هيعمل نفس الخطوات من الأول تانى (Execution جديد)

(لو قدرت أحسن النقطة دى فأنا بحسن فى الـ performance)

- الحل اللى وفروه لحل المشكلة دى هو الـ stored Procedure

(الـ query اللى هيتنفذ كتير أو يكون complicated أعمله فى SP)

Stored Procedure :

- عبارة عن Precompiled collection of one or more Queries

- بيتم تخزينها فى الـ SQL Server

- الفوائد من الـ Stored procedure :

(1- الـ Code Reusability : أستخدم الـ query أكتر من مرة بدون ما أكتب الـ query تانى

2- الـ performance : لأن الـ SP عبارة عن Precompiled فبالتالى مش هحتاج الـ parsing والـ Optimizing

3- الـ Security : ممكن تتحكم فى الـ Execution Permission لـ SP

4- الـ Modularity : بتـ Break down الـ Complicated Task لـ Smaller task)

- SP Syntax :

```
CREATE PROCEDURE Procedure_Name @Parameter DataType , …
AS
BEGIN
    -- Your SQL statements go here
END;
```

- SP Invoking :

```
Procedure_Name ()
```

Or

```
Execute Procedure_Name ()
```

- الـ SP جواها كل أنواع الـ Queries (على عكس الـ Function & View)

- بخفى الـ Behavior على الـ Network (Security)

- بخفى الـ Business Rule عن الـ Network (Security)

- بخفى الـ DB Objects (Security)

- عدد الـ Character أقل تماماً (بالتالى Performance أعلى)

- أهم شئ هو الـ Performance Improvement :

(مع الـ First Call بيبدأ يعوض عن الـ Parameter وبعدها الـ Parsing ومن ثم الـ Optimizing ومن ثم الـ Query Tree و يحفظ الـ Tree فى الـ SQL و بعد كدا Execution Plan

الـ Call لنفس الـ SP تانى : هيبدأ من الـ Saved Query Tree ويعملها Execution)

- الـ SP موجودة فى الـ Programmability Folder

*** SP Types :**

1) Built In SP

2) User Defined SP

3) Triggers (Special Type of User Defined SP)

- لو عاوز أخد Multiple Values من SP:

هستخدم Insert based on Execute

```
Insert into Table_Name
Execute Procedure_Name
```

(نفس فكرة الـ Insert based on select)

- لو هاخد Single Value بس من SP :

هستخدم Variable

```
`set @X = Execute Procedure_Name
```

أ و هستخدم الـ Output Keyword فى الـ SP Parameter

```
create procedure getage @id int , @age int output
as
    select @age = St_Age
    from Student
    where St_Id = @id

declare @x int
execute getage 4 , @x output
select @x
```

*** Return Function Vs Return SP :**

- فى الـ Function :

الـ return بترجع Table \ Value

الـ Return is mandatory

- فى الـ SP :

الـ return بترجع One value ونوعها Int بس (أى حاجة تانية هيديك ERROR)

الـ Int value دى هيا Indictor ما بين الـ DB Developer و الـ APP Developer بتعبر عن الـ SP Behavior (الـ Return is optional)

*** Triggers :**

- عبارة عن Special Type of SP

- Can’t Call

- Doesn’t have Parameters

- عبارة عن Implicit code موجود جوا الـ server بيـ run \ fire automatically لما بيحصل Event معين

- Trigger levels :

1) On Server Level

2) On Database Level

3) On Table Level

*** Trigger types :**

1) DML Triggers :

- After \ for

- Instead of

2) DDL Triggers

*** الـ DML Triggers :**

```
CREATE TRIGGER Trigger_Name
ON Table_Name
AFTER\Instead of Event(Insert\Update\Delete)
AS
BEGIN
    -- Trigger logic here
END;
```

- الـ Trigger هيـ Fire سواء الـ Query أثر فى الـ Table أو لا

- الـ Trigger بيورث إسم الـ Schema بتاعت الـ Object

- الـ Update ممكن أستخدمها فى IF (بتـ return الـ Column اتعمله update ولا لا "T \ F")

- ممكن أعمل Drop لـ Trigger

- ممكن أعمله Disable \ Enable بدل ما أمسحه كـ object (بدل الـ Drop)

```
Drop Trigger Trigger_Name

Alter table Table_Name Disable\Enable Trigger Trigger_Name
```

---

* الـ DDL Triggers :

- عبارة عن Triggers بتـ respone لـ events اللى على Database \ Server Level خاصة الـ Events لـ Creation \ Alteration \ Deletion لـ Database Objects

```
-- Create a test database
CREATE DATABASE TestDB;
GO

-- Create a table in the test database
USE TestDB;
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50)
);
GO

-- Create a DDL trigger that captures CREATE_TABLE events
CREATE TRIGGER DDLTrigger_CreateTable
ON DATABASE
FOR CREATE_TABLE
AS
BEGIN
    PRINT 'A new table was created in the database.';
END;
GO
-- Test the trigger by creating another table
USE TestDB;
CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
```

```
    DepartmentName NVARCHAR(50)
);
```

*** Triggers for Auditing :**

- الفايدة الأمثل من الـ Triggers

(بتـ store معلومات عن الـ query)

- عندى Two Tables فى الـ Temp DB ولكن إستخدامهم وبيظهروا فى الحالة دى بس وهما :

Inserted Table

Deleted Table

(لو عملت select * from Inserted \ Deleted هيظهر Error إن مفيش Object إسمه كدا)

(إلا لو عملت الـ Query جوا Trigger مش هيدى Error)

- الـ Inserted Table فيه الـ record بعد Event

- الـ Deleted Table فيه الـ Records قبل Event

- ظهور الـ tables دى بيبقى وقت الـ fire بس (بعد كدا مقدرش أجيبها)

*** Output Keyword :**

- بتعمل Runtime Trigger (وقت الـ query بس)

```
delete from Student
output deleted.st_fname
where st_id=2
```

- هقدر ساعتها أستخدم الـ Inserted والـ Deleted

XML