17. Stored Procedures

Table: Appointments

Id	Date	DoctorId	PatientId
1	2020-06-12	1	2
2	2020-06-13	3	2
3	2020-06-14	3	1
4	2020-06-13	1	4
5	2020-06-13	3	6
6	2020-06-14	2	3
7	2020-06-15	2	2
8	2020-06-15	2	2

Table: Doctors

Id	Name
1	Williams
2	Smith
3	Clark
4	Johnson

Table: Patients

Id	Name
1	Robert
2	James
3	David
4	Michael
5	Oliver
6	Mary

Data Script:

```
CREATE TABLE Appointments(
    Id int NOT NULL,
    Date date NOT NULL,
    DoctorId int NOT NULL,
    PatientId int NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Doctors(

    Id int NOT NULL,
    Name varchar(50) NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Patients(

    Id int NOT NULL,
    Name varchar(50) NOT NULL,
    PRIMARY KEY (Id)
);

INSERT INTO Appointments(Id, Date, DoctorId, PatientId)
VALUES (1, '06/12/2020', 1, 2), (2, '06/13/2020', 3, 2),(3, '06/14/2020', 3, 1),
(4, '06/13/2020', 1, 4), (5, '06/13/2020', 3, 6), (6, '06/14/2020', 2, 3);

INSERT INTO Doctors(Id, Name)
VALUES (1, 'Williams'), (2, 'Smith'), (3, 'Clark'), (4, 'Johnson');

INSERT INTO Patients(Id, Name)
VALUES (1, 'Robert'), (2, 'James'), (3, 'David'), (4, 'Michael'), (5, 'Oliver'),
(6, 'Mary');
```


Stored Procedures

By expanding a database you can see all stored procedures in the “Programmability” folder as shown below.
Format of a stored procedure is shown below.

```
CREATE PROCEDURE [dbo].[StoredProcedureName]

@Parameter1 DataType1 input/output
@Parameter2 DataType2 input/output
@Parameter3 DataType3 input/output
---------------------------------
---------------------------------

AS

--Content of the SQL Statement

GO
```

We can execute the stored procedure as shown below

```
EXEC StoredProcedureName Value1, Value2, Value3, ..
```

List Stored Procedures

In the “Programmability” folder you can see all stored procedures.

Create a Stored Procedure

A stored procedure can be created by following below steps.

Steps:

1. Go to the “New Query” editor and type “CREATE PROCEDURE” to start the stored procedure.
2. Give a unique name for the stored procedure.
3. Declare input and output parameters at the start

   Input Parameters – You should declare input parameters using “@” prefix with the data type.

   Output Parameters – You should declare output parameters using “@” prefix with the data type. At the end you should use the keyword “OUTPUT”. When the parameter ends with “OUTPUT”, it is considered as an output parameter. Otherwise it will be considered as an input parameter.
4. Type “AS” keyword before the SQL statement.
5. Include the SQL statement.
6. Use the “GO” keyword at the end.

7. Finally press the “Execute” button to create the stored procedure. You will notice a message, If the stored procedure creation is successful.

Modify a Stored Procedure

You can change an existing stored procedure by following below steps.

Steps:
1. Select the stored procedure that you want to modify in the “Programmability” folder.
2. Right-click on it and select “Modify”.
3. Change the content of the stored procedure.
4. Finally press the “Execute” button to alter the stored procedure.

Delete a Stored Procedure

You can change an existing stored procedure by following below steps.

Steps:
1. Select the stored procedure that you want to delete in the “Programmability” folder.
2. Right-click on it and select “Delete”.
3. Confirm “OK” to delete it.

Rename a Stored Procedure

You can remove an existing stored procedure by following below steps.

Steps:
1. Select the stored procedure that you want to rename in the “Programmability” folder.
2. Right-click on it and select “Rename”.
3. Change the name with a new name.

SELECT

This script shows how to create a stored procedure to list all appointments.

```
CREATE PROCEDURE [dbo].[GetAppointments]

AS

SELECT Appointments.Id, Appointments.Date, Doctors.Name, Patients.Name
FROM Appointments LEFT JOIN Doctors
ON Appointments.DoctorId = Doctors.Id
LEFT JOIN Patients
ON Appointments.DoctorId = Patients.Id
GO
```

The Below command will execute the stored procedure.

```
EXEC GetAppointments;
```

```
CREATE PROCEDURE [dbo].[GetAppointmentById]

@AppointmentId INT,
@PatientId INT OUTPUT,
@DoctorId INT OUTPUT,
@Date DATE

AS

SET @PatientId = (SELECT PatientId FROM Appointments WHERE Id =
@AppointmentId);

SET @DoctorId = (SELECT DoctorId FROM Appointments WHERE Id =
@AppointmentId);

SET @Date = (SELECT Date FROM Appointments WHERE Id = @AppointmentId);
```

Below query will execute the stored procedure.

```
DECLARE @PatientId INT, @DoctorId INT, @Date DATE;
EXEC GetAppointmentById 1, @PatientId OUTPUT, @DoctorId OUTPUT, @Date
OUTPUT;
SELECT @PatientId PatientId, @DoctorId DoctorId, @Date Date;
```

INSERT

```
CREATE PROCEDURE [dbo].[InsertAppointment]

@AppointmentId INT
@Date Date
@DoctorId INT
@PatientId INT

AS

INSERT INTO Appointments(Id, Date, DoctorId, PatientId)
VALUES (@AppointmentId, @Date, @DoctorId, @PatientId)
```

```
EXEC InsertAppointment 1,'12/7/2020', 1, 1;
```

UPDATE

```
CREATE PROCEDURE [dbo].[UpdateAppointment]

@AppointmentId INT
@Date Date
@DoctorId INT
@PatientId INT

AS

UPDATE Appointments

SET AppointmentId = @AppointmentId
, Date = @Date
, DoctorId = @DoctorId
, PatientId = @PatientId)

WHERE Appointments.Id= @AppointmentId
```

```
EXEC UpdateAppointment 1,'12/7/2020', 1, 1;
```

DELETE

```
CREATE PROCEDURE [dbo].[DeleteAppointment]

@AppointmentId INT

AS

DELETE FROM Appointments
WHERE Appointments.Id = @AppointmentId
```

```
EXEC DeleteAppointment 7;
```

Functions

We can create two types of functions in SQL Server.

- Scalar-valued Functions

- Table-valued Functions

Scalar-valued Functions

Scalar valued functions return a value.

```
ALTER FUNCTION [dbo].[fnGetAppointmentDate] (@AppointmentId INT) RETURNS
DATE

AS

BEGIN

DECLARE @Date DATE = (SELECT Date FROM Appointments WHERE Id =
@AppointmentId);

      RETURN @Date;
END
```

Table-valued Functions

Table valued functions return a table.

```
ALTER FUNCTION [dbo].[fnGetAppointments] () RETURNS TABLE

AS
RETURN
(
      SELECT * FROM Appointments;
);
Go
```


Stored Procedures vs Functions

Stored Procedure	Functions

Output is not compulsory	Should have a output
Cannot be called within a function	Can be called within a stored procedure
Both input & output parameters are allowed	Only input parameters are allowed

Nested Stored Procedure

(Call other Stored Procedures from a Stored Procedure)

Content of a stored procedure may be broken down into several reusable stored procedures. It will simplify the code and be more efficient. We can call other stored procedures within a stored procedure. Such a stored procedure is called a nested stored procedure.

Stored Procedure 1:

```
CREATE PROCEDURE [dbo].[GetLatestAppointmentDateByPatientId]

@PatientId INT,
@Date DATE OUTPUT

AS

SET @Date = (SELECT MIN(Date) FROM Appointments
WHERE PatientId = @PatientId)
GO
```

Stored Procedure 2:

```
CREATE PROCEDURE [dbo].[GetAllAppointmentDatesByPatientId]

@PatientId INT,
@LatestDate DATE OUTPUT

AS

EXECUTE GetLatestAppointmentDateByPatientId @PatientId, @LatestDate
OUTPUT;

SELECT DISTINCT Date FROM Appointments
WHERE PatientId = @PatientId

GO
```

Execute:

```
DECLARE @LatestDate DATE;

EXECUTE GetAllAppointmentDatesByPatientId 2, @LatestDate OUTPUT;

SELECT @LatestDate
```

Call Functions within a Function

Function 1:

```
CREATE FUNCTION [dbo].[FnGetPatientNameById] (@PatientId INT) RETURNS
INT

AS
BEGIN
      RETURN (SELECT Name FROM Patients WHERE Id = @PatientId)
END
```

Function 2:

```
CREATE FUNCTION [dbo].[FnGetDoctorNameById] (@DoctorId INT) RETURNS INT

AS
BEGIN
	RETURN (SELECT Name FROM Doctors WHERE Id = @DoctorId)
END
```

Function 3:

```
CREATE FUNCTION [dbo].[FnGetPatientNameAndDoctorName] (@PatientId INT,
@DoctorId INT) RETURNS VARCHAR(100)

AS
BEGIN
	RETURN (SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId))
END
```

Execute:

```
SELECT [dbo].[FnGetPatientNameAndDoctorName](1, 2) [Patient / Doctor]
```

Call Functions within a Stored Procedure

Stored Procedure:

```
CREATE PROCEDURE [dbo].[GetPatientNameAndDoctorName]
@PatientId INT, @DoctorId INT

AS
BEGIN
      SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId) AS [Patient / Doctor]
END
```

Execute:

```
EXEC GetPatientNameAndDoctorName 1, 2
```

```
SELECT [dbo].[FnGetPatientNameAndDoctorName](1, 2) [Patient / Doctor]
```

Calling Functions within a Stored Procedure

The below stored procedure is used to concatenate the patient's name and doctor's name.

Stored Procedure:

```
CREATE PROCEDURE [dbo].[GetPatientNameAndDoctorName]

@PatientId INT, @DoctorId INT

AS

BEGIN

	SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId) AS [Patient / Doctor]

END
```

Execute:

```
EXEC GetPatientNameAndDoctorName 1, 2
```