SQL Server 2012 – Database Development

Lesson 5: Working with Joins and Subqueries

Lesson Objectives

- In this lesson, you will learn:
  - What are joins?
  - Types of Joins
  - Using Subqueries
  - Restrictions on Subqueries

Joins

- Retrieving data from several tables ,based on a relationship between certain columns in these tables
- Most of the time it would be the Foreign key
- One can join maximum 256 tables in single query

Types of Joins

- INNER join
  - Equijoin
  - Nonequijoin
- Outer join
  - LEFT outer
  - RIGHT outer
  - FULL outer
- Self join
- Cross join

Introducing T- SQL JOIN Syntax

```
SELECT   <<Column List>>
FROM   Table name1
[INNER]  JOIN Table name2
ON join criteria
WHERE condition criteria
```

OR

```
SELECT   <<Column List>>
FROM   Table name1 , Table name2
WHERE Join criteria
AND condition criteria
```

Inner Join

```
USE Northwind
GO

SELECT ProductID, ProductName, CategoryName
FROM Products, Categories
WHERE Products.CategoryID=Categories.CategoryID
GO
```

INNER JOIN – More than 2 tables

- Number of joins = number of tables - 1

```
USE northwind
GO

SELECT  categoryname, description,
productname, productid, companyname, suppliers.city
FROM products, categories, suppliers
WHERE Products.categoryid = Categories.categoryid
AND Products.supplierid = Suppliers.supplierid
AND  Suppliers.city = 'London'
ORDER by  productname
GO
```

INNER JOIN – More than 2 tables

Example :-

```
SELECT  categoryname, description, productname,
productid, companyname, Suppliers.city
FROM products
INNER JOIN categories
ON Products.categoryid = Categories.categoryid
INNER JOIN Suppliers
ON  Products.supplierid = Suppliers.supplierid
WHERE Suppliers.city = 'london'
ORDER by Prodcts.productname
```

OUTER JOIN

- Inner joins eliminates rows which doesn't have a match in the joining tables
- Outer joins returns all rows from one of the joining tables and matched rows from the others
- Outer joins can be further classified as
  - LEFT OUTER JOIN or LEFT JOIN
  - RIGHT OUTER JOIN or RIGHT JOIN
  - FULL OUTER JOIN or FULL JOIN

Using Left Outer Joins

- Left Outer - all records from left table and corresponding matching records from right

T-SQL Syntax

```
SELECT  categoryname, description,
productname, productid
FROM CATEGORIES
LEFT OUTER JOIN PRODUCTS
ON PRODUCTS.categoryid = CATEGORIES.categoryid
GO
```

RIGHT OUTER Join

- T-SQL Syntax

```
SELECT  categoryname, description,
productname, productid
FROM CATEGORIES
RIGHT OUTER JOIN PRODUCTS
ON PRODUCTS.categoryid = CATEGORIES.categoryid
GO
```

FULL OUTER JOIN

- FULL OUTER JOIN - includes all rows from both tables

```
USE Northwind;
GO
SELECT  categoryname, description,
productname, productid
FROM CATEGORIES
FULL OUTER JOIN PRODUCTS
ON PRODUCTS.categoryid = CATEGORIES.categoryid
GO
```

SELF JOIN

- A self join is when a table joins with itself
- Self join is possible when the table has references to itself

  For example to query employees along with managers who are also employees

```
SELECT  emp.EmployeeID , emp.EmployeeName, mgr.EmpoyeeName
FROM  Employee emp
INNER JOIN Employee mgr
ON emp.ManagerID = mgr.EmployeeID
```

CROSS JOIN

- Cross join does not have a WHERE clause
- Result set is the number of rows in the first table multiplied by the number of rows in the second table- a Cartesian product

```
USE Northwind;
GO
SELECT  categoryname, description, productname, productid
FROM CATEGORIES ,PRODUCTS
GO
```

```
USE Northwind;
GO
SELECT  categoryname, description, productname, productid
FROM CATEGORIES CROSS JOIN PRODUCTS
ORDER BY CategoryName
```

Subquery

- A sub query is an SQL statement that is used within another SQL statement
- Subqueries are used to handle query requests that are expressed as the results of other queries
- Subquery can be embedded in WHERE /HAVING statement

```
USE Northwind;
GO
SELECT EmployeeID,EmployeeName
FROM Employees
WHERE Region=
 (SELECT Region from Employees
  WHERE EmployeeID=12345)
```

Subquery restrictions

- Subquery_select_list can consist of only one column name
- Subqueries can be nested inside the where or having clause
- Subquery can appear almost anywhere an expression can be used, if it returns a single value
- Subqueries cannot manipulate their results internally i.e. ORDER BY Clause cannot be used inside the subquery
- If the sub query returns more than 1 row then outer query has to use appropirate operator like IN , ANY etc.

Types of Subqueries

- Single Row Sub query
- Multi Row Sub query
- Sub query for Existence
- Correlated Sub Query

Multi Row Subquery

- Subqueries returns a list of zero or more values and can include a GROUP BY or HAVING clause
- >ALL means greater than every value
- >ANY means greater than at least one value

```
SELECT ProductID, ProductName, UnitPrice
FROM PRODUCTS WHERE SupplierID  IN
(SELECT SupplierID
 FROM Suppliers
 WHERE CITY="NEW York")
```

Multi Row Subquery

```
SELECT ProductID, ProductName, SupplierID
FROM Products
WHERE UnitPrice > ALL
(Select UnitPrice
 FROM  Products
 WHERE SupplierID=10098)
```

EXISTS

- EXISTS checks for a existence of a condition
- The EXISTS condition is considered "to be met" if the subquery returns at least one row.

```
SELECT SupplierID
FROM suppliers
WHERE EXISTS
  (select 'A'
    from orders
    where suppliers.supplier_id = orders.supplier_id);
```

Summary

- In this lesson, you have learnt:
  - Joins are used to fetch data from more than 2 tables.
  - Join types: equijoin, non-equijoin, outer join, self join
  - Subquery is query within a query or nested query
  - Subquery types

Review Question

- Question 1: If we do not include a join condition then the result leads to _______
- Question 2: _______ return all rows from at least one of the tables in the FROM clause
- Question 3: When subquery depends on the outer query for its execution then it is ______ subquery.
- Question 4: Subquery _______ evaluates to TRUE or FALSE rather than returning any data