Departement Informatika
Department of Informatics

MS SQL 2008

Data retrieval

Chapter 8 (Part II) | Hotek, 2008

Learning objectives

- Admin
- Retrieving data from more than one table

Admin

- No practical sessions in week of 2013/03/18 – 2013/03/29
  - Thursday Timetable (2013/03/19)
  - Friday Timetable (2013/03/20)
  - Lectures End (2013/03/20)
  - Public Holiday (2013/03/21)
  - March Recess - (2013/03/21 to 2013/04/01)

- Semester test is upcoming soon – refer to ClickUP

Combining data from more than one table

- Accomplished using the “JOIN” operator in the FROM clause of the query
  - INNER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
  - CROSS JOIN

INNER JOIN

- An INNER JOIN is a join in which the values in the columns being joined are compared using a comparison operator
  - Specified in the FROM clause

```
USE AdventureWorks
GO
SELECT *
FROM HumanResources.Employee AS e
    INNER JOIN Person.Contact AS c
    ON e.ContactID = c.ContactID
ORDER BY c.LastName
```

```
SELECT a.FirstName,
a.LastName, b.Title
FROM Person.Contact a INNER
JOIN
HumanResources.Employee b
on a.ContactID = b.ContactID
GO
```

INNER JOIN

- Leave all search arguments in the WHERE clause

```
SELECT a.ProductID, a.Name, b.SalesOrderID
FROM Production.Product a INNER JOIN
Sales.SalesOrderDetail b
        ON a.ProductID = b.ProductID
WHERE a.Color = 'Black'
GO
```

OUTER JOIN

- Outer joins return all rows from at least one of the tables or views mentioned in the FROM clause, as long as those rows meet any WHERE or HAVING search conditions
  - LEFT OUTER JOIN - all rows from the left table are returned
  - RIGHT OUTER JOIN - all rows from the right table are returned
  - FULL OUTER JOIN - all rows from both tables are returned

LEFT OUTER JOIN

```
USE AdventureWorks;
GO
SELECT p.Name, pr.ProductReviewID
FROM Production.Product p
LEFT OUTER JOIN Production.ProductReview pr
ON p.ProductID = pr.ProductID
ORDER BY pr.ProductReviewID DESC
```

RIGHT OUTER JOIN

```
USE AdventureWorks
GO


SELECT st.Name AS Territory, sp.SalesPersonID
FROM Sales.SalesTerritory st
RIGHT OUTER JOIN Sales.SalesPerson sp
ON st.TerritoryID = sp.TerritoryID
ORDER BY 1 ASC
GO
```

FULL OUTER JOIN

```
USE AdventureWorks
GO

SELECT a.ProductID, a.Name, b.SalesOrderID
FROM Production.Product a FULL OUTER JOIN
Sales.SalesOrderDetail b
        ON a.ProductID = b.ProductID
WHERE a.Color = 'Black'
ORDER BY 1
GO
```

CROSS JOIN

- A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join.

- The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table

```
USE AdventureWorks;
GO

SELECT p.SalesPersonID, t.Name
AS Territory
FROM Sales.SalesPerson p
CROSS JOIN Sales.SalesTerritory t
ORDER BY p.SalesPersonID;
```

CROSS JOIN

- If a WHERE clause is added, the cross join behaves as an inner join
- Hotek (2008) advises against the CROSS JOIN

```
USE AdventureWorks;
GO
SELECT p.SalesPersonID, t.Name AS Territory
FROM Sales.SalesPerson p
CROSS JOIN Sales.SalesTerritory t
WHERE p.TerritoryID = t.TerritoryID
ORDER BY p.SalesPersonID;
```

SELF JOIN

- A table can be joined to itself in a self-join
- Provide a table alias for at least one of instance of the table name

```
USE AdventureWorks;
GO

SELECT b.LoginID AS ManagerName, a.ManagerID, a.LoginID, a.EmployeeID
FROM HumanResources.Employee AS a
   JOIN HumanResources.Employee AS b
      ON a.ManagerID = b.EmployeeID
ORDER BY b.LoginID, a.LoginID
```

SELF JOIN

```
USE AdventureWorks;
GO

SELECT b.ProductID, b.Name, b.ListPrice, a.UnitPrice AS 'Sales Price'
FROM Sales.SalesOrderDetail AS a
    JOIN Production.Product AS b
    ON a.ProductID = b.ProductID AND a.UnitPrice < b.ListPrice
WHERE b.ProductID = 718
GO
```

Sub queries

- This is when SELECT statements are embedded within SELECT statements

```
SELECT a.ProductID, a.Name, a.ListPrice
FROM Production.Product a
WHERE a.ListPrice > (SELECT AVG(b.ListPrice) FROM
Production.Product b)
GO
```

DISTINCT keyword

- Used to retrieve unique results

```
USE AdventureWorks;
GO

SELECT b.ProductID, b.Name, b.ListPrice, a.UnitPrice AS 'Sales Price'
FROM Sales.SalesOrderDetail AS a
   JOIN Production.Product AS b
   ON a.ProductID = b.ProductID AND a.UnitPrice < b.ListPrice
WHERE b.ProductID = 718

SELECT DISTINCT b.ProductID, b.Name, b.ListPrice, a.UnitPrice AS 'Sales Price'
FROM Sales.SalesOrderDetail AS a
   JOIN Production.Product AS b
   ON a.ProductID = b.ProductID AND a.UnitPrice < b.ListPrice
WHERE b.ProductID = 718
```

DISTINCT with COUNT clause

```
USE AdventureWorks;
GO

SELECT COUNT(*) FROM HumanResources.Employee

SELECT COUNT(ManagerID) FROM HumanResources.Employee

SELECT DISTINCT COUNT(ManagerID) FROM HumanResources.Employee

SELECT COUNT(DISTINCT ManagerID) FROM HumanResources.Employee
GO
```

Practical Exercise

- Will be marked in the next class after the holiday. Refer to the schedule on ClickUP.
- Write queries for:
  - INNER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN