Module 4

Querying Multiple Tables

Module Overview

- Understanding Joins
- Querying with Inner Joins
- Querying with Outer Joins
- Querying with Cross Joins and Self Joins

Lesson 1: Understanding Joins

- The FROM Clause and Virtual Tables
- Join Terminology: Cartesian Product
- Overview of Join Types
- T-SQL Syntax Choices
- Demonstration: Understanding Joins

The FROM Clause and Virtual Tables

- FROM clause determines source tables to be used in SELECT statement
- FROM clause can contain tables and operators
- Result set of FROM clause is virtual table
  - Subsequent logical operations in SELECT statement consume this virtual table
- FROM clause can establish table aliases for use by subsequent phases of query

Join Terminology: Cartesian Product

- Characteristics of a Cartesian product
  - Output or intermediate result of FROM clause
  - Combine all possible combinations of two sets
  - In T-SQL queries, usually not desired
    - Special case: table of numbers

Name
Davis
Funk
King

Product
Alice Mutton
Crab Meat
Ipoh Coffee

Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipoh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipoh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipoh Coffee

Overview of Join Types

- Join types in FROM clauses specify the operations performed on the virtual table:

Join Type	Description
Cross	Combines all rows in both tables (creates Cartesian product)
Inner	Starts with Cartesian product; applies filter to match rows between tables based on predicate
Outer	Starts with Cartesian product; all rows from designated table preserved, matching rows from other table retrieved. Additional NULLs inserted as placeholders

T-SQL Syntax Choices

- ANSI SQL-92
  - Tables joined by JOIN operator in FROM Clause

```
SELECT ...
FROM   Table1 JOIN Table2
       ON <on_predicate>
```

- ANSI SQL-89
  - Tables joined by commas in FROM Clause
    - Not recommended: accidental Cartesian products!

```
SELECT ...
FROM   Table1, Table2
WHERE  <where_predicate>
```

Demonstration: Understanding Joins

In this demonstration, you will see how to:

- Use joins

Lesson 2: Querying with Inner Joins

- Understanding Inner Joins
- Inner Join Syntax
- Inner Join Examples
- Demonstration: Querying with Inner Joins

Understanding Inner Joins

- Returns only rows where a match is found in both input tables
- Matches rows based on attributes supplied in predicate
  - ON clause in SQL-92 syntax (preferred)
  - WHERE clause in SQL-89 syntax
- Why filter in ON clause?
  - Logical separation between filtering for purposes of join and filtering results in WHERE
  - Typically no difference to query optimizer
- If join predicate operator is =, also known as equi-join

Inner Join Syntax

- List tables in FROM Clause separated by JOIN operator
- Table aliases preferred
- Table order does not matter

```
FROM t1 JOIN t2
    ON t1.column = t2.column
```

```
SELECT o.orderid,
       o.orderdate,
       od.productid,
       od.unitprice,
       od.qty
FROM Sales.Orders AS o
      JOIN Sales.OrderDetails AS od
       ON o.orderid = od.orderid;
```

Inner Join Examples

- Join based on single matching attribute

```
SELECT ...
FROM Production.Categories AS C
    JOIN Production.Products AS P
        ON C.categoryid = P.categoryid;
```

- Join based on multiple matching attributes (composite join)

```
-- List cities and countries where both --
customers and employees live
SELECT DISTINCT e.city, e.country
FROM Sales.Customers AS c
JOIN    HR.Employees AS e
    ON c.city = e.city AND
    c.country = e.country;
```

Demonstration: Querying with Inner Joins

In this demonstration, you will see how to:

- Use inner joins

Lesson 3: Querying with Outer Joins

- Understanding Outer Joins
- Outer Join Syntax
- Outer Join Examples
- Demonstration: Querying with Outer Joins

Understanding Outer Joins

- Returns all rows from one table and any matching rows from second table
- One table's rows are "preserved"
  - Designated with LEFT, RIGHT, FULL keyword
  - All rows from preserved table output to result set
- Matches from other table retrieved
- Additional rows added to results for nonmatched rows
  - NULLs added in places where attributes do not match
- Example: return all customers and, for those who have placed orders, return order information; customers without matching orders will display NULL for order details

Outer Join Syntax

- Return all rows from first table, only matches from second:

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Return all rows from second table, only matches from first:

```
FROM t1 RIGHT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Return only rows from first table, with no match in second:

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
WHERE     t2.col IS NULL
```

Outer Join Examples

- All customers with order details if present:

```
SELECT c.custid, c.contactname, o.orderid,
o.orderdate
FROM Sales.Customers AS C
LEFT OUTER JOIN Sales.Orders AS O
    ON c.custid = o.custid;
```

- Customers who did not place orders:

```
SELECT c.custid, c.contactname, o.orderid,
o.orderdate
FROM Sales.Customers AS C LEFT OUTER JOIN
Sales.Orders AS O
    ON c.custid = o.custid
WHERE o.orderid IS NULL;
```

Demonstration: Querying with Outer Joins

In this demonstration, you will see how to:

- Use outer joins

Lesson 4: Querying with Cross Joins and Self Joins

- Understanding Cross Joins
- Cross Join Syntax
- Cross Join Examples
- Understanding Self Joins
- Self Join Examples
- Demonstration: Querying with Cross Joins and Self Joins

Understanding Cross Joins

- Combine each row from first table with each row from second table
- All possible combinations output
- Logical foundation for inner and outer joins
  - Inner join starts with Cartesian product, adds filter
  - Outer join takes Cartesian output, filtered, adds back nonmatching rows (with NULL placeholders)
- Due to Cartesian product output, not typically a desired form of join
- Some useful exceptions:
  - Table of numbers, generating data for testing

Cross Join Syntax

- No matching performed, no ON clause used
- Return all rows from left table combined with each row from right table (ANSI SQL-92 syntax):

```
SELECT  ...
FROM t1 CROSS JOIN t2
```

- Return all rows from left table combined with each row from right table (ANSI SQL-89 syntax):

```
SELECT  ...
FROM t1, t2
```

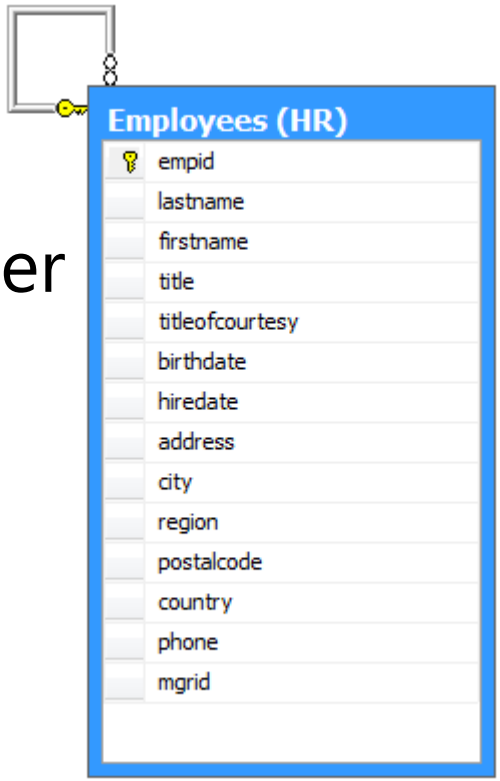
Cross Join Examples

- Create test data by returning all combinations of two inputs:

```
SELECT e1.firstname, e2.lastname
FROM HR.Employees AS e1
CROSS JOIN HR.Employees AS e2;
```

Understanding Self Joins

- Why use self joins?
  - Compare rows in same table to each other
- Create two instances of same table in FROM clause
  - At least one alias required
- Example: Return all employees and the name of the employee's manager



Self Join Examples

- Return all employees with ID of employee's manager when a manager exists (inner join):

```
SELECT  e.empid, e.lastname,
        e.title, e.mgrid, m.lastname
FROM    HR.Employees AS e
JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

- Return all employees with ID of manager (outer join). This will return NULL for the CEO:

```
SELECT  e. empid, e.lastname,
            e.title, m.mgrid
FROM HR.Employees AS e
LEFT OUTER JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

Demonstration: Querying with Cross Joins and Self Joins

In this demonstration, you will see how to:

- Use self joins and cross joins

Lab: Querying Multiple Tables

- Exercise 1: Writing Queries That Use Inner Joins
- Exercise 2: Writing Queries That Use Multiple-Table Inner Joins
- Exercise 3: Writing Queries That Use Self Joins
- Exercise 4: Writing Queries That Use Outer Joins
- Exercise 5: Writing Queries That Use Cross Joins

Logon Information
Virtual machine: **20761C-MIA-SQL**
User name: **ADVENTUREWORKS\Student**
Password: **Pa55w.rd**

Estimated Time: 50 minutes

Lab Scenario

You are an Adventure Works business analyst who will be writing reports using corporate databases stored in SQL Server. You have been given a set of business requirements for data and you will write T-SQL queries to retrieve the specified data from the databases. You notice that the data is stored in separate tables, so you will need to write queries using various join operations.

Module Review and Takeaways

- Review Question(s)
- Best Practice