

Module 12

Using Set Operators

Module Overview

- Writing Queries with the UNION Operator
- Using EXCEPT and INTERSECT
- Using APPLY

Lesson 1: Writing Queries with the UNION Operator

- Interactions Between Sets
- Using the UNION Operator
- Using the UNION ALL Operator
- Demonstration: Using UNION and UNION ALL

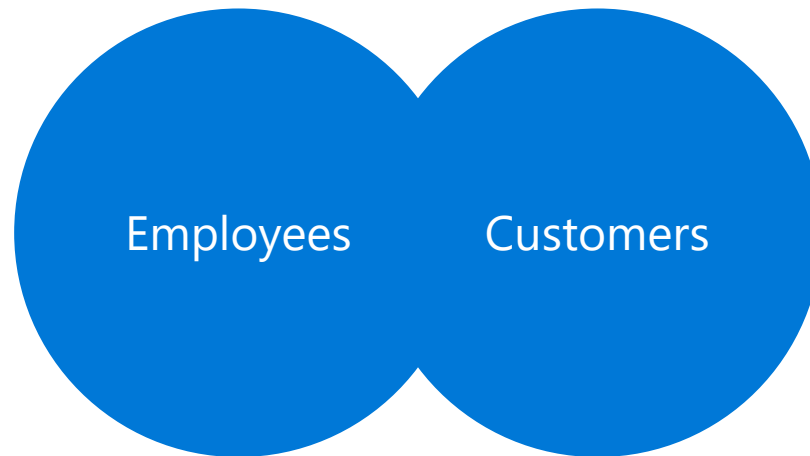
Interactions Between Sets

- The results of two input queries may be further manipulated
- Sets may be combined, compared, or operated against each other
- Both sets must have the same number of compatible columns
- ORDER BY not allowed in input queries, but may be used for result of set operation
- NULLs considered equal when comparing sets

```
<SELECT query_1>  
<set_operator>  
<SELECT query_2>  
[ORDER BY <sort_list>];
```

Using the UNION Operator

- UNION returns a result set of distinct rows combined from both input sets
- Duplicates are removed during query processing (affects performance)



```
-- only distinct rows from both queries are returned
SELECT country, region, city FROM HR.Employees
UNION
SELECT country, region, city FROM Sales.Customers;
```

Using the UNION ALL Operator

- UNION ALL returns a result set with all rows from both input sets
- To avoid the performance penalty caused by filtering duplicates, use UNION ALL over UNION whenever requirements allow it

```
-- all rows from both queries will be returned
SELECT country, region, city FROM HR.Employees
UNION ALL
SELECT country, region, city FROM Sales.Customers;
```

Demonstration: Using UNION and UNION ALL

In this demonstration, you will see how to:

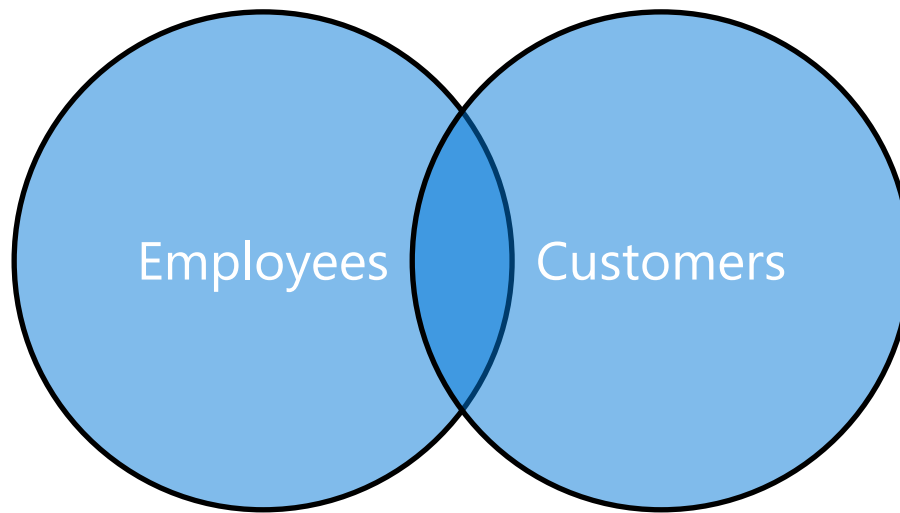
- Use UNION and UNION ALL

Lesson 2: Using EXCEPT and INTERSECT

- Using the INTERSECT Operator
- Using the EXCEPT Operator
- Demonstration: Using EXCEPT and INTERSECT

Using the INTERSECT Operator

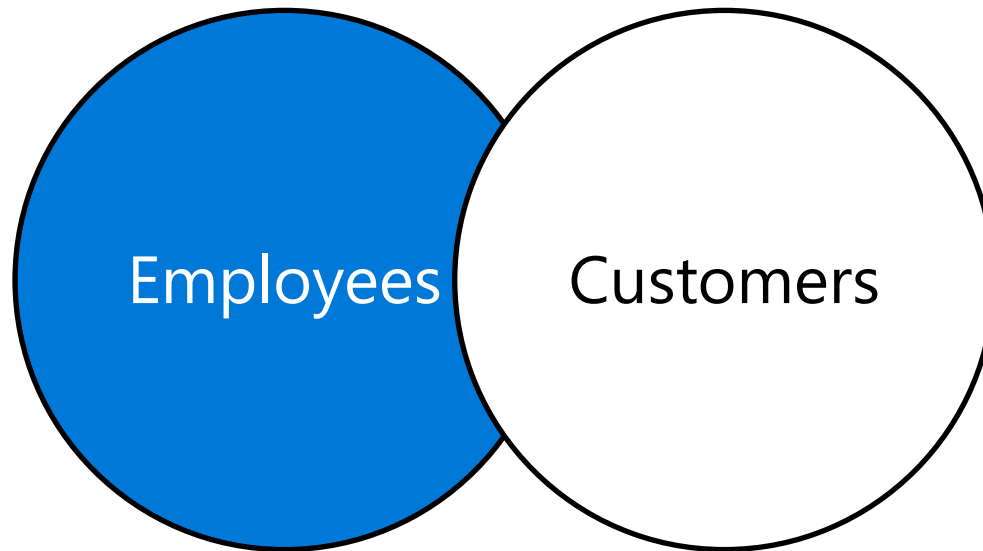
- INTERSECT returns the distinct set of rows that appear in both input result sets



```
-- only rows that exist in both queries will be returned
SELECT country, region, city FROM HR.Employees
INTERSECT
SELECT country, region, city FROM Sales.Customers;
```

Using the EXCEPT Operator

- EXCEPT returns only distinct rows that appear in the left set but not the right
 - The order in which sets are specified matters



```
-- only rows from Employees will be returned
SELECT country, region, city FROM HR.Employees
EXCEPT
SELECT country, region, city FROM Sales.Customers;
```

Demonstration: Using EXCEPT and INTERSECT

In this demonstration, you will see how to:

- Use INTERSECT and EXCEPT

Lesson 3: Using APPLY

- Using the APPLY Operator
- The CROSS APPLY Operator
- The OUTER APPLY Operator
- CROSS APPLY and OUTER APPLY Features
- Demonstration: Using CROSS APPLY and OUTER APPLY

Using the APPLY Operator

- APPLY is a table operator used in the FROM clause
- Two forms—CROSS APPLY and OUTER APPLY
- Operates on two input tables, referred to as left and right
- Right table may be any table expression including a derived table or a table-valued function

```
SELECT <column_list>  
FROM <left_table_source> AS <alias>  
[CROSS][OUTER] APPLY  
    <right_table_source> AS <alias>;
```

The CROSS APPLY Operator

- CROSS APPLY applies the right table source to each row in the left table source
 - Only rows with results in both the left table source and right table source are returned
- Most INNER JOIN statements can be rewritten as CROSS APPLY statements

```
SELECT o.orderid, o.orderdate,  
       od.productid, od.unitprice, od.qty  
FROM Sales.Orders AS o  
CROSS APPLY (SELECT productid, unitprice, qty  
              FROM Sales.OrderDetails AS so  
              WHERE so.orderid = o.orderid  
            ) AS od;
```

The OUTER APPLY Operator

- OUTER APPLY applies the right table source to each row in the left table source
 - All rows from the left table source are returned—values from the right table source are returned where they exist, otherwise NULL is returned
- Most LEFT OUTER JOIN statements can be rewritten as OUTER APPLY statements

```
SELECT DISTINCT s.country AS supplier_country,  
c.country as customer_country  
FROM Production.Suppliers AS s  
OUTER APPLY (SELECT country  
FROM Sales.Customers AS cu  
WHERE cu.country = s.country  
) AS c  
ORDER BY supplier_country;
```

CROSS APPLY and OUTER APPLY Features

- CROSS APPLY and OUTER APPLY allow query expressions that could not appear in a JOIN to return as part of a single result set
 - For example, table-valued functions (TVFs)

```
SELECT S.supplierid, s.companyname,  
P.productid, P.productname, P.unitprice  
FROM Production.Suppliers AS S  
CROSS APPLY dbo.fn_TopProductsByShipper(S.supplierid) AS P;
```


Demonstration: Using CROSS APPLY and OUTER APPLY

In this demonstration, you will see how to:

- Use forms of the APPLY Operator

Lab: Using Set Operators

- Exercise 1: Writing Queries That Use UNION Set Operators and UNION ALL Multi-Set Operators
- Exercise 2: Writing Queries That Use the CROSS APPLY and OUTER APPLY Operators
- Exercise 3: Writing Queries That Use the EXCEPT and INTERSECT Operators

Logon Information

Virtual machine: **20761C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Estimated Time: 60 minutes

Lab Scenario

As a business analyst for Adventure Works, you will be writing reports using corporate databases stored in SQL Server. You have been provided with a set of business requirements for data and you will write T-SQL queries to retrieve the specified data from the databases. Because of the complex business requirements, you will need to prepare combined results from multiple queries using set operators.

Module Review and Takeaways

- Review Question(s)