

# **NET3000**

## **Database Concepts and SQL**

Instructor: Phil Kaufman

### **Lecture 3**

#### **Joins, Update, Insert, Delete**

September 21, 2015

# Introducing Joins

- SQL joins are used to combine rows from two or more tables.
- An SQL JOIN clause is used to combine rows from two or more tables, based on a **common field** between them.

# Types of Joins

- Here is the list of different SQL JOINS you can use:
  - **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
  - **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table
  - **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
  - **FULL JOIN**: Return all rows when there is a match in ONE of the tables
- The most common type of join is: **SQL INNER JOIN (simple join)**. An SQL INNER JOIN return all rows from multiple tables where the join condition is met.

# SQL INNER JOIN

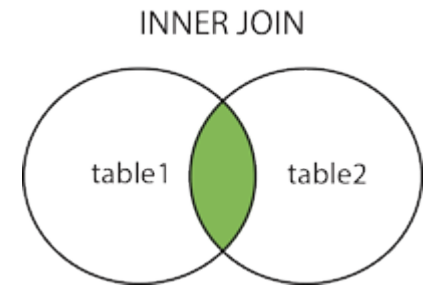
The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in **both** tables.

- SQL INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
      INNER JOIN table2
      ON table1.column_name=table2.column_name;
```

or:

```
SELECT column_name(s)
FROM table1
      JOIN table2
      ON table1.column_name=table2.column_name;
```



**Notice: INNER JOIN is the same as JOIN.**

# SQL INNER JOIN

- Let's look at a selection from the "Orders" table:
- Then, have a look at a selection from the "Customers" table:
- Notice that the "CustomerID" column in the "Orders" table refers to the customer in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

# SQL INNER JOIN

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
    INNER JOIN Customers
    ON Orders.CustomerID = Customers.CustomerID;
```

[http://www.w3schools.com/sql/trysql.asp?filename=trysql\\_select\\_join](http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_join)

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

# Inner Joins

- Include a WHERE clause to restrict the rows that are returned in the result set.
- The columns used as the join condition do not need to have the same column name but must share a common data type.
- Do not use a null value as a join condition because null values do not evaluate equally with one another.
- SQL Server does not guarantee an order in the result set unless one is specified with an ORDER BY clause.
- You can use multiple JOIN clauses and join more than two tables in the same query.

# SQL LEFT JOIN

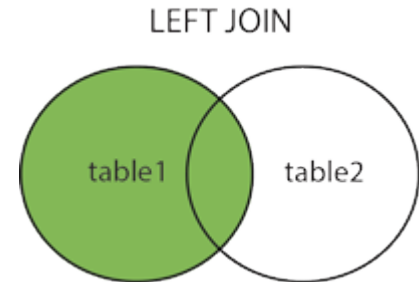
- The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

- SQL LEFT JOIN Syntax

- ```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

or:

- ```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```



**Notice: In some databases LEFT JOIN is called LEFT OUTER JOIN.**



# SQL LEFT JOIN

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
  LEFT JOIN Orders
    ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

# SQL LEFT JOIN

**Note:** The LEFT JOIN keyword returns all the rows from the left table (Customers), even if there are no matches in the right table (Orders).

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

# SQL RIGHT JOIN

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

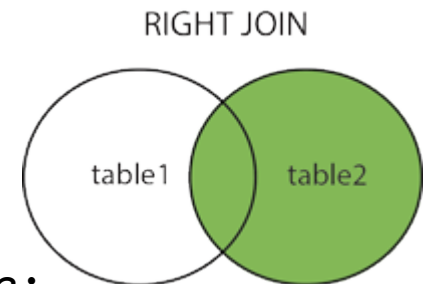
## SQL RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table1
    RIGHT JOIN table2
    ON table1.column_name=table2.column_name;
```

or:

```
SELECT column_name(s)
FROM table1
    RIGHT OUTER JOIN table2
    ON table1.column_name=table2.column_name;
```

**Notice: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.**



# SQL RIGHT JOIN

```
SELECT Orders.OrderID, Employees.FirstName
FROM Orders
      RIGHT JOIN Employees
      ON Orders.EmployeeID=Employees.EmployeeID
ORDER BY Orders.OrderID;
```

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	12/8/1968	EmpID1.pic	Education includes a BA in psychology.....
2	Fuller	Andrew	2/19/1952	EmpID2.pic	Andrew received his BTS commercial and....
3	Leverling	Janet	8/30/1963	EmpID3.pic	Janet has a BS degree in chemistry....

# SQL RIGHT JOIN

**Note:** The RIGHT JOIN keyword returns all the rows from the right table (Employees), even if there are no matches in the left table (Orders).

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	12/8/1968	EmpID1.pic	Education includes a BA in psychology.....
2	Fuller	Andrew	2/19/1952	EmpID2.pic	Andrew received his BTS commercial and....
3	Leverling	Janet	8/30/1963	EmpID3.pic	Janet has a BS degree in chemistry....

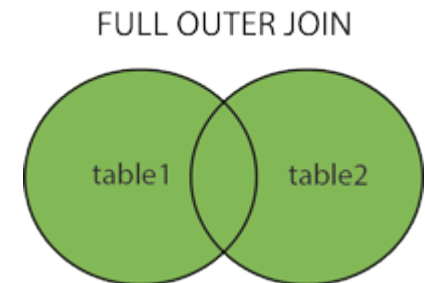
# FULL OUTER JOIN

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.

## SQL FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
     FULL OUTER JOIN table2
     ON table1.column_name=table2.column_name;
```



# FULL OUTER JOIN

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

# FULL OUTER JOIN

The following SQL statement selects all customers, and all orders:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
      FULL OUTER JOIN Orders
      ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

A selection from the result set may look like this:

CustomerName	OrderID
Alfreds Futterkiste	
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365
	10382
	10351



# FULL OUTER JOIN

**Note:** The FULL OUTER JOIN keyword returns all the rows from the left table (Customers), and all the rows from the right table (Orders). If there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

CustomerName	OrderID
Alfreds Futterkiste	
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365
	10382
	10351

# When to Use Left or Right Outer Joins

- Use left or right outer joins when you need a complete list of data from one of the joined tables.
- Use a left outer join to display all rows from the first-named table (the table on the left of the JOIN clause).
- Use a right outer join to display all rows from the second-named table (the table on the right of the JOIN clause).
- Do not use a null value as a join condition because null values do not evaluate equally with one another.
- You can abbreviate the LEFT OUTER JOIN or RIGHT OUTER JOIN clause as LEFT JOIN or RIGHT JOIN.

# Good to know: Using Cross Joins

- Cross joins display every combination of all rows in the joined tables.
- A common column is not required, and the ON clause is not used for cross joins.
- Cross joins are rarely used on a normalized database, but you can use them to generate test data for a database or lists of all possible combinations for checklists or business templates.
- When you use cross joins, SQL produces a Cartesian product in which the number of rows in the result set is equal to the number of rows in the first table, multiplied by the number of rows in the second table.
- For example, if there are 8 rows in one table and 9 rows in the other table, SQL returns a total of 72 rows.

# 3 More commands

- INSERT
- UPDATE
- DELETE

# INSERT INTO

- Is used to insert new records in a table.
- It is possible to write the INSERT INTO statement in two forms:
  - The first form does not specify the column names where the data will be inserted, only their values:
    - INSERT INTO *table\_name*  
VALUES (*value1,value2,value3,...*);
  - The second form specifies both the column names and the values to be inserted:
    - INSERT INTO *table\_name* (*column1,column2,column3,...*)  
VALUES (*value1,value2,value3,...*);

# INSERT INTO

- Example:

```
INSERT INTO Customers (CustomerName, ContactName,  
Address, City, PostalCode, Country)  
VALUES ('Cardinal','Tom B. Erichsen','Skagen  
21','Stavanger','4006','Norway');
```

# UPDATE

- The UPDATE statement is used to update existing records in a table.

```
UPDATE table_name  
SET column1 = value1, column2 = value2,...  
WHERE some_column = some_value;
```

- Notice the WHERE clause in the SQL UPDATE statement!
- The WHERE clause specifies which **record or records** that should be updated.
- **If you omit the WHERE clause, all records will be updated!**

# UPDATE

- Example:

```
UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City = 'Hamburg'  
WHERE CustomerName = 'Alfreds Futterkiste';
```



# DELETE

- The DELETE statement is used to delete rows in a table.

```
DELETE FROM table_name  
WHERE some_column = some_value;
```

- Notice the WHERE clause in the SQL DELETE statement!
- The WHERE clause specifies which record or records that should be deleted.
- **If you omit the WHERE clause, all records will be deleted!**

# DELETE

- Example:

```
DELETE FROM Customers  
WHERE CustomerName = 'Alfreds Futterkiste'  
AND ContactName = 'Maria Anders';
```

# Practice

- Use the sample database we downloaded in Lab2 called AdventureWorks and practice the joins, insert, update and delete.
- Remember that the more you practice the easier it gets for you to understand the material presented in the lecture and lab time.

# Conclusion

We learned about:

- Joins
- Insert
- Update
- Delete