Utilizing Joins

TRAINSIGNAL

THE GLOBAL LEADER IN PROFESSIONAL COMPUTER TRAINING™

Statement Components

```
SELECT <columns>
FROM <tables>
WHERE <conditions>
GROUP BY <columns>
HAVING <conditions>
ORDER BY <columns>
```


Join Basics

Join Basics

Relational Concepts
How Joins Work
Join Syntax
Best Practices and Coding Conventions

Relational Concepts

Data is divided across multiple tables
- Storage space
- Flexibility in retrieving data
- Performance

Must combine tables to make data "human readable"

How Joins Work

One column from each table to create "seam"
Seam is used to match data from one table to the other
Can join two or more tables

Customers	
CustomerID	Name
42	Christopher
43	Karin
44	Susan
45	Dave
46	Brian

Orders		
OrderID	CustomerID	OrderDate
1	42	12/5/2012
2	42	12/6/2012
3	43	12/6/2012
4	43	12/7/2012
5	44	12/7/2012

Join Syntax

```
-- ANSI Standard Joins
SELECT <columns>
FROM Schema.Table1 AS t1
  INNER JOIN Schema.Table2 AS t2 ON t1.Column = t2.Column;

-- WHERE line joins
-- SQL will rewrite
-- Not preferred method
SELECT <columns>
FROM Schema.Table1 AS t1
  , Schema.Table2 AS t2
WHERE t1.Column = t2.Column;
```


Coding Best Practices and Conventions

Always alias table names
- `Schema.Table AS t`

Always use two part naming for columns
- `t.ColumnName`

Place each join on separate line
- `FROM Schema.Table1 AS t1`
  `INNER JOIN Schema.Table2 AS t2 ON t1.Column = t2.Column`

Place tables in logical order

Join Types

Join Types

Inner Joins
Outer Joins
Cross Joins
Self Joins

Inner Joins

Most common type of join
Only rows that match are returned
Find customers' orders

Inner Joins

Customers	
CustomerID	Name
42	Christopher
43	Karin
44	Susan
45	Dave
46	Brian

Orders		
OrderID	CustomerID	OrderDate
1	42	12/5/2012
2	42	12/6/2012
3	43	12/6/2012
4	43	12/7/2012
5	44	12/7/2012

OrderID	CustomerID	OrderDate	Name
1	42	12/5/2012	Christopher
2	42	12/6/2012	Christopher
3	43	12/6/2012	Karin
4	43	12/7/2012	Karin
5	44	12/7/2012	Susan

Outer Joins

Rows from tables are returned even if they don't match

Types:

- `FULL`
  - Returns rows from both tables even if they don't match the join statement
  - Find customers without orders and orphaned orders
- `LEFT` or `RIGHT`
  - No difference except in order of tables in `FROM` clause
  - Direction points at table to retrieve non-matching rows from
  - Find customers who haven't placed orders

Outer Joins

Customers

CustomerID	Name
42	Christopher
43	Karin
44	Susan
45	Dave
46	Brian

Orders

OrderID	CustomerID	OrderDate
1	42	12/5/2012
2	42	12/6/2012
3	43	12/6/2012
4	43	12/7/2012
5	44	12/7/2012

OrderID	CustomerID	OrderDate	Name
1	42	12/5/2012	Christopher
2	42	12/6/2012	Christopher
3	43	12/6/2012	Karin
4	43	12/7/2012	Karin
5	44	12/7/2012	Susan
NULL	45	NULL	Dave
NULL	46	NULL	Brian

Self Joins

Join a table to itself

Generally a sign of poor design and normalization

Notes

- Tables must be aliased
- Need to filter rows matching themselves
  - Typically done with a non-equi join

Self Joins

Orders		
OrderID	ShipMethod	OrderDate
1	Express	12/5/2012
2	Express	12/6/2012
3	Standard	12/6/2012
4	Standard	12/7/2012
5	Overnight	12/7/2012

lhs.OrderID	ShipMethod	rhs.OrderID
1	Express	2
3	Standard	4

Cross Joins

All rows match from each table
- No ON statement

Also called a Cartesian Product

Uses
- Generating test data
- See what's "going on behind the scenes"

Cross Joins

Customers	
CustomerID	Name
42	Christopher
43	Karin
44	Susan

Orders		
OrderID	CustomerID	OrderDate
1	42	12/5/2012
2	42	12/6/2012
3	43	12/6/2012

OrderID	CustomerID	OrderDate	Name
1	42	12/5/2012	Christopher
2	42	12/6/2012	Christopher
3	42	12/6/2012	Christopher
1	43	12/5/2012	Karin
2	43	12/6/2012	Karin
3	43	12/6/2012	Karin
1	44	12/5/2012	Susan
2	44	12/6/2012	Susan
3	44	12/6/2012	Susan