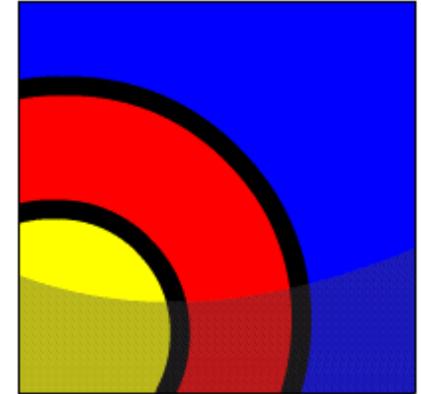


# Review of SQL Joins

## What Will I Learn?

In this lesson, you will review how to:

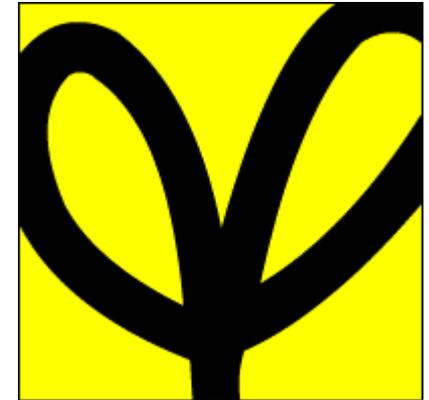
- Access data from more than one table using an equijoin
- Access data from more than one table using a nonequijoin
- Access data from more than one table using an outer join that results in a Cartesian product





## Why Learn It?

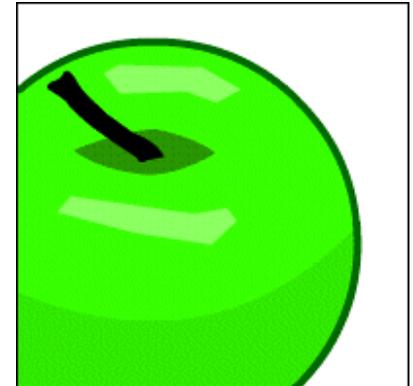
Going over previously learned material reinforces basic concepts and prepares you for more complicated constructs.



## Tell Me/Show Me

### Equijoin

An equijoin combines rows that have equal values for the specified columns. This is sometimes called a "simple" join. The basic syntax is:



```
SELECT table1.column, table2.column ...  
FROM table1, table2  
WHERE table1.column1 = table2.column2;
```



# Tell Me/Show Me

## Equijoin

The example shown joins two tables, `wf_world_regions` and `wf_countries`, to display the region name alongside the country name. Table aliases are used to shorten the syntax.

```
SELECT r.region_name, c.country_name, c.airports
FROM wf_world_regions r, wf_countries c
WHERE r.region_id = c.region_id
ORDER BY c.country_name;
```

REGION_NAME	COUNTRY_NAME	AIRPORTS
Caribbean	Anguilla	3
Oceania	Antarctica	28
Caribbean	Antiqua and Barbuda	3
Caribbean	Aruba	1



# Tell Me/Show Me

## Nonequijoin

Nonequijoin combines tables that have no exact matching columns. This join is often used when a column value in table A falls between a range of values specified by two columns in table B.

Examine the following tables:

```
SELECT employee_id,  
        last_name,salary  
FROM employees;
```

EMPLOYEE_ID	LAST_NAME	SALARY
100	King	24000
101	Kochlar	17000
102	De Haan	17000
103	Hunold	9000
104	Ernst	6000
107	Lorentz	4200

```
SELECT *  
FROM job_grades;
```

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000



# Tell Me/Show Me

## Nonequijoin

In this example, the grade for each student is matched up with the equivalent grade code.

```
SELECT e.employee_id, e.last_name, j.grade_level
FROM employees e, job_grades j
WHERE e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

EMPLOYEE_ID	LAST_NAME	GRADE_LEVEL
144	Vargas	A
143	Matos	A
142	Davies	B
141	Rajs	B
107	Lorentz	B
200	Whalen	B



## Tell Me/Show Me

### Outer Join

Outer join combines rows that have equivalent values for the specified columns plus those rows in one of the tables that have no matching value in the other table. To indicate which table might have missing data, use a plus sign (+) after the table's column name in the query.

```
SELECT table1.column, table2.column ...  
FROM table1, table2  
WHERE table1.column(+) = table2.column;  
or  
WHERE table1.column = table2.column(+);  
NEVER table1.column(+) = table2.column(+);
```



# Tell Me/Show Me

## Outer Join

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_name, d.department_id
FROM   employees e, departments d
WHERE  e.department_id = d.department_id (+)
ORDER BY NVL(e.department_id,1) ASC
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME	DEPARTMENT_ID
178	Grant	-	-	-
200	Whalen	10	Administration	10
202	Fay	20	Marketing	20
201	Hartstein	20	Marketing	20
124	Mourgos	50	Shipping	50
141	Rajs	50	Shipping	50
142	Davies	50	Shipping	50

## Tell Me/Show Me

### Cartesian Product

When a join query does not specify a condition in the `WHERE` clause, the result is a Cartesian product. This combines each row of one table with each row of the other resulting in many rows. For example, the Cartesian product of two tables, each with 100 rows, has 10,000 rows!

```
SELECT table1.column, table2.column ...  
FROM table1, table2;
```



# Tell Me/Show Me

## Cartesian Product

In the following example, every country name in the `wf_countries` table is paired with every population entry in the `wf_facts` table. This amounts to  $244 * 637 = 155,428$  rows!

```
SELECT country_name, language_id
FROM wf_countries, wf_spoken_languages;
```

COUNTRY_NAME	LANGUAGE_ID
United Arab Emirates	460
United Arab Emirates	1850
United Arab Emirates	460
United Arab Emirates	560
United Arab Emirates	900
United Arab Emirates	1650
United Arab Emirates	1650
United Arab Emirates	0
United Arab Emirates	460

# Tell Me / Show Me

## Terminology

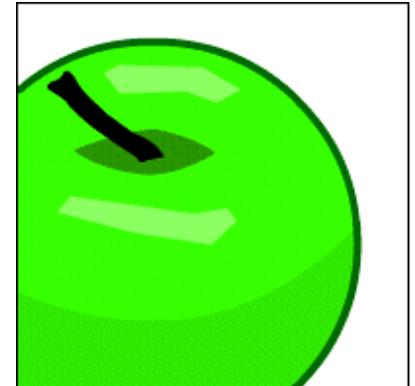
Key terms used in this lesson include:

Equijoin

Nonequijoin

Outer join

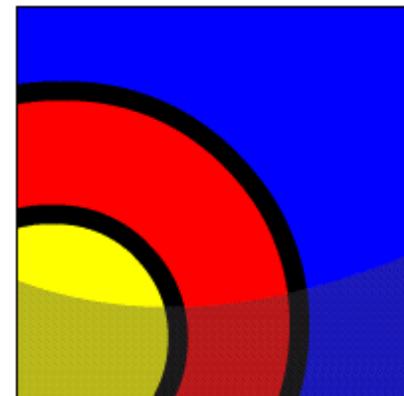
Cartesian product



## Summary

In this lesson, you reviewed how to:

- Access data from more than one table using an equijoin
- Access data from more than one table using a nonequijoin
- Access data from more than one table using an outer join that results in a Cartesian product



## Try It/Solve It

The exercises in this lesson cover the following types of joins in SQL:

- Equijoin
- Nonequijoin
- Outer join
- Cartesian product

