TECH
on the Net

SQL: Joins

An **SQL join** is used to combine rows from multiple tables. An SQL join is performed whenever two or more tables is listed in the SQL FROM clause of an SQL statement.

There are different kinds of SQL joins. Let's take a look at a few examples.

SQL Inner Join (simple join)

Chances are, you've already written an SQL statement that uses an SQL inner join. It is the most common type of SQL join. SQL inner joins return all rows from multiple tables where the join condition is met.

For example,

```
SELECT suppliers.supplier_id, suppliers.supplier_name, orders.order_date
FROM suppliers, orders
WHERE suppliers.supplier_id = orders.supplier_id;
```

This SQL inner join example would return all rows from the suppliers and orders tables where there is a matching supplier_id value in both the suppliers and orders tables.

Let's look at some data to explain how SQL inner joins work:

We have a table called **suppliers** with two fields (supplier_id and supplier_ name). It contains the following data:

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

We have another table called **orders** with three fields (order_id, supplier_id, and order_date). It contains the following data:

order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13

If we run the SQL statement (that contains an inner join) below:

```
SELECT suppliers.supplier_id, suppliers.supplier_name, orders.order_date
FROM suppliers, orders
WHERE suppliers.supplier_id = orders.supplier_id;
```

Our result set would look like this:

supplier_id	name	order_date
10000	IBM	2003/05/12
10001	Hewlett Packard	2003/05/13

The rows for *Microsoft* and *NVIDIA* from the supplier table would be omitted, since the supplier_id's 10002 and 10003 do not exist in both tables.

Outer Join

Another type of join is called an SQL outer join. This type of join returns all rows from one table and **only** those

rows from a secondary table where the joined fields are equal (join condition is met).

For example,

```
select suppliers.supplier_id, suppliers.supplier_name, orders.order_date
from suppliers, orders
where suppliers.supplier_id = orders.supplier_id(+);
```

This SQL outer join example would return all rows from the suppliers table and only those rows from the orders table where the joined fields are equal.

The (+) after the orders.supplier_id field indicates that, if a supplier_id value in the suppliers table does not exist in the orders table, all fields in the orders table will display as <null> in the result set.

The above SQL outer join statement could also be written as follows:

```
select suppliers.supplier_id, suppliers.supplier_name, orders.order_date
from suppliers, orders
where orders.supplier_id(+) = suppliers.supplier_id;
```

Let's look at some data to explain how SQL outer joins work:

We have a table called **suppliers** with two fields (supplier_id and name). It contains the following data:

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

We have a second table called **orders** with three fields (order_id, supplier_id, and order_date). It contains the following data:

order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13

If we run the SQL statement (that contains an outer join) below:

```
select suppliers.supplier_id, suppliers.supplier_name, orders.order_date
from suppliers, orders
where suppliers.supplier_id = orders.supplier_id(+);
```

Our result set would look like this:

supplier_id	supplier_name	order_date
10000	IBM	2003/05/12
10001	Hewlett Packard	2003/05/13
10002	Microsoft	<null>
10003	NVIDIA	<null>

The rows for *Microsoft* and *NVIDIA* would be included because an SQL outer join was used. However, you will notice that the order_date field for those records contains a <null> value.