Compare types of joins

Previously, you explored SQL joins and how to use them to join data from multiple tables when these tables share a common column. You also examined how there are different types of joins, and each of them returns different rows from the tables being joined. In this reading, you'll review these concepts and more closely analyze the syntax needed for each type of join.

Inner joins

The first type of join that you might perform is an inner join. *INNER JOIN* returns rows matching on a specified column that exists in more than one table.

Image not found or type unknown

Venn diagram with two circles labeled "left table" and "right table". The intersection is highligh

It only returns the rows where there is a match, but like other types of joins, it returns all specified columns from all joined tables. For example, if the query joins two tables with *SELECT **, all columns in both of the tables are returned.

Note: If a column exists in both of the tables, it is returned twice when *SELECT ** is used.

The syntax of an inner join

To write a query using *INNER JOIN*, you can use the following syntax:

```
SELECT *

FROM employees

INNER JOIN machines ON employees.device_id = machines.device_id;

SELECT thing1, thing2, thingX FROM table1 inner join table2 ON table1.commun_colomn =
 table2.common_colomn;
```

You must specify the two tables to join by including the first or left table after *FROM* and the second or right table after *INNER JOIN*.

After the name of the right table, use the *ON* keyword and the = operator to indicate the column you are joining the tables on. It's important that you specify both the table and column names in this portion of the join by placing a period (.) between the table and the column.

In addition to selecting all columns, you can select only certain columns. For example, if you only want the join to return the *username*, *operating_system* and *device_id* columns, you can write this query:

```
SELECT username, operating_system, employees.device_id

FROM  employees

INNER JOIN machines ON employees.device_id = machines.device_id;
```

it makes more sense for it to be all in one row for me so heres an explanation of each part how it works

```
SELECT thing_1, thing_2, thing_X FROM table1 INNER JOIN table2 ON table1.common_column =
 table2.common_column;
```

Note: In the example query, *username* and *operating_system* only appear in one of the two tables, so they are written with just the column name. On the other hand, because *device_id* appears in both tables, it's necessary to indicate which one to return by specifying both the table and column name (*employees.device_id*).

Outer joins

Outer joins expand what is returned from a join. Each type of outer join returns all rows from either one table or both tables.

Left joins

When joining two tables, *LEFT JOIN* returns all the records of the first table, but only returns rows of the second table that match on a specified column.

Venn diagram with two circles labeled "left table" and "right table". The left circle and the inte

Image not found or type unknown

The syntax for using *LEFT JOIN* is demonstrated in the following query:

```
SELECT *

FROM employees

LEFT JOIN machines ON employees.device_id = machines.device_id;
```

As with all joins, you should specify the first or left table as the table that comes after *FROM* and the second or right table as the table that comes after *LEFT JOIN*. In the example query, because *employees* is the left table, all of its records are returned. Only records that match on the *device_id* column are returned from the right table, *machines*.

Right joins

When joining two tables, *RIGHT JOIN* returns all of the records of the second table, but only returns rows from the first table that match on a specified column.

Venn diagram with two circles labeled "left table" and "right table". The right circle and the int

Image not found or type unknown

The following query demonstrates the syntax for *RIGHT JOIN*:

```
SELECT *

FROM employees

RIGHT JOIN machines ON employees.device_id = machines.device_id;
```

RIGHT JOIN has the same syntax as *LEFT JOIN*, with the only difference being the keyword *RIGHT JOIN* instructs SQL to produce different output. The query returns all records from *machines*, which is the second or right table. Only matching records are returned from *employees*, which is the first or left table.

Note: You can use *LEFT JOIN* and *RIGHT JOIN* and return the exact same results if you use the tables in reverse order. The following *RIGHT JOIN* query returns the exact same result as the *LEFT JOIN* query demonstrated in the previous section:

```
SELECT *

FROM machines

RIGHT JOIN employees ON employees.device_id = machines.device_id;
```

All that you have to do is switch the order of the tables that appear before and after the keyword used for the join, and you will have swapped the left and right tables.

Full outer joins

FULL OUTER JOIN returns all records from both tables. You can think of it as a way of completely merging two tables.

Venn diagram with two circles labeled "left table" and "right table". Both circles are highlighte

Image not found or type unknown

You can review the syntax for using *FULL OUTER JOIN* in the following query:

```
SELECT *

FROM employees

FULL OUTER JOIN machines ON employees.device_id = machines.device_id;
```

The results of a *FULL OUTER JOIN* query include all records from both tables. Similar to *INNER JOIN*, the order of tables does not change the results of the query.

Key takeaways

When working in SQL, there are multiple ways to join tables. All joins return the records that match on a specified column. *INNER JOIN* will return only these records. Outer joins also return all other records from one or both of the tables. *LEFT JOIN* returns all records from the first or left table, *RIGHT JOIN* returns all records from the second or right table, and *FULL OUTER JOIN* returns all records from both tables.

Revision #3
Created 10 July 2023 14:32:33 by naruzkurai
Updated 10 July 2023 16:20:26 by naruzkurai