SQL DML. Part 3.

Joins

Joins

CITY UNIVERSITY LONDON

The join operation puts rows of tables together so that you can retrieve data from more than one table

```
SELECT *
FROM mccMatches
INNER JOIN mccGround
ON mccMatches.ground_name = mccGround.ground_name;
```

```
SELECT *
FROM mccMatches T1
INNER JOIN mccGround T2
ON T1.ground_name = T2.ground_name;
```

More clear form

- The join condition, “`ON T1.ground_name = T2.ground_name`”, tells the DBMS which rows to join together (concatenates them).
- It only concatenates rows where the join condition is true, in this case where the primary key (in `mccGround`) and foreign key (in `mccMatches`) values are identical.
- The output of the query is a single table which is the product of the input tables.

Specified Columns

CITY UNIVERSITY LONDON
EST 1894

If you don't want all the columns in the output, specify the columns you do want

```
SELECT opposing_team, g_town
FROM mccMatches T1
INNER JOIN mccGround T2
ON T1.ground_name = T2.ground_name;
```

or

```
SELECT opposing_team, T1.ground_name, g_town
FROM mccMatches T1
INNER JOIN mccGround T2
ON T1.ground_name = T2.ground_name;
```

Cartesian Product

CITY UNIVERSITY LONDON
EST 1894

When there are no conditions and we just want to have all the combination of pairs.

```
SELECT * FROM mccMatches T1, mccGround T2;
```

You can use a condition on top of that and get the same results as Join.

```
SELECT * FROM mccMatches T1, mccGround T2
WHERE T1.ground_name = T2.ground_name;
```

BUT! It is much slower, since it builds the product and then filters.
Do not use it instead of join!

Examples

CITY UNIVERSITY LONDON

Join Player and Match_performance tables.

```
SELECT name, T2.match_date, batting_score
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
WHERE T1.registration_number = T2.registration_number;
```

Examples

CITY UNIVERSITY LONDON

Now, only players whose first name is “Michael”.

```
SELECT name, T2.match_date, batting_score
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
WHERE T1.registration_number = T2.registration_number AND T1.name
LIKE "Michael %";
```

Examples

CITY UNIVERSITY LONDON

Now, only players over the age of 35.

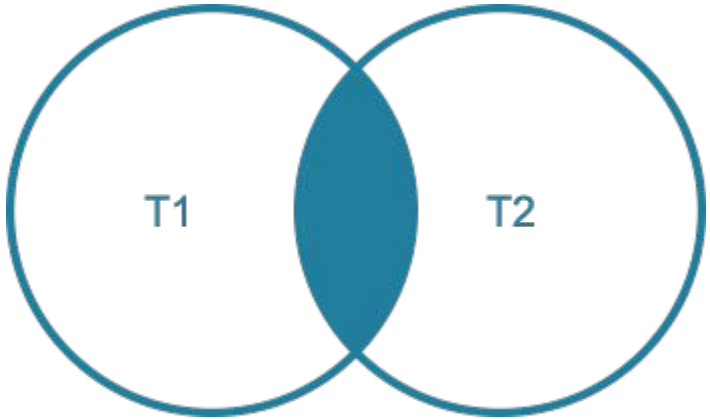
```
SELECT name, age, T2.match_date, batting_score
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
WHERE T1.registration_number = T2.registration_number AND T1.age
>= 35;
```

INNER JOIN

CITY UNIVERSITY LONDON
EST 1894

Inner Join takes all the pairs of rows from the left table and the right table on some condition.

If a row from the left table does not match to any in the right table, it is omitted. The same with a row in the right table.



Three (and more) Table Join

CITY UNIVERSITY LONDON
EST 1894

```
SELECT name, batting_score, ground_name
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
ON T1.registration_number = T2.registration_number
INNER JOIN mccMatches T3
ON T2.match_date = T3.match_date;
```

Three (and more) Table Join

CITY UNIVERSITY LONDON
EST 1894

Now, show only with batting score more than 100.

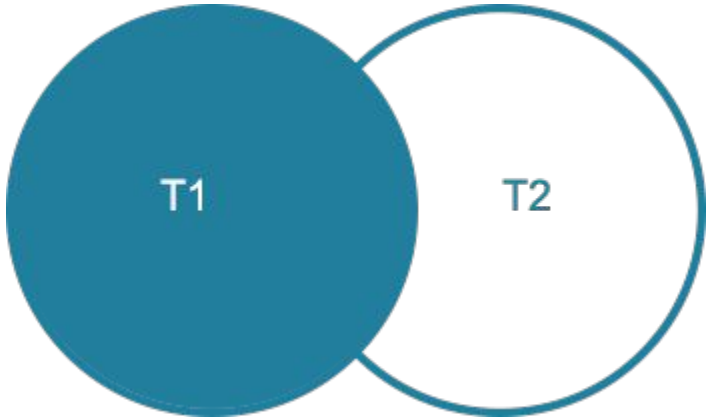
```
SELECT name, batting_score, ground_name
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
ON T1.registration_number = T2.registration_number
INNER JOIN mccMatches T3
ON T2.match_date = T3.match_date
WHERE T2.batting_score > 100;
```

Outer Joins

CITY UNIVERSITY LONDON
EST 1894

The LEFT JOIN clause allows you to select rows from the both left and right tables that match, plus all rows from the left table (T1) even though there is no match found for them in the right table (T2).

Analogously, RIGHT JOIN.



Left Join

CITY UNIVERSITY
LONDON

```
SELECT name, batting_score, T2.registration_number
FROM mccPlayer T1
LEFT JOIN mccMatch_performance T2
ON T1.registration_number = T2.registration_number;
```

Self Join. Example.

CITY UNIVERSITY LONDON

Find the players who live in the same town as “Hashim Amla”.

The following query will find from which town Hashim Amla is.

```
SELECT p_town
FROM mccPlayer
WHERE name = 'Hashim Amla';
```

Self Join. Example.

CITY UNIVERSITY LONDON
EST 1894

```sql
SELECT T2.name, T2.p_street
FROM mccPlayer T1
INNER JOIN mccPlayer T2
ON T1.p_town = T2.p_town
WHERE T1.name = 'Hashim Amla';
```

We use a usual inner join

Note that two aliases for mccPlayer, T1 and T2, are needed: T1 to find the town value and T2 to search for other players who live in the same town. It is also needed for the output.

In Part 4 we will learn how to do this query efficiently. However, self-joins are usual thing.

Self Join. Example.

CITY UNIVERSITY LONDON

Which other Ground is in the same town as the 'Leaside' ground?

```
SELECT T2.ground_name, T2.g_street
FROM mccGround T1
INNER JOIN mccGround T2
ON T1.g_town = T2.g_town AND T1.ground_name = 'Leaside';
```

Union

CITY UNIVERSITY LONDON

Find the names of all towns, for both grounds and players.

It automatically unifies the chosen tuples.

```
SELECT p_town
FROM mccPlayer
UNION
SELECT g_town
FROM mccGround;
```

Union. Example.

CITY UNIVERSITY LONDON

Find the players from Alnwick and with batting_score bigger than 100.

```
SELECT name, p_street
FROM mccPlayer T1
INNER JOIN mccMatch_performance T2
ON T1.registration_number = T2.registration_number
WHERE batting_score > 100
UNION
SELECT DISTINCT name, p_street
FROM mccPlayer
WHERE p_town = 'Alnwick';
```

Union Compatability

CITY UNIVERSITY LONDON

- To perform a union operation, the corresponding columns from each input table must have exactly the same data type.
- The column names do not have to be the same.