Module 2-3

Joins

Keys

In a relational database, all rows must be unique. The column or combination of columns that make it unique are referred to as **key(s)**.

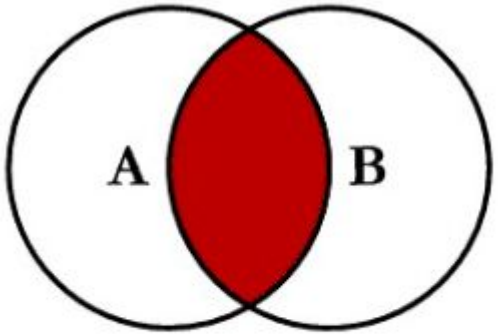
- **Primary Key**: column or columns in a table that uniquely identify the row. These cannot be duplicated.
  - If you say that SSN is your key, there cannot be more than one row with the same SSN.
- **Foreign Key**: A primary key present on another table.

Joins

Joins in SQL allow us to pull in data from several tables.

Joins : Inner Join

An inner join returns the rows in Table A that has a matching key value in Table B, the Venn Diagram representation is as follows:




```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Joins : Inner Join Example

Consider the following example: **I want the country name and capital city name.**

- Note that the country table has a numeric field called capital
- By design, this number corresponds to the id field of the city table.

We can therefore join these two tables:

```
select
A.name,
B.name,
B.population
FROM country A
JOIN city B ON A.capital = B.id;
```

Log | 1: country [232]

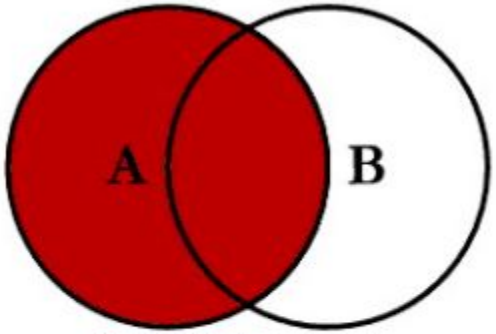
*	name	name	population
1	Afghanistan	Kabul	1780000
2	Netherlands	Amsterdam	731200
3	Netherlands Antilles	Willemstad	2345
4	Albania	Tirana	270000
5	Algeria	Alger	2168000
6	American Samoa	Fagatogo	2323
7	Andorra	Andorra la Vella	21189
8	Angola	Luanda	2022000
9	Anguilla	The Valley	595

Format: Text | 0.011/0.001 sec | 232/3 | 1-10

Let’s write some joins!

Joins : Left Outer Join

The Left Outer Join returns all the rows on the “left” side table of the join, it will attempt to match to the right side. If there is match… If it can’t find a match it includes it in the result, but with NULL values.




```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

Joins : Left Outer Join Example

For this example, **I have manually removed all the rows on the countrylanguage table for China and Switzerland**.

```
SELECT A.name, B.language
FROM COUNTRY A
LEFT OUTER JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna
4	Switzerland	(null)
5	China	(null)

Note that the country codes for China and Switzerland don't exist, so the Left Outer Join instead creates these NULL placeholders.

Joins : Left Outer Join vs Inner Join

With the same data set as the previous slide, let's compare the LEFT OUTER vs an INNER.

```
SELECT A.name, B.language
FROM COUNTRY A
JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna

```
SELECT A.name, B.language
FROM COUNTRY A
LEFT OUTER JOIN countrylanguage B
ON A.code = B.countrycode
WHERE name IN ('Switzerland', 'China', 'Belize');
```

*	name	language
1	Belize	English
2	Belize	Maya Languages
3	Belize	Garifuna
4	Switzerland	(null)
5	China	(null)

With the INNER JOIN, the rows for which there are no matches on the key are dropped from the final result set.

Unions

A union is a combination of two result sets. The following pattern is used:

[SQL Query 1]

UNION

[SQL Query 2]

Unions Example

Consider the following query:

```
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Danish'
UNION
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Swedish'
ORDER BY countrycode;
```

*	countrycode	language	percentage
1	DNK	Danish	93.5
2	FRO	Danish	0.0
3	NOR	Danish	0.4
4	GRL	Danish	12.5
5	DNK	Swedish	0.3
6	SWE	Swedish	89.5
7	NOR	Swedish	0.3
8	FIN	Swedish	5.7

Result of query first query (language = ‘Danish’)

Result of query second query (language = ‘Danish’)

Union All

Suppose we changed the previous to only return the language instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```

*	language
1	Danish
2	Swedish

Note that the query only returns the unique values associated with countrylanguage

Union All

In situations like this, we can override this behavior by specifying UNION ALL instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION ALL
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```

*	language
1	Danish
2	Danish
3	Danish
4	Danish
5	Swedish
6	Swedish
7	Swedish
8	Swedish

Duplicates are now not taken into account.