Bases de datos y SQL para no desarrolladores

¿Qué es una base de datos relacional?

Podemos pensar en una base de datos como una estructura de datos que almacena información organizada.

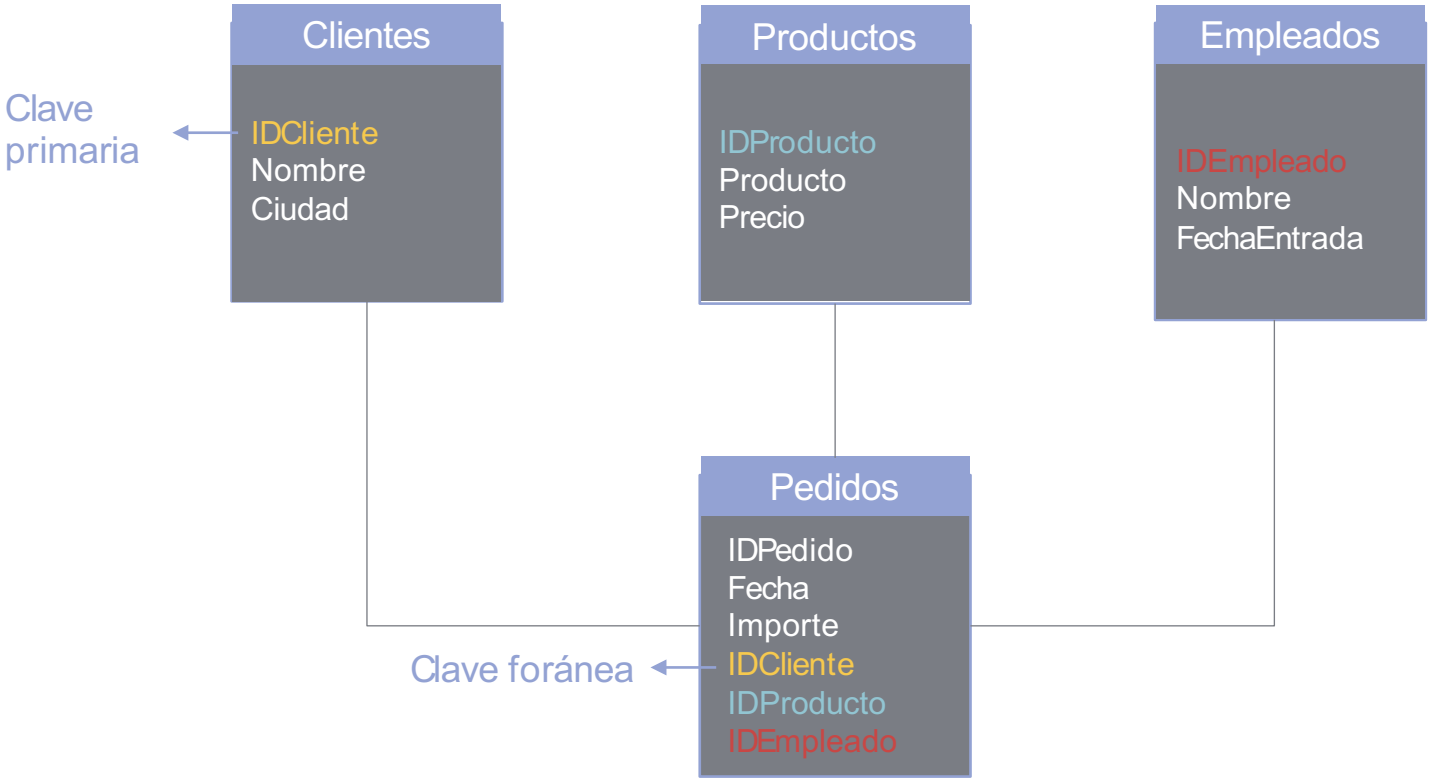
La mayoría de las bases de datos contienen varias tablas, cada una de las cuales puede incluir varios campos diferentes. Por tanto, podemos pensar en una base de datos relacional como un conjunto de tablas interrelacionadas.

Ejemplo

Por ejemplo, la base de datos de una empresa puede incluir tablas para productos, empleados y pedidos. Cada una de estas tablas tendría diferentes campos que son relevantes para la información almacenada en la tabla.

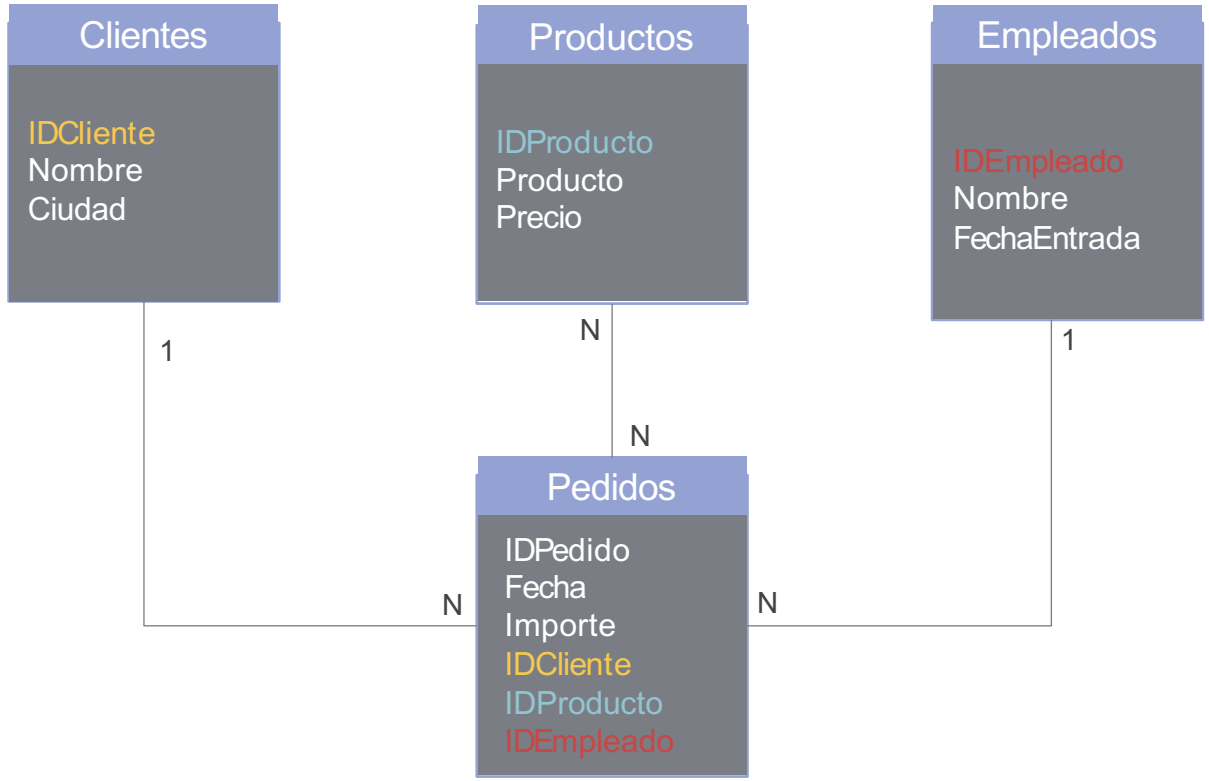
¿Cómo relacionamos las tablas?

- Relacionamos las tablas a través de las **claves**



¿Cómo relacionamos las tablas?

- Relacionamos las tablas a través de las **claves**



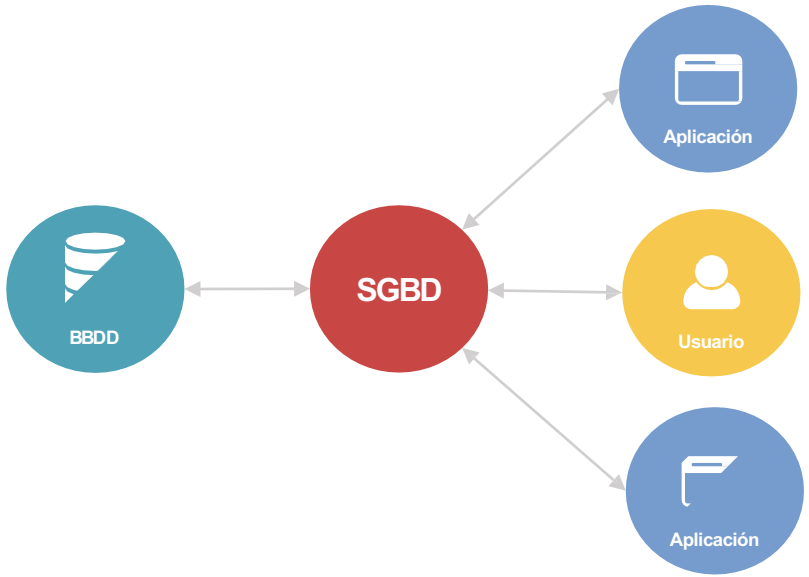
Nota

Las relaciones pueden ser:

- **1 a 1** – un registro se asocia a uno y solo un registro de otra tabla
- **1 a N** (muchos) – un registro se puede asociar a uno o varios registros de otra tabla
- **N a N** – varios registros de una tabla se asocian a varios registros de otra tabla

¿Cómo podemos trabajar con BBDD?

Los gestores de bases de datos (SGBD o DBMS Database management systems) son un software que actúa como enlace entre los usuarios finales, las aplicaciones y la propia base de datos.



Nota

Algunos de los principals SGDB de bases de datos relacionales son:

- Microsoft SQL Server
- MySQL
- PostgreSQL
- Oracle Database:
- SQLite

Nota

Existen también bases de datos no relacionales (NoSQL), en las que nose require estructura de datos fijas. Normalmente son empleadas en entornos distribuidos donde se gestiona un importante volumen de datos.

¿Qué es y para qué sirve SQL?

- La programación **SQL** permite interactuar con una base de datos. Los comandos SQL nos permiten actualizar, obtener y calcular información en bases de datos relacionales.
- Principales comandos SQL
  - **Consulta** de datos: SELECT, FROM
  - **Filtrado** de datos: WHERE, AND, OR, NOT
  - **Agregación** de datos: GROUP BY, HAVING, SUM, COUNT
  - **Agrupación** de datos: JOIN, LEFT JOIN, RIGHT JOIN

Nota

SQL recibe el nombre por las siglas en inglés *Structured Query Language*, es decir, Lenguaje de consulta Estructurada

Consulta de datos

SELECT & FROM

- **SELECT** – nos permite seleccionar qué campos de la tabla queremos
- **FROM** – indica desde qué tabla queremos consultar los datos

```
SELECT
        columna1,
        columna2,

        columnaN
FROM
        nombreTabla
```

Nota

Se utiliza * para seleccionar todos los campos de una tabla:

```
SELECT *
FROM nombreTabla
```

Consulta de datos

SELECT & FROM

Consulta

```
1 SELECT *
2 FROM `bigquery-public-data.world_bank_global_population.population_by_country`
3 LIMIT 1000
```

Resultado

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.4 sec elapsed, 125.4 KB processed)

Job information | Results | JSON | Execution details

Row	country	country_code	year_1960	year_1961	year_1962	year_1963	year_1964	year_1965	year_1966	year_1967	year_196
1	Sint Maarten (Dutch part)	SXM	null	null	null	null	null	null	null	null	
2	Not classified	INX	null	null	null	null	null	null	null	null	
3	West Bank and Gaza	PSE	null	null	null	null	null	null	null	null	
4	Serbia	SRB	null	null	null	null	null	null	null	null	
5	Cambodia	KHM	5722370	5872966	6028431	6183584	6331449	6467197	6585035	6685960	677
6	United Arab Emirates	ARE	92418	100796	112118	125130	138039	149857	159976	169771	182
7	Cyprus	CYP	572930	576395	577691	577913	578625	580966	585309	591308	598
8	Lower middle income	LMC	928490499	949753699	971742838	994420244	1017725630	1041611332	1066074552	1091126634	1116788

Nota

Se utiliza **LIMIT** para delimitar el número de resultados que devuelve la consulta. Es útil emplearlo en tablas con muchos registros, puesto que de lo contrario la consulta puede demorarse.

Consulta de datos

SELECT & FROM

Consulta

```
1 SELECT country, country_code, year_2018
2 FROM `bigquery-public-data.world_bank_global_population.population_by_country`
3 LIMIT 1000
```

Resultado

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 7 KB processed)

Job information | Results | JSON | Execution details

Row	country	country_code	year_2018
1	Sint Maarten (Dutch part)	SXM	40654
2	Not classified	INX	*null*
3	West Bank and Gaza	PSE	4569087
4	Serbia	SRB	6982084
5	Cambodia	KHM	16249798
6	United Arab Emirates	ARE	9630959
7	Cyprus	CYP	1189265
8	Lower middle income	LMC	3022905169

Consulta de datos
ORDER BY

- **ORDER BY** – nos permite ordenar los resultados de la consulta

Consulta

```
1 SELECT country, country_code, year_2018
2 FROM `bigquery-public-data.world_bank_global_population.population_by_country`
3 ORDER BY country ASC
4 LIMIT 1000
```

Resultado

SAVE RESULTS EXPLORE DATA

Query complete (0.4 sec elapsed, 7 KB processed)

Job information | Results | JSON | Execution details

Row	country	country_code	year_2018
1	Afghanistan	AFG	37172386
2	Albania	ALB	2866376
3	Algeria	DZA	42228429
4	American Samoa	ASM	55465
5	Andorra	AND	77006
6	Angola	AGO	30809762
7	Antigua and Barbuda	ATG	96286
8	Arab World	ARB	419790588

Sintaxis

```
SELECT
        columna1,
        columna2
        …
        columnaN
FROM
        nombreTabla

ORDER BY columna1,
columna2, …, columnaN
ASC | DESC;
```

Consulta de datos
WHERE

- **WHERE** – nos permite filtrar los datos de la tabla

Consulta

```
1 SELECT country, country_code, year_2018
2 FROM `bigquery-public-data.world_bank_global_population.population_by_country`
3 WHERE country = "Spain"
4 LIMIT 1000
```

Resultado

Query complete (0.4 sec elapsed, 7 KB processed)

Job information | Results | JSON | Execution details

Row	country	country_code	year_2018
1	Spain	ESP	46723749

Sintaxis

```
SELECT    columna1,
          columna2
          …
          columnaN
FROM      nombreTabla
WHERE     condición
```

Consulta de datos

WHERE: OR

- **OR** – devuelve las filas que cumplen una condición U otra

Consulta

```
SELECT country, country_code, year_2018
FROM `bigquery-public-data.world_bank_global_population.population_by_country`
WHERE country = "Spain" OR country = "Germany"
LIMIT 10
```

Resultado

Fila	country	country_code	year_2018
1	Germany	DEU	82927922
2	Spain	ESP	46723749

Sintaxis

```
SELECT
        columna1,
        columna2
        …
        columnaN
FROM
        nombreTabla

WHERE   condición1
        OR
        condición2
        OR
        condiciónN
```

Consulta de datos

WHERE: BETWEEN

- **BETWEEN** – selecciona valores que están dentro de un rango. Los dos extremos se incluyen

Consulta

```
SELECT country, country_code, year_2018
FROM `bigquery-public-data.world_bank_global_population.population_by_country`
WHERE year_2018 BETWEEN 40000000 AND 50000000
ORDER BY year_2018 DESC
LIMIT 100
```

Resultado

	country_code	year_2018	
1	Colombia	COL	49648685
2	Spain	ESP	46723749
3	Ukraine	UKR	44622516
4	Argentina	ARG	44494502
5	Uganda	UGA	42723139
6	Algeria	DZA	42228429
7	Sudan	SDN	41801533
8	Small states	SST	40575321

Sintaxis

```
SELECT
        columna1,
        columna2
        …
        columnaN
FROM
        nombreTabla

WHERE   BETWEEN
        valor1
        AND
        valor2
```

Consulta de datos

WHERE: AND, NOT

- **AND:** devuelve las filas que cumplen una condición Y otra
- **NOT:** devuelve las filas que NO cumplen la condición

Consulta

```
SELECT country, country_code, year_2018
FROM `bigquery-public-data.world_bank_global_population.population_by_country`
WHERE year_2018 BETWEEN 40000000 AND 50000000 AND NOT country = "Spain"
ORDER BY year_2018 DESC
LIMIT 100
```

Resultado

	country	country_code	year_2018
1	Colombia	COL	49648685
2	Ukraine	UKR	44622516
3	Argentina	ARG	44494502
4	Uganda	UGA	42723139
5	Algeria	DZA	42228429
6	Sudan	SDN	41801533
7	Small states	SST	40575321

Sintaxis

```
SELECT
        columna1,
        columna2
        …
        columnaN
FROM
        nombreTabla

WHERE   condición1
        AND
        condición2
        AND
        condiciónN
```

Agregación de datos
GROUP BY

- **GROUP BY** – agrega las filas que cumplen una cierta condición
- Suele ser empleado con funciones de agregación (SUM, COUNT, MAX, MIN, AVG)

Consulta

```
SELECT
  cal_year,
  incident_group,
  COUNT (incident_number) AS incidencias
FROM
  `bigquery-public-data.london_fire_brigade.fire_brigade_service_calls`
GROUP BY
  cal_year, incident_group
LIMIT 1000
```

Resultado

Fila	cal_year	incident_group	incidencias
1	2017	False Alarm	15732
2	2017	Fire	6434
3	2017	Special Service	10081

Agregación de datos
GROUP BY

Consulta

```
SELECT
  year,
  major_category,
  COUNT (*) AS numeroCrimenes
FROM
  `bigquery-public-data.london_crime.crime_by_lsoa`
WHERE year BETWEEN 2015 AND 2016
GROUP BY
    year, major_category
ORDER BY numeroCrimenes DESC
LIMIT 1000
```

Resultado

Fila	year	major_category	numeroCrimenes
1	2016	Theft and Handling	440700
2	2015	Theft and Handling	440700
3	2016	Violence Against the Person	352416
4	2015	Violence Against the Person	352416
5	2015	Criminal Damage	229908
6	2016	Criminal Damage	229908
7	2016	Drugs	131052
8	2015	Drugs	131052
9	2016	Burglary	115956
10	2015	Burglary	115956

Agregación de datos
GROUP BY

Consulta

```
SELECT
  year,
  major_category,
  minor_category,
  SUM (value) AS sumaValor,
  COUNT (minor_category) AS numeroCrimenes
FROM
  `bigquery-public-data.london_crime.crime_by_lsoa`
WHERE year = 2016 AND major_category = "Theft and Handling"
GROUP BY
    year, major_category, minor_category
ORDER BY sumaValor DESC
LIMIT 1000
```

Resultado

Fila	year	major_category	minor_category	sumaValor	numeroCrimenes
1	2016	Theft and Handling	Other Theft	103807	58020
2	2016	Theft and Handling	Theft From Motor Vehicle	51319	58020
3	2016	Theft and Handling	Theft From Shops	46957	46308
4	2016	Theft and Handling	Other Theft Person	34868	57720
5	2016	Theft and Handling	Theft/Taking Of Motor Vehicle	26366	58008
6	2016	Theft and Handling	Theft/Taking of Pedal Cycle	18001	57444
7	2016	Theft and Handling	Motor Vehicle Interference & Tampering	11438	57828
8	2016	Theft and Handling	Handling Stolen Goods	1377	47352

Agregación de datos
HAVING

- **HAVING** – filtra resultados de las funciones agregadas (e.g. SUM, COUNT, MAX, MIN, AVG)

Consulta

```
SELECT
  year,
  major_category,
  minor_category,
  SUM (value) AS sumaValor,
  COUNT (minor_category) AS numeroCrimenes
FROM
  `bigquery-public-data.london_crime.crime_by_lsoa`
WHERE
  year = 2016 AND major_category = "Theft and Handling"
GROUP BY
    year, major_category, minor_category
HAVING
  sumaValor > 30000
ORDER BY sumaValor DESC
LIMIT 1000
```

Resultado

Fila	year	major_category	minor_category	sumaValor	numeroCrimenes
1	2016	Theft and Handling	Other Theft	103807	58020
2	2016	Theft and Handling	Theft From Motor Vehicle	51319	58020
3	2016	Theft and Handling	Theft From Shops	46957	46308
4	2016	Theft and Handling	Other Theft Person	34868	57720

Agrupación de datos
INNER JOIN

- **JOIN** – comando utilizado para combinar filas de dos o más tablas. Ambas tablas deben tener una columna en común.

Tabla: Clientes

ClienteID*	NombreCliente	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla

Tabla: Pedidos

PedidoID*	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20

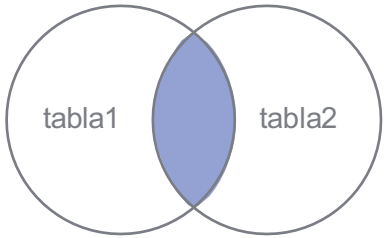
TU TURNO

¿Cuántos pedidos ha habido por ciudad?

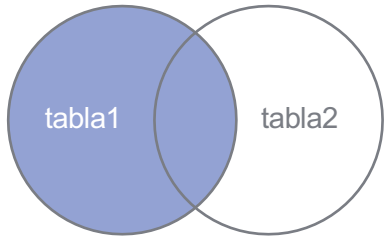
Agrupación de datos

Tipos de JOIN

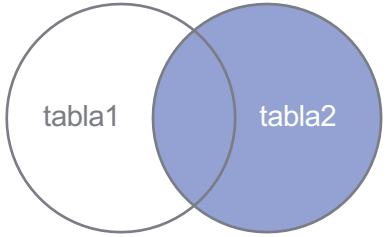
INNER JOIN



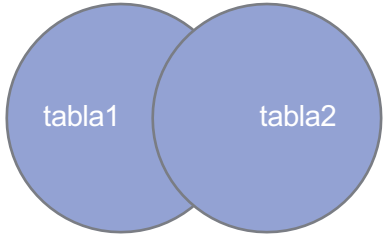
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



Agrupación de datos

INNER JOIN

Tabla: Clientes

ClienteID	NombreCliente	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla

Clientes.Ciudad

Tabla: Pedidos

PedidoID	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20

TU TURNO

¿Cuántos pedidos ha habido por ciudad?

CONSEJO

1. Identificar el campo que ambas tablas tienen en común
2. Elegir la tabla principal
3. Identificar el nombre de los campos que se quieren seleccionar

Agrupación de datos

INNER JOIN

Tabla: Clientes

ClienteID	Nombre	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla
4	Laura	Barcelona
5	Rocio	Alicante

Clientes.ClienteID

Clientes.Nombre

Clientes.Ciudad

Tabla: Pedidos

PedidoID	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20
104	6	16/12/20
105	7	16/12/20

Pedidos.PedidoID

Pedidos.ClienteID

TU TURNO

¿Cuántos pedidos ha habido por ciudad?

CONSEJO

1. Identificar el campo que ambas tablas tienen en común
2. Identificar el nombre de los campos que se quieren seleccionar

Agrupación de datos

INNER JOIN

```
SELECT
        Pedidos.PedidoID,
        Pedidos.ClienteID,
        Clientes.Nombre
        Clientes.Ciudad,
FROM
        Pedidos
INNER JOIN
        Clientes ON Pedidos.ClienteID = Clientes.ClienteID
```

PedidoID	ClienteID	Nombre	Ciudad
101	1	María	Valencia
102	3	Juan	Madrid
103	1	María	Valencia

CONSEJO

1. Identificar el campo que ambas tablas tienen en común
2. Elegir la tabla principal
3. Identificar el nombre de los campos que se quieren seleccionar

Agrupación de datos

INNER JOIN

Tabla: Clientes

ClienteID	Nombre	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla
4	Laura	Barcelona
5	Rocio	Alicante

Tabla: Pedidos

PedidoID	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20
104	6	16/12/20
105	7	16/12/20

EJEMPLO

```
SELECT
        Pedidos.PedidoID,
        Pedidos.ClienteID,
        Clientes.Nombre,
        Clientes.Ciudad
FROM
        Pedidos
INNER JOIN
        Clientes ON
Pedidos.ClienteID =
Clientes.ClienteID
```

Agrupación de datos

INNER JOIN

Tabla: Clientes

ClienteID	Nombre	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla
4	Laura	Barcelona
5	Rocio	Alicante

Tabla: Pedidos

PedidoID	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20
104	6	16/12/20
105	7	16/12/20

PedidoID	ClienteID
101	1
102	3
103	1

EJEMPLO

```
SELECT
        Pedidos.PedidoID,
        Pedidos.ClienteID,
        Clientes.Nombre,
        Clientes.Ciudad
FROM
        Pedidos
INNER JOIN
        Clientes ON
Pedidos.ClienteID =
Clientes.ClienteID
```

Agregación de datos

INNER JOIN

Tabla: Clientes

ClienteID	Nombre	Ciudad
1	María	Valencia
2	Pepe	Madrid
3	Juan	Sevilla
4	Laura	Barcelona
5	Rocio	Alicante

Tabla: Pedidos

PedidoID	ClienteID	FechaPedido
101	1	14/11/20
102	3	25/11/20
103	1	15/12/20
104	6	16/12/20
105	7	16/12/20

PedidoID	ClienteID	Nombre	Ciudad
101	1	María	Valencia
102	3	Juan	Madrid
103	1	María	Valencia

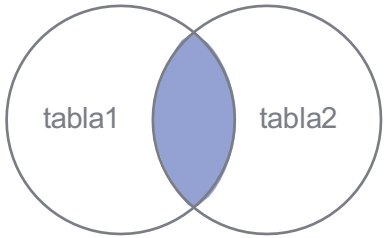
EJEMPLO

```
SELECT
        Pedidos.PedidoID,
        Clientes.ClienteID,
        Clientes.Nombre,
        Clientes.Ciudad
FROM
        Pedidos
INNER JOIN
        Clientes ON
Pedidos.ClienteID =
Clientes.ClienteID
```

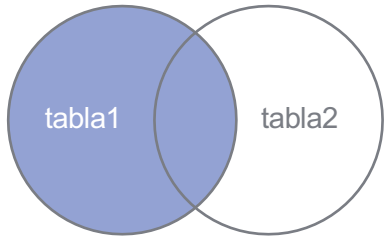
Agrupación de datos

Tipos de JOIN

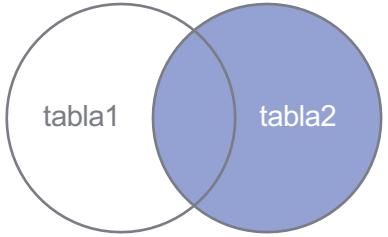
INNER JOIN



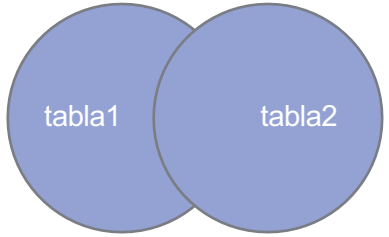
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



TU TURNO

¿Qué resultado devolvería cada uno de estos joins en el caso de las tablas de clientes y pedidos?