Gabriel Gómez
gomezcgabriel1998@gmail.com

¿Que es una base de datos ?

¿Que es una base de datos ?

Def. Es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema de gestión de base de datos (DBMS)

¿Que es una base de datos ?

Def. Es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema de gestión de base de datos (DBMS)

Def. Programa capaz de almacenar gran cantidad de datos, relacionados y estructurados, que pueden ser consultados rápidamente de acuerdo con las características selectivas que se deseen

Bases de datos Relacional (SQL)

- Datos organizados en tablas
- Tienen identificadores que conectan (relacionan) información entre tablas

Ventajas

- Empresas/organizaciones grandes y complejas
- Fácil manipulación de la información por medio de consultas (SQL)

Bases de datos No Relacional (NoSQL)

- Datos organizados en documentos, gráficas y sistemas de llave-valor
- No tienen identificadores que conectan (relacionan) información entre datos
- No se puede manipular los datos mediante SQL

Ventajas

- Esquemas de datos flexibles
- Buen rendimiento

Esquema de las BD

El esquema de las bases de datos es el como va almacenarse la información

Persona				
Identificacion	**Nombre**	**Apellido**	**Sexo**	**Telefono**
123	Alex	Garcia	M	8281111
456	Sofia	Sandoval	F	8282222
789	Lina	Peña	F	8283333
987	Maria	Arteaga	F	8284444
654	Alvaro	Perez	M	8285555
321	Andres	Gonzalez	M	8286666

Características

- Tabla
- Atributos
- Tuplas

¿Por qué son importantes las bases de datos?

https://padlet.com/gomezcgabriel1998/vozm001xfiy4zdfl

Llaves para relacionar tablas

LLaves primarias

Es un atributo o un conjunto de atributos que identifican una sola tupla en una tabla

Persona				
Identificacion	**Nombre**	**Apellido**	**Sexo**	**Telefono**
123	Alex	Garcia	M	8281111
456	Sofia	Sandoval	F	8282222
789	Lina	Peña	F	8283333
987	Maria	Arteaga	F	8284444
654	Alvaro	Perez	M	8285555
321	Andres	Gonzalez	M	8286666

Nombre	Apellido	Direccion	Apto	Telefono
Pedro	Perez	cll 105	201	3214
Daniel	Ramírez	cll 103	305	3659
Juanita	Gómez	cll 105	203	6985

Llaves para relacionar tablas

LLaves foráneas

- Es un atributo o un conjunto de atributos que son una llave primaria de otra tabla
- Establecen relación entre tablas compartiendo un atributo o un conjunto de atributos

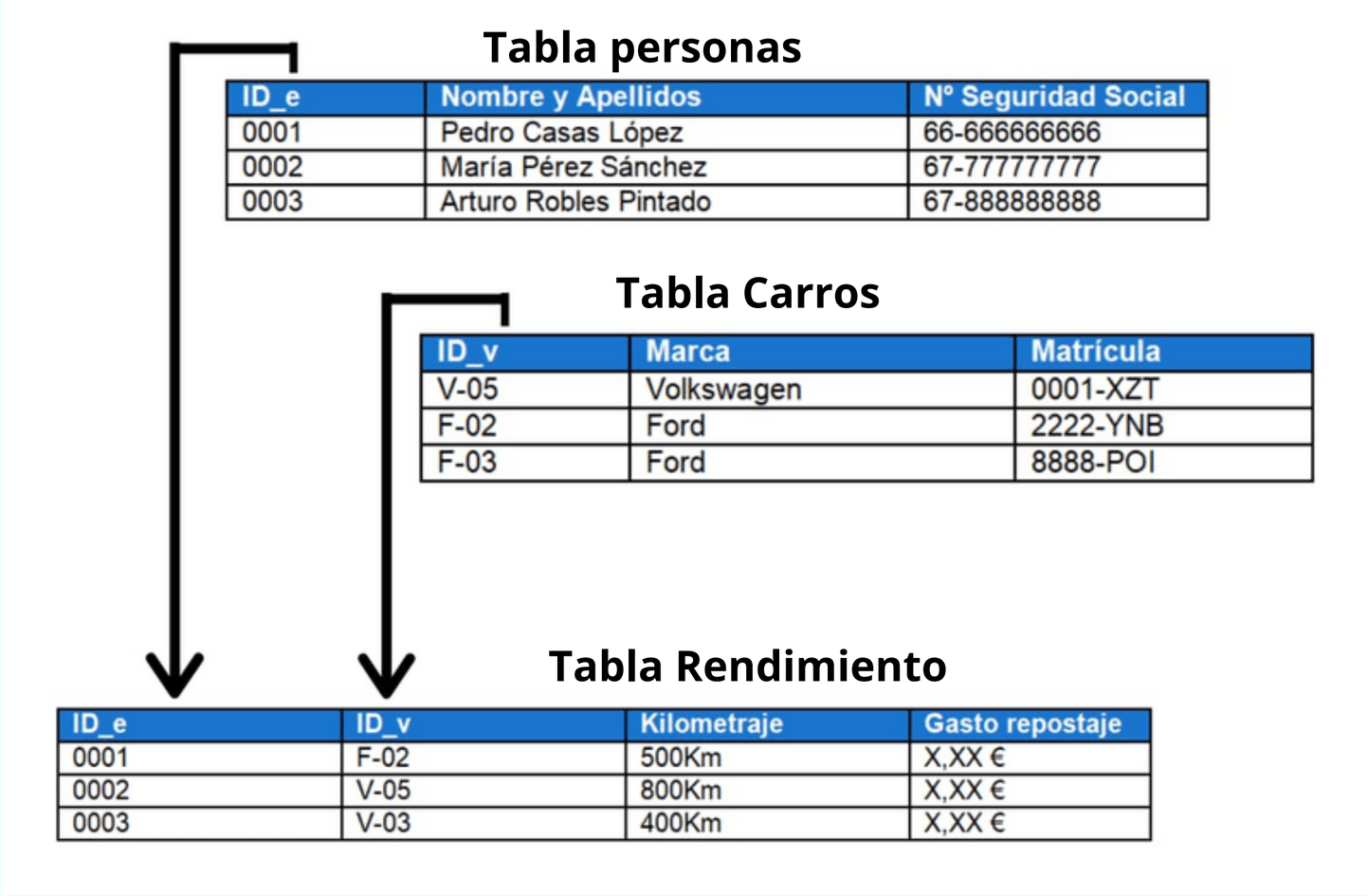


Tabla personas

ID_e	Nombre y Apellidos	Nº Seguridad Social
0001	Pedro Casas López	66-666666666
0002	María Pérez Sánchez	67-777777777
0003	Arturo Robles Pintado	67-888888888

Tabla Carros

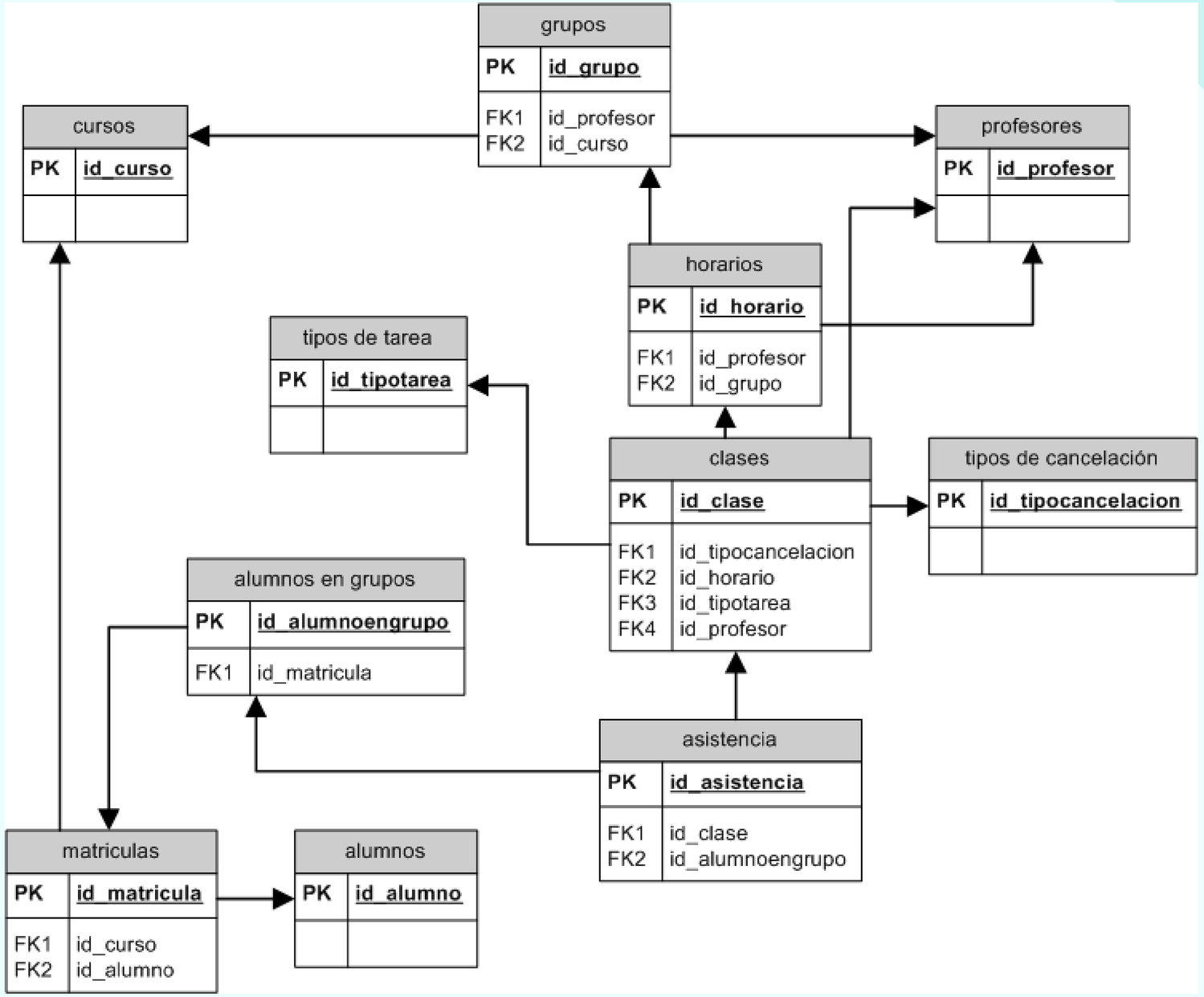
ID_v	Marca	Matrícula
V-05	Volkswagen	0001-XZT
F-02	Ford	2222-YNB
F-03	Ford	8888-POI

Tabla Rendimiento

ID_e	ID_v	Kilometraje	Gasto repostaje
0001	F-02	500Km	X,XX €
0002	V-05	800Km	X,XX €
0003	V-03	400Km	X,XX €

Llaves foráneas

- Llave primaria o primary key (PK)
- Llave foránea o foreign key (FK)



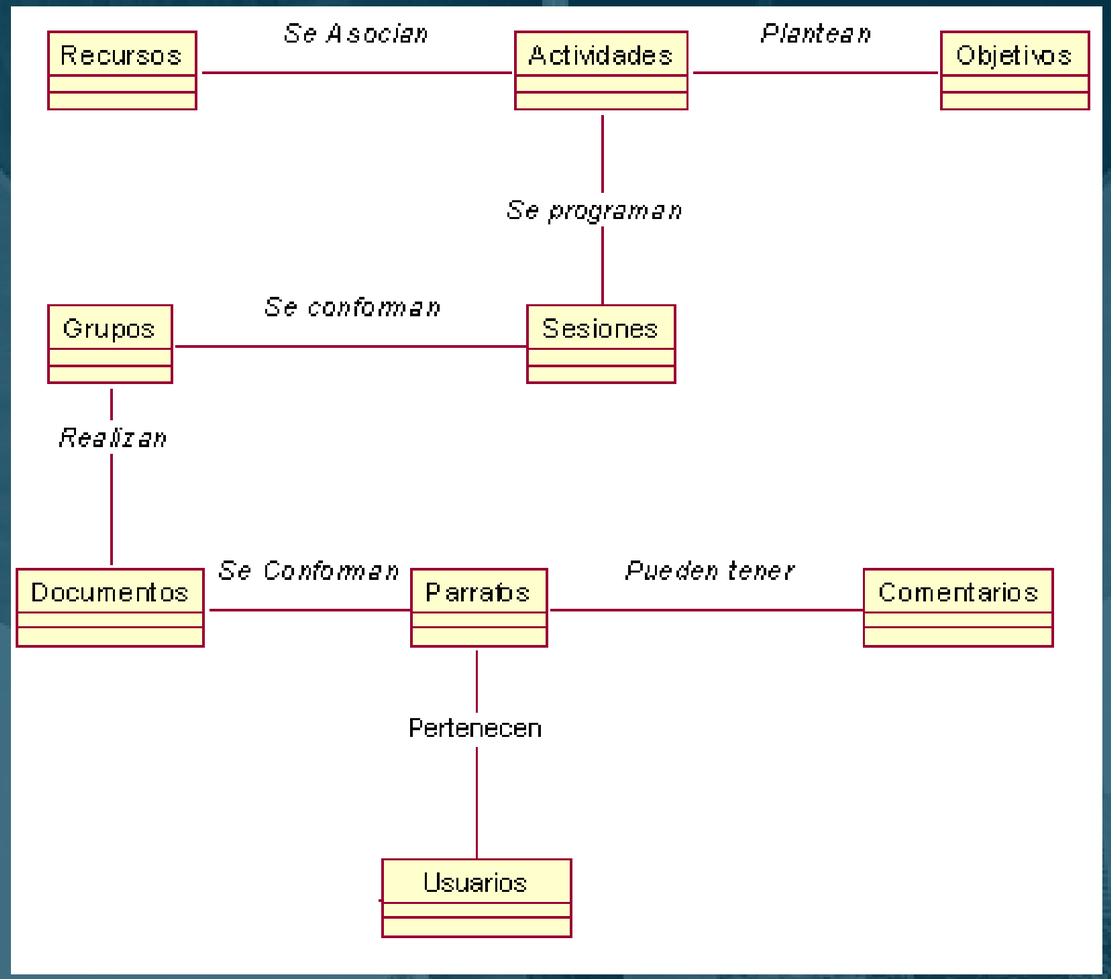
Modelo Conceptual

Objetivo: Identifcar las relaciones de mas alto nivel entre entidades

Características

- Se conectan las entidades (tablas) dependiendo de su relación
- No se especifican atributos
- No se muestran llaves primarias
- No se muestran llaves foraneas

Modelo Conceptual

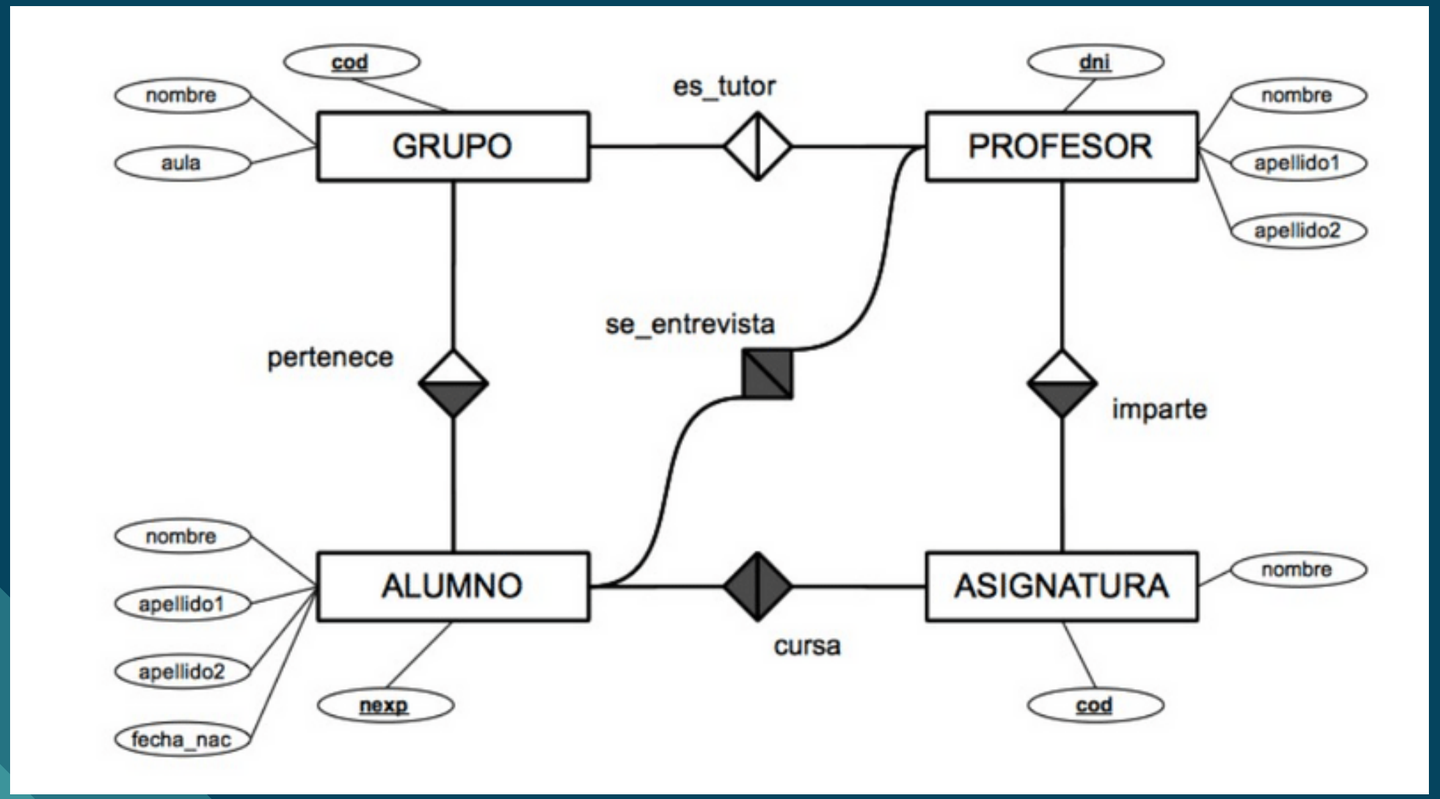


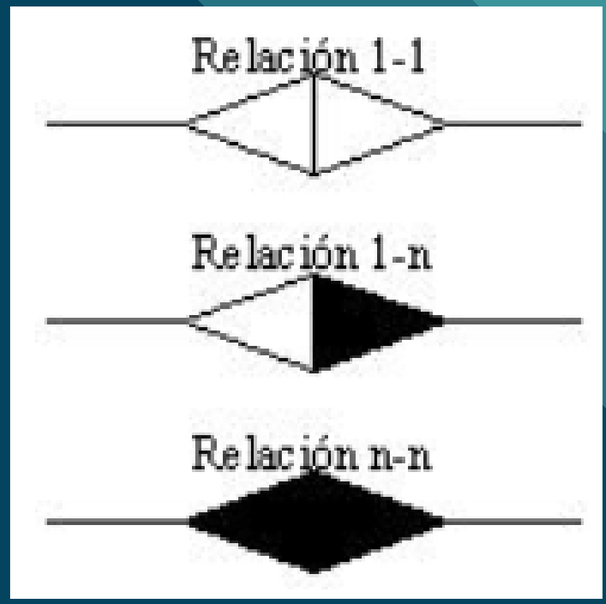
Modelo Entidad - Relación

Objetivo: Establecer las relaciones entre tablas en detalle y más especificado

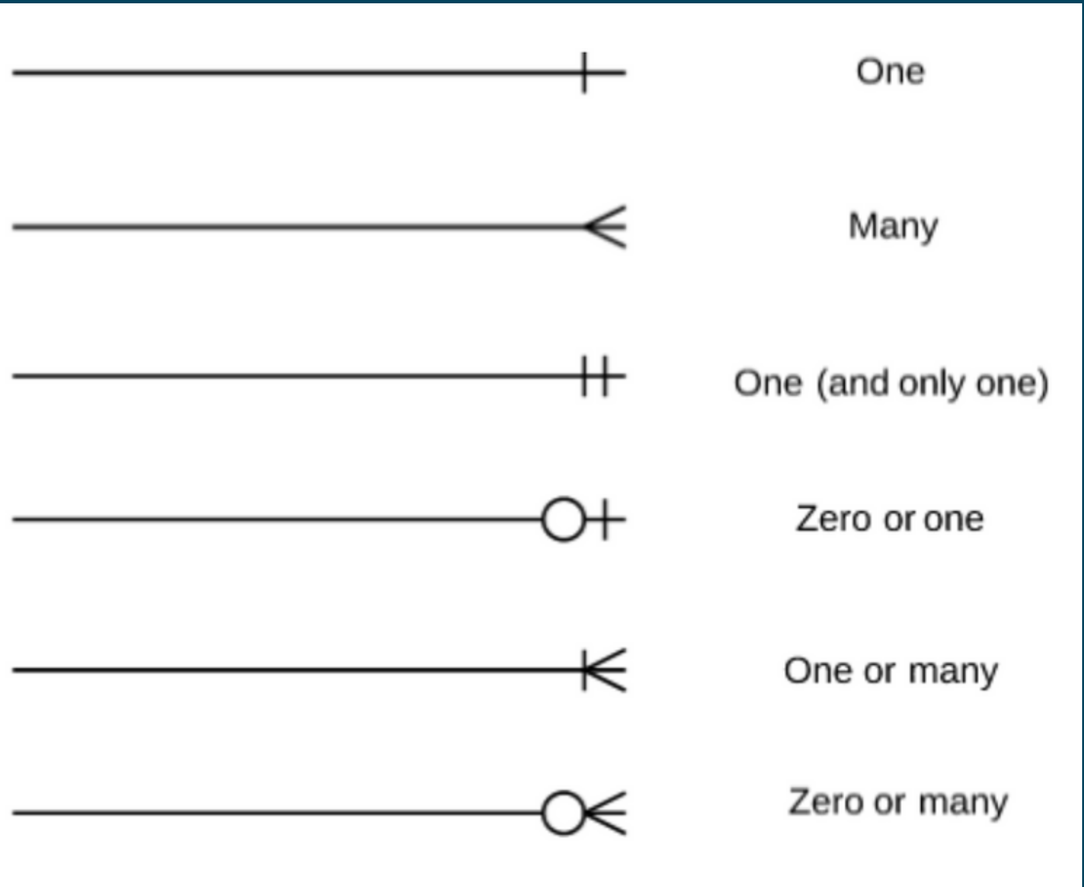
- Se conectan las entidades (tablas) dependiendo de su relación
- Se especifican atributos
- Se muestran llaves primarias
- Se muestran llaves foraneas

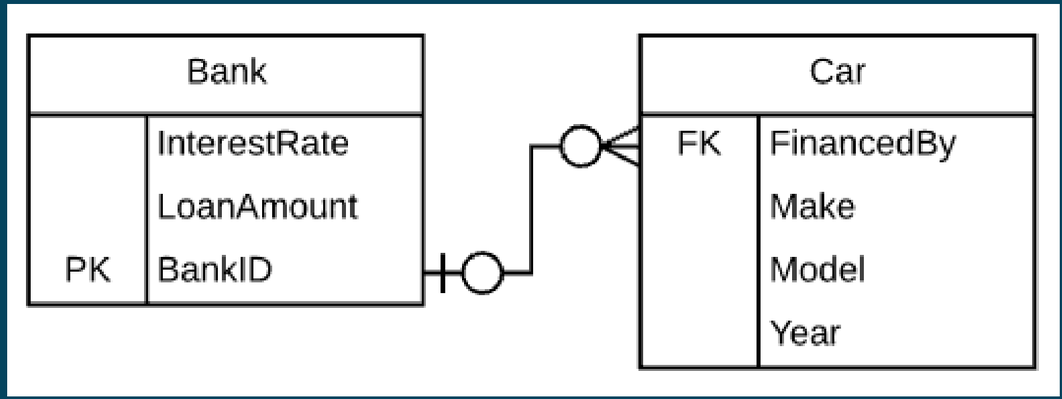
Modelo Entidad - Relación



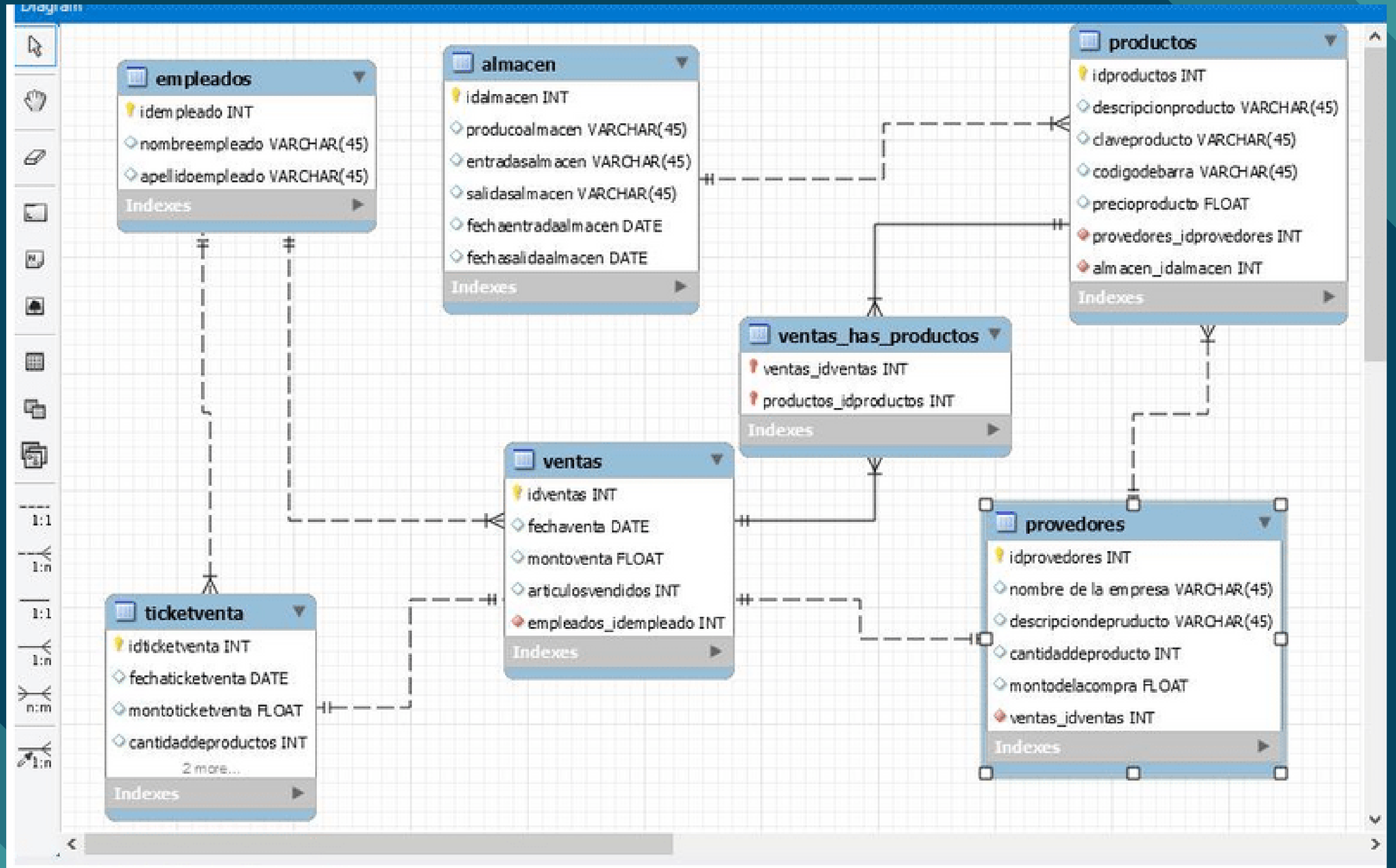


Modelo Entidad - Relación





Modelo Entidad - Relación



Ejercicio

Si vamos a gestionar los datos de nuestro restaurante,

- Que entidades podemos identificar?
- Podemos representarlas como tablas?
- Podemos asignar atributos a dichas tablas?
- Podemos relacionarlas?

SQL (Structured Query Language)

Que es ?

SQL (Structured Query Language)

Que es ?

Es un lenguaje de programación diseñado para **almacenar, manipular** y **recuperar** datos en una base de datos **relacional**.

Componentes

- Comandos
- Cláusulas
- Operadores (lógicos y comparación)
- Funciones

Tipos de comandos SQL

DDL

Lenguaje de definición de datos

Crean y definen:

- bases de datos
- campos
- índices

DML

Lenguaje de manipulación de datos

Generan consultas para:

- ordenar datos
- filtrar datos
- extraer datos

Comandos DDL y DML

COMANDOS TIPO DDL	
COMANDO	**DEFINICION**
CREATE	Crea nuevas bases de datos, tablas, campos e índices.
DROP	Elimina bases de datos, tablas, campos e índices.
ALTER	Modifica estructura, diseño de una base de datos

COMANDOS TIPO DML	
COMANDO	**DEFINICION**
SELECT	Consulta registros dentro de la base de datos
INSERT	Inserta o agrega datos en una tabla o entidad
DELETE	Elimina registros de una entidad bajo ciertos criterios.
UPDATE	Modifica o actualiza los datos existentes dentro de las entidades.

Cláusulas

Cláusulas	
Cláusula	**Descripción**
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

Operadores Lógicos

Operadores Lógicos	
Operador	**Uso**
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Operadores de comparación

Operadores de Comparación	
Operador	**Uso**
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Operador BETWEEN

Se utiliza para buscar dentro de un rango establecido

Consulta

```
SELECT * FROM `empleados`
WHERE `id_empleado` BETWEEN 1000 AND 1019
```

Resultado de registros

id_empleado	nombre	apellido	edad
1011	daniela	hernandez	50
1014	juan	rodriguez	35
1015	valentina	ramirez	28

Operador LIKE

Se utliliza para encontrar patrones o expresiones regulares

Consulta

```
SELECT * FROM `empleados`
WHERE `apellido` LIKE '%ez'
```

% -> Representa 0,1 o muchos caracteres

_ -> Representa solo 1 caracter

Resultado de registros

id_empleado	nombre	apellido	edad
1011	daniela	hernandez	50
1014	juan	rodriguez	35
1015	valentina	ramirez	28

Operador LIKE

También aplica en tipos de datos numéricos

Consulta

```
SELECT * FROM `empleados`
WHERE `id_empleado`
LIKE '10_5'
```

Resultado de registros

id_empleado	nombre	apellido	edad
1015	valentina	ramirez	28

En conjunto

Operador LIKE

Consulta

```
SELECT * FROM empleados
WHERE apellido LIKE '_end%'
```

Resultado de registros

id_empleado	nombre	apellido	edad
1020	rosana	mendoza	45

Operador IN

Se ultiliza para validar la existencia de un registro en un grupo en específico

Consulta

```
SELECT * FROM empleados
WHERE apellido IN ('rodriguez','mendoza','velasquez','aguirre')
```

id_empleado	nombre	apellido	edad
1014	juan	rodriguez	35
1020	rosana	mendoza	45
1012	laura	aguirre	36
1018	rodrigo	velasquez	34

AUTOINCREMENT

Se utiliza para automatizar la asignación de valores

- Por defecto, el autoincrement inicia en 1
- Al momento de insertar, no se debe especificar el campo que contiene autoincrement

ID	Nombre	Apellido
1	Gabriel	Gomez
2	Manuela	Peralta
3	Daniel	Davila
...	...	...

AUTOINCREMENT

```
CREATE TABLE personas(
id int NOT NULL AUTO_INCREMENT,
nombre varchar(255) NOT NULL,
apellido varchar(255),
edad int,
PRIMARY KEY (id)
);
```

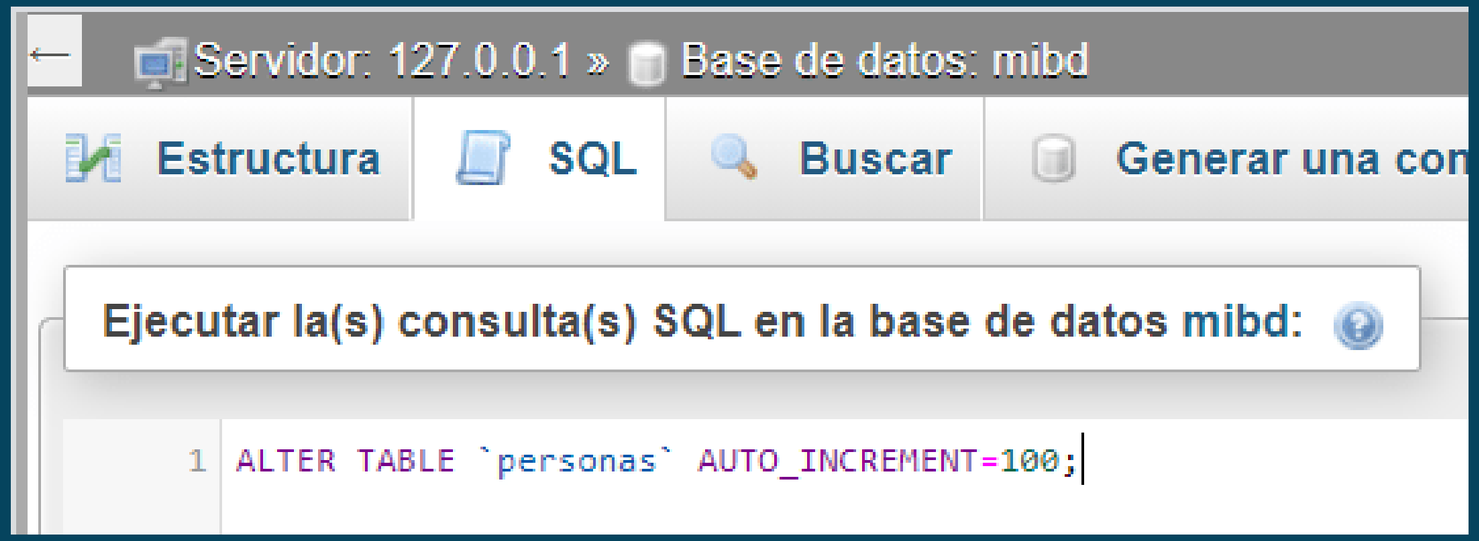
AUTOINCREMENT

En el insert into no se especifica el campo del autoincrement

```
1 INSERT INTO `personas`
2 (apellido,edad,nombre)
3 VALUES('gomez',15, 'gabriel');
```

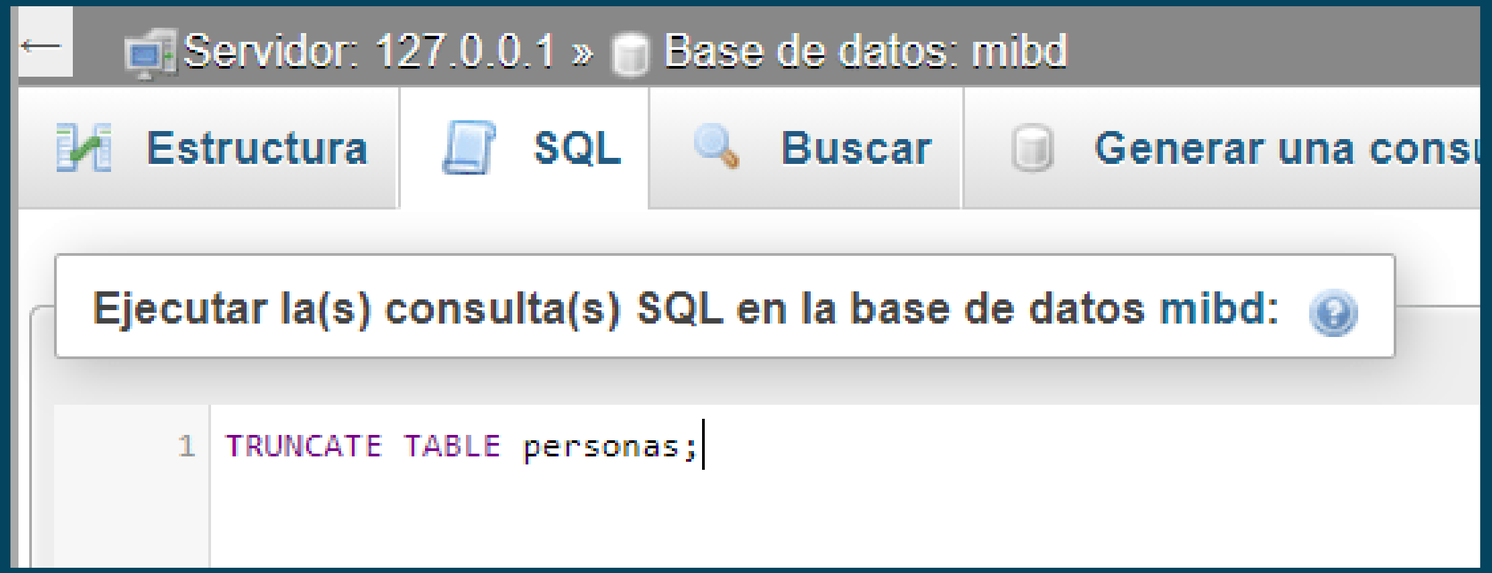
AUTOINCREMENT

- Por defecto empieza en 1
- Podemos modificar a nuestro gusto



TRUNCATE

Este comando se utiliza para borrar los registros de una tabla pero manteniendo su estructura



NORMALIZACIÓN

Objetivo

La normalización es un proceso de simplificación de datos

Características

- Tener almacenado los datos con el menor espacio posible
- Eliminar datos repetidos
- Eliminar errores lógicos
- Datos ordenados

NORMALIZACIÓN

Niveles

- Primera forma normal (1FN)
- Segunda forma normal (2 FN)
- Tercera forma normal (3 FN)
- Forma normal Boyce Codd (BC)
- Cuarta forma normal (4 FN)
- Quinta forma normal (5 FN)

NOTA

La simplificación debe darse sin que haya perdida de información

1 FN

Identificar los grupos de reptición

Matrícula	Nombre	Dirección	Teléfono	Materia	Num Materia	Carrera
1	Sergio	Puebla 22	56565656	Base de datos	123	Sistemas
1	Sergio	Puebla 22	56565656	Programación web	234	Sistemas
1	Sergio	Puebla 22	56565656	Programación visual	234	Sistemas
2	Ana	Reforma 1	23232323	Base de datos	123	Sistemas

1 FN

¿Que columnas están repitiendo datos?

Matrícula	Nombre	Dirección	Teléfono	Materia	Num Materia	Carrera
1	Sergio	Puebla 22	56565656	Base de datos	123	Sistemas
1	Sergio	Puebla 22	56565656	Programación web	234	Sistemas
1	Sergio	Puebla 22	56565656	Programación visual	234	Sistemas
2	Ana	Reforma 1	23232323	Base de datos	123	Sistemas

1 FN

¿Que columnas están repitiendo datos?

¿Que columnas no están repitiendo datos?

Matrícula	Nombre	Dirección	Teléfono	Materia	Num Materia	Carrera
1	Sergio	Puebla 22	56565656	Base de datos	123	Sistemas
1	Sergio	Puebla 22	56565656	Programación web	234	Sistemas
1	Sergio	Puebla 22	56565656	Programación visual	234	Sistemas
2	Ana	Reforma 1	23232323	Base de datos	123	Sistemas

1 FN

Repiten datos: Matricula, Nombre, Dirección, Teléfono, Carrera

No repiten datos: Materia , Num materia

Matrícula	Nombre	Dirección	Teléfono	Materia	Num Materia	Carrera
1	Sergio	Puebla 22	56565656	Base de datos	123	Sistemas
1	Sergio	Puebla 22	56565656	Programación web	234	Sistemas
1	Sergio	Puebla 22	56565656	Programación visual	234	Sistemas
2	Ana	Reforma 1	23232323	Base de datos	123	Sistemas

1 FN

Tabla alumnos

Matrícula	Nombre	Dirección	Teléfono	Carrera
1	Sergio	Puebla 22	56565656	Sistemas
2	Ana	Reforma 1	23232323	Sistemas

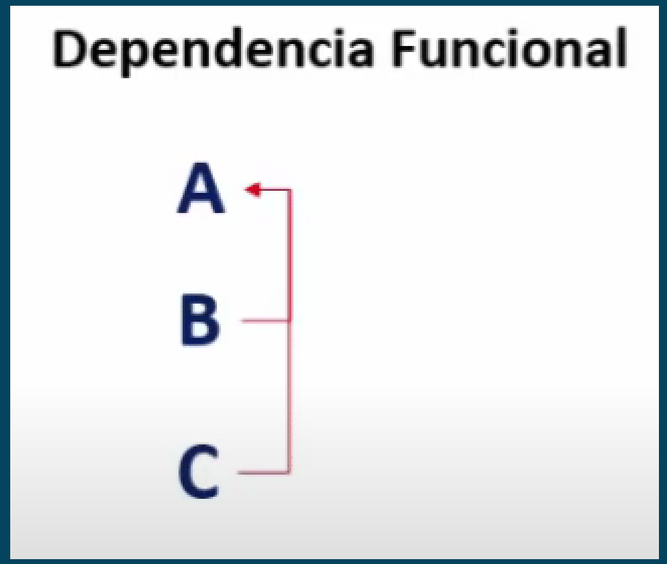
Nota: No se puede perder la relación

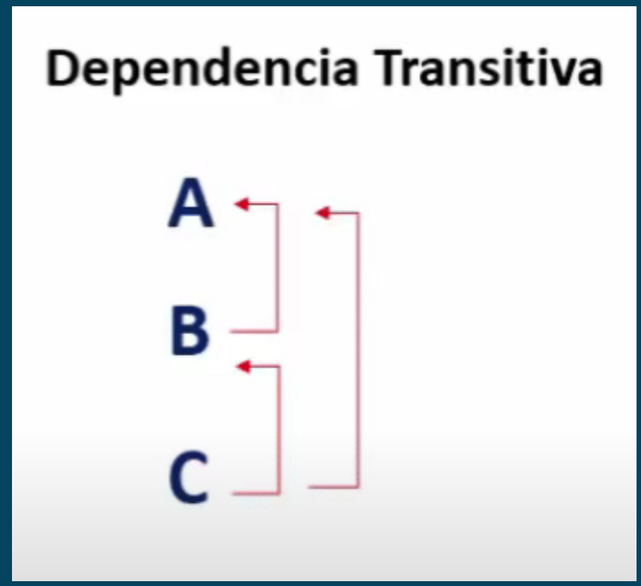
Tabla materias alumno

Matrícula	Materia	Num Materia
1	Base de datos	123
1	Programación web	234
1	Programación visual	234
2	Base de datos	123

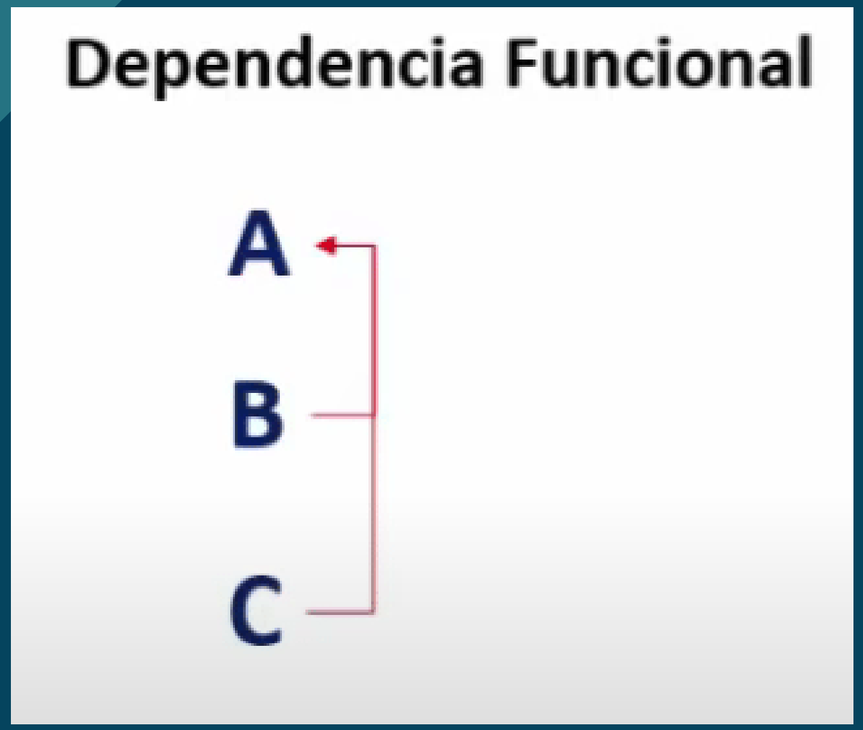
2 FN

- La tabla ya debe estar en primera forma normal (1 FN)
- Identificar las dependencias funcionales y transitivas





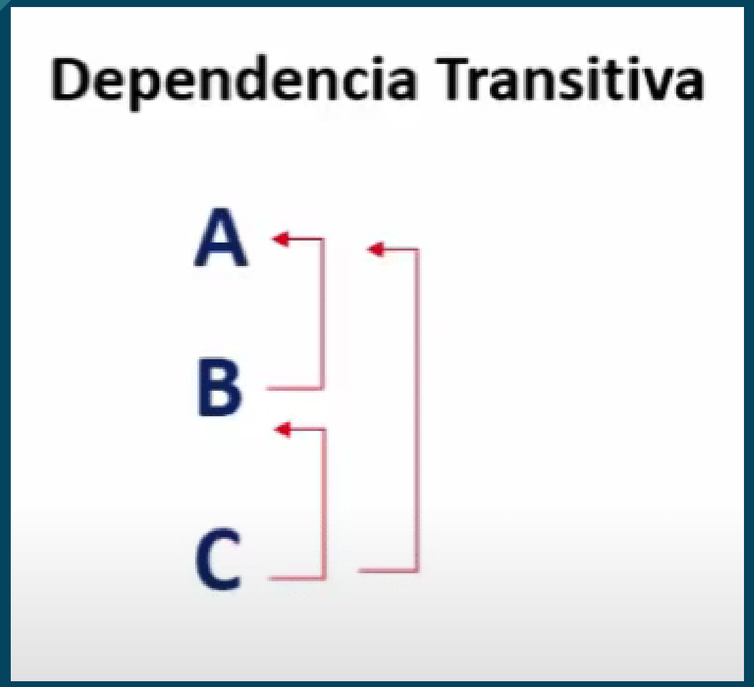
Dependencia funcional



El atributo A será llave primaria de los atributos B y C

Los atributos B y C dependerán funcionalmente de A

Dependencia transitiva



El atributo A será llave primaria de B

El atributo B será llave primaria de C

En base a este ejemplo:

¿Cuantas dependencias funcionales identifican?

¿Cuales son?

¿Cual será la relación entre A y C?

2 FN

Tabla alumnos

Matrícula	Nombre	Dirección	Teléfono	Carrera
1	Sergio	Puebla 22	56565656	Sistemas
2	Ana	Reforma 1	23232323	Sistemas

Tabla materias alumno

Matrícula	Materia	Num Materia
1	Base de datos	123
1	Programación web	234
1	Programación visual	234
2	Base de datos	123

En la segunda tabla:
Tiene dos llaves
Matricula como: llave ____
Num Materia como: llave ___

¿ Que dependencias identifican?

2 FN

Tabla alumnos

Matrícula	Nombre	Dirección	Teléfono	Carrera
1	Sergio	Puebla 22	56565656	Sistemas
2	Ana	Reforma 1	23232323	Sistemas

Se simplificaron los datos✔
Se eliminaron los datos repetidos✔
Se eliminaron los errores lógicos✔
Se mantuvo la integridad de la BD✔

Tabla alumno materia

Matrícula	Num Materia
1	123
1	234
1	234
2	123

Tabla materias

Materia	Num Materia
Base de datos	123
Programación web	234
Programación visual	234
Base de datos	123

Columnas calculadas

Objetivo: Visualizar nuevas columnas a partir de otras columnas

- Utilizando operadores matemáticos se crean los nuevos valores a la nueva columna

Columnas calculadas

Tabla original

id	nombre	edad	apellido
1	gabriel	22	gomez
2	daniel	25	ramirez
3	andrea	48	hernandez
4	juan	17	perez
5	laura	55	bonilla
6	pablo	21	mendez

Visualizando una columna calculada

id	nombre	apellido	edad	nueva_edad
1	gabriel	gomez	22	27
2	daniel	ramirez	25	30
3	andrea	hernandez	48	53
4	juan	perez	17	22
5	laura	bonilla	55	60
6	pablo	mendez	21	26

Columnas calculadas

Sintaxis

```
SELECT nombreProducto,precio,cantidad, (precio*cantidad) AS TOTAL  FROM `venta`
```

Funciones de cadena

Funciones para la manipulacion de datos de tipo cadena (varchar)

Función	Propósito
CHR(n)	Nos devuelve el carácter cuyo valor en binario es n
CONCAT(cad1, cad2)	Nos devuelve cad1 concatenada con cad2
UPPER(cad)	Convierte cad a mayúsculas
LOWER(cad)	Convierte cad a minúsculas
LPAD(cad1,n[,cad2])	Con esta función añadimos caracteres a cad1 por la izquierda hasta una longitud máxima dada por n
INITCAP(cad)	Convierte la primera letra de cad a mayúscula
LTRIM(cad [,set])	Elimina un conjunto de caracteres a la izquierda de cad, siendo set el conjunto de caracteres a eliminar
RPAD(cad1, n[,cad2])	Con esta función añadimos caracteres de la misma forma que con la función LPAD pero esta vez los añadimos a la derecha
RTRIM(cad[,set])	Hace lo mismo que LTRIM pero por la derecha
REPLACE(cad,cadena_buscada [,cadena_sustitucion])	Sustituye un conjunto de caracteres de 0 o más caracteres, devuelve cad con cada ocurrencia de cadena_buscada sustituida por cadena_sustitucion

Funciones matemáticas

Funciones para la manipulacion de datos de tipo numericos

- Funciones para una sola columna o un solo valor
- Estadísticas

Funciones matemáticas

Funciones para un solo valor

Función	Propósito
ABS(n)	Nos devuelve el valor absoluto de n
CEIL(n)	Nos devuelve el valor entero igual o inmediatamente superior a n
FLOOR(n)	Nos devuelve el valor entero igual o inmediatamente inferior a n
MOD(m,n)	Nos devuelve el resto de la división de m entre n
POWER(m, exponente)	Calcula la potencia de m elevado a exponente
SIGN(valor)	Nos devuelve el signo de valor
NVL(valor, expresión)	Función que nos sustituye valor por expresión siempre que valor sea NULL
ROUND(número[, m])	Nos redondea numero a m decimales
SQRT(n)	Nos devuelve la raíz cuadrada de n
TRUNC(número[,m])	Trunca los números para que tengan m decimales.

Funciones matemáticas

Ejemplo: función techo

```
SELECT nombreProducto, CEIL(precio) FROM `venta`
```

Ejemplo: función piso

```
SELECT nombreProducto, FLOOR(precio) FROM `venta`
```

Ejemplo: raiz cuadrada

```
SELECT nombreProducto, SQRT(precio) FROM `venta`
```

Ejemplo: redondear valor a m decimales

```
SELECT nombreProducto, ROUND(precio,3) FROM `venta`
```

Funciones matemáticas

Estadísticas

Función	Propósito
AVG(n)	Nos devuelve la media de n
COUNT(*|expresión)	Nos devuelve el número de veces que aparece expresión.
MAX(expresión)	Nos devuelve el valor máximo de expresión
MIN (expresión)	Nos devuelve el valor mínimo de expresión
VARIANCE(expresión)	Nos devuelve la varianza de expresión
SUM(expresión)	Nos devuelve la suma de valores de expresión.

Funciones matemáticas

Promedio de los precios en la tabla venta

```
SELECT AVG(precio) FROM `venta`
```

Contar cuantos precio son superiores a 20

```
SELECT COUNT(precio) FROM `venta` WHERE precio > 20
```

Precio maximo de las ventas

```
SELECT MAX(precio) FROM `venta`
```

Precio maximo de las ventas inferiores a 80

```
SELECT MAX(precio) FROM `venta` WHERE precio < 80
```

Agrupamiento

Objetivo: agrupar en un solo registro información de un valor en específico

cedula	idPlato	precio	ingredientes
1019	1	15.5	3
1019	2	25.3	5
1019	3	10.1	2
1020	1	15.5	3
1020	2	25.3	5
1020	4	23.6	5
1021	4	23.6	5
1021	1	15.5	3

Agrupamiento

Agrupar por cedula de chef, saber cuantos platos realiza

```
SELECT cedula, COUNT(idPlato) AS numPlatos  FROM `platoschefs` WHERE precio > 10 GROUP BY cedula
```

formas de filtrar

Claúsula WHERE

```
SELECT cedula, COUNT(idPlato) AS numPlatos FROM `platoschefs` WHERE precio > 10 GROUP BY cedula
```

Claúsula HAVING

```
SELECT cedula, COUNT(idPlato) AS numPlatos  FROM `platoschefs` GROUP BY cedula HAVING cedula > 1019
```

HAVING es muy similar a la cláusula WHERE, pero en vez de afectar a las filas de la tabla, afecta a los grupos obtenidos.

Agrupamiento

Agrupar por cedula de chef, saber cuantos platos realiza y la suma total de sus precios

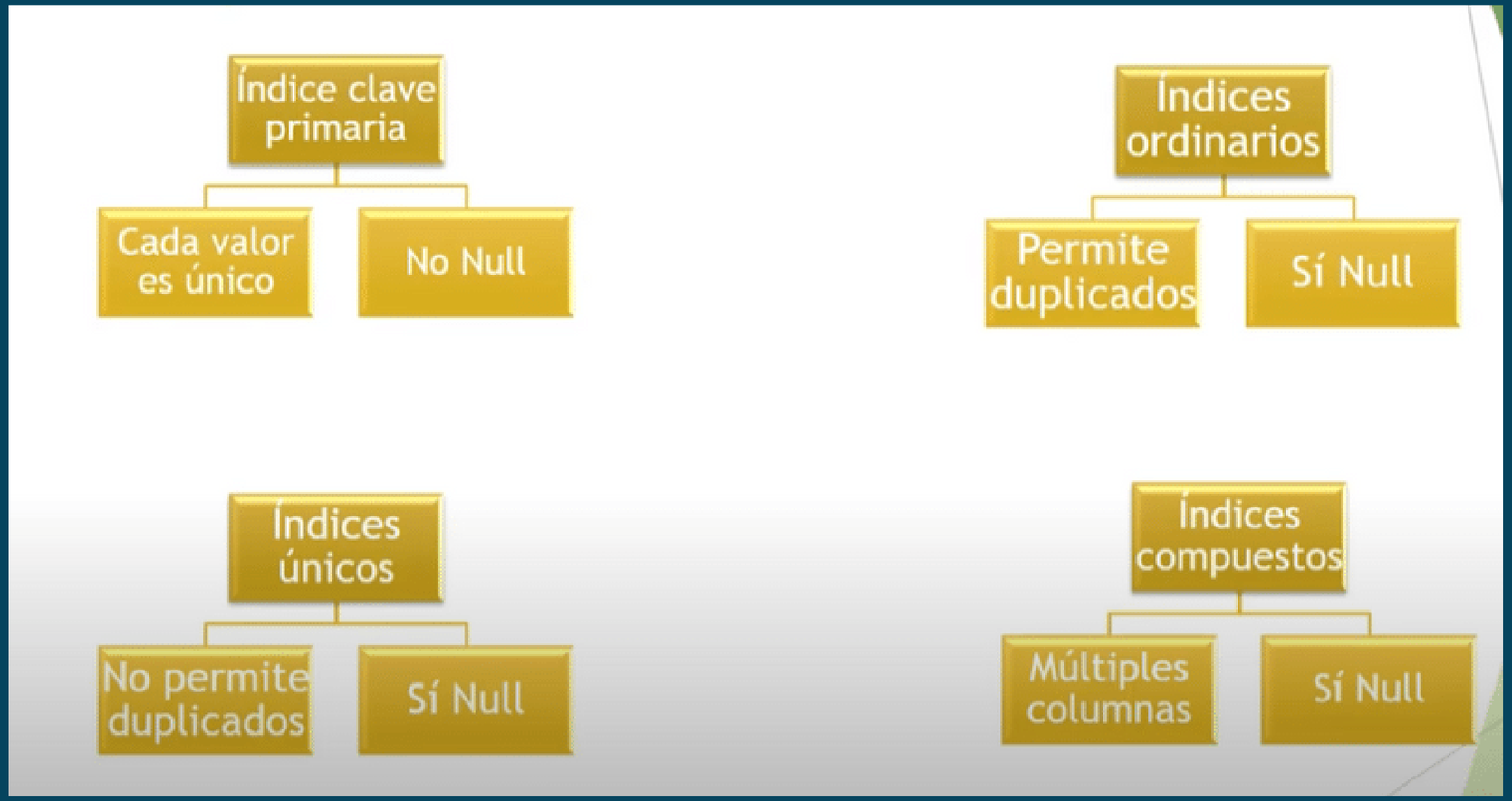
```
SELECT cedula, precio, COUNT(idPlato)  AS numPlatos , SUM(precio) AS valorTotal  FROM `platoschefs` GROUP BY cedula
```

Misma consulta pero el valor total con 2 decimales

```
SELECT cedula, precio, COUNT(idPlato)  AS numPlatos , ROUND(SUM(precio),2) AS valorTotal  FROM `platoschefs` GROUP BY cedula
```

índices

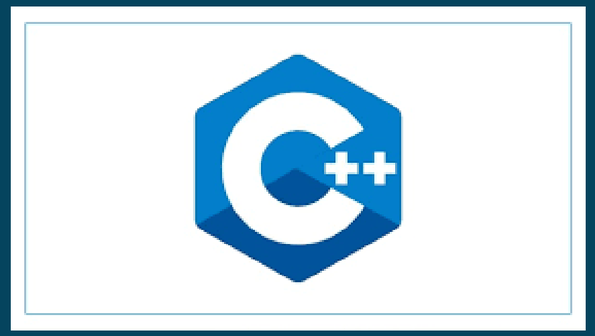
Los índices en SQL nos permiten aumentar la velocidad de respuesta de las consultas



índices

¿Qué son exactamente?

Son punteros que hacen referencia al valor que tiene cada uno de los registros en una BD



índices

índice ordinario

Sintaxis para crear índice

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Sintaxis para eliminar índice

```
DROP INDEX index_name ON table_name;
```

índices

índice ordinario

Una vez teniendo establecidos los índices, podemos realizar una consulta para observar como quedan ordenados los registros

SELECT nombreProducto FROM venta

Claúsula LIMIT

- Esta claúsula nos permite especificar cuantos registros obtendremos de una consulta con la claúsula SELECT
- Tambien especificar cuantos registros eliminaremos cuando realizamos un DELETE

Sintaxis

```
SELECT * FROM `platoschefs` LIMIT 1,5
```

Si no se especifica el primer parámetro se tomarán los registros desde el 0 por defecto

Funciones de control

- Las funciones de control nos permite establecer condicionales en nuestras consultas
- Alteran el flujo de la consulta según las respuestas obtenidas

Funciones de control

- Sintaxis

```
case
 when  then
 ...
 else  end
```

```
select editorial,
 case count(*)
  when 1 then 1
  else 'mas de 1' end as 'cantidad'
 from libros
 group by editorial;
```

JOINS

- Nos permiten encontrar datos entre tablas que esten relacionadas

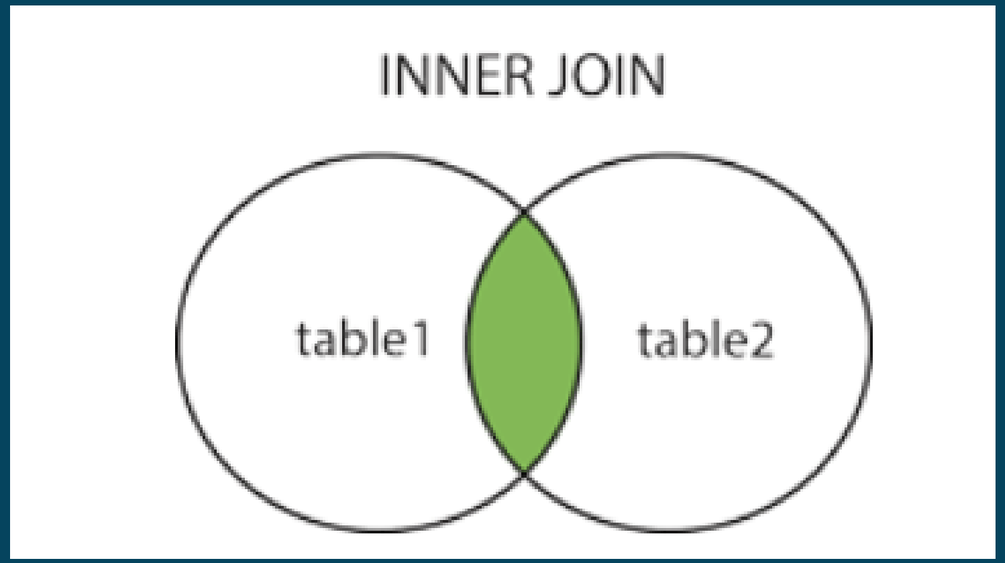
- Tipos de joins
- inner join
- left join
- right join
- full join
-

INNER JOIN

- Selecciona los registros que tienen valores en común

Sintaxis

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



INNER JOIN

- Selecciona los registros que tienen valores en común

Tabla orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Tabla customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

INNER JOIN

Consulta

```sql
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Resultado

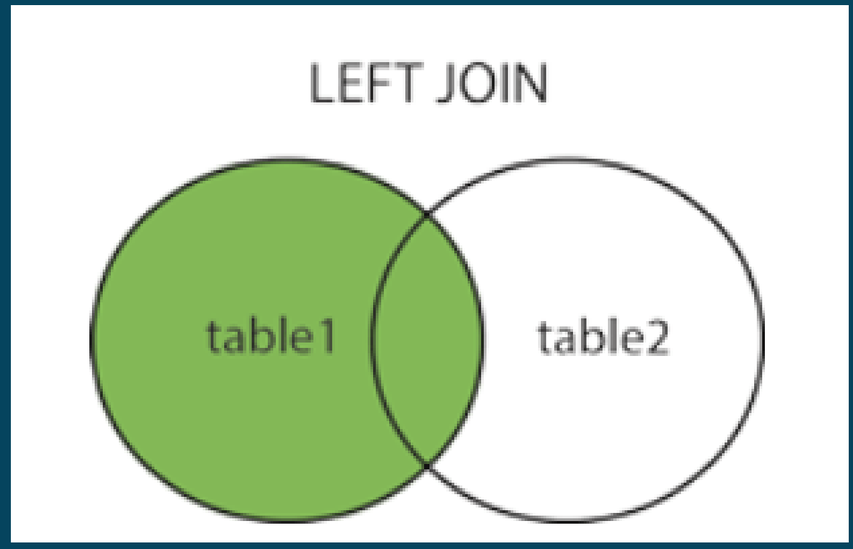
OrderID	CustomerID
10308	2

LEFT JOIN

Selecciona todos los registros de la primera tabla (izquierda gráficamente) y los registro que tengan valores en común con la tabla de la derecha

Sintaxis

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

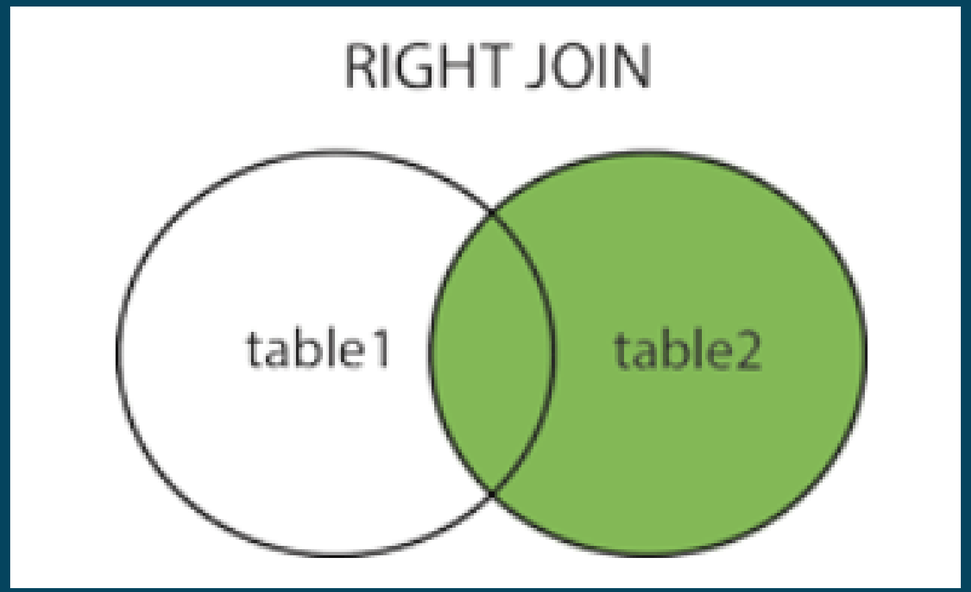


RIGHT JOIN

Selecciona todos los registros de la segunda tabla (derecha gráficamente) y los registro que tengan valores en común con la tabla de la izquierda

Sintaxis

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

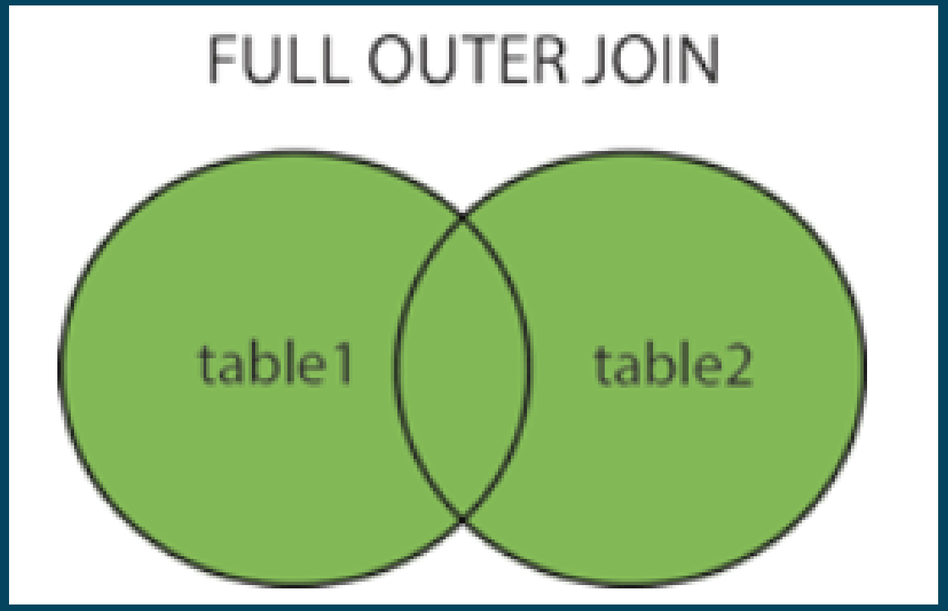


FULL JOIN o FULL OUTER JOIN

Selecciona todos los registros cuando encuentra algún registro con valores en común en las dos tablas

Sintaxis

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



FULL JOIN o FULL OUTER JOIN

Tabla orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Tabla customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

FULL JOIN o FULL OUTER JOIN

Resultado de la consulta

CustomerName	OrderID
Alfreds Futterkiste	*Null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	*Null*

Vistas

- Las vistas son tablas temporales que son construidas en base a los resultados obtenidos desde una consulta

- Una vista contiene filas y columnas para mostrar los datos obtenidos

Sintaxis

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Vistas

Actualizar vistas

- Nos permiten actualizar la información en cuanto en registros como en columnas

Sintaxis

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Ejemplo

```
CREATE VIEW
vista_futbolistas AS
SELECT futbolistas.id, nombre, apellidos FROM futbolistas
INNER JOIN tarjetas_amarillas
ON futbolistas.id = tarjetas_amarillas.id_futbolista;
```

Vistas

Eliminar vistas

Sintaxis

```
DROP VIEW view_name;
```