SOC Servei d'Ocupació de Catalunya

nascor formación

MINISTERIO DE TRABAJO E INMIGRACIÓN

SERVICIO PÚBLICO DE EMPLEO ESTATAL

Certificado de profesionalidad IFCD0210

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

¿Qué es una BBDD?

Una **base de datos relacional** es una recopilación de elementos de **datos** con relaciones predefinidas entre ellos. Estos elementos se organizan como un conjunto de tablas con columnas y filas. Las tablas se utilizan **para** guardar información sobre los objetos que se van a representar en la **base de datos**.

Características

Una **base de datos** se compone de varias **tablas**, relacionadas entre sí.

No pueden existir dos tablas con el mismo nombre ni registro.

Cada tabla es a su vez un conjunto de **atributos** (columnas) y **registros** (filas).

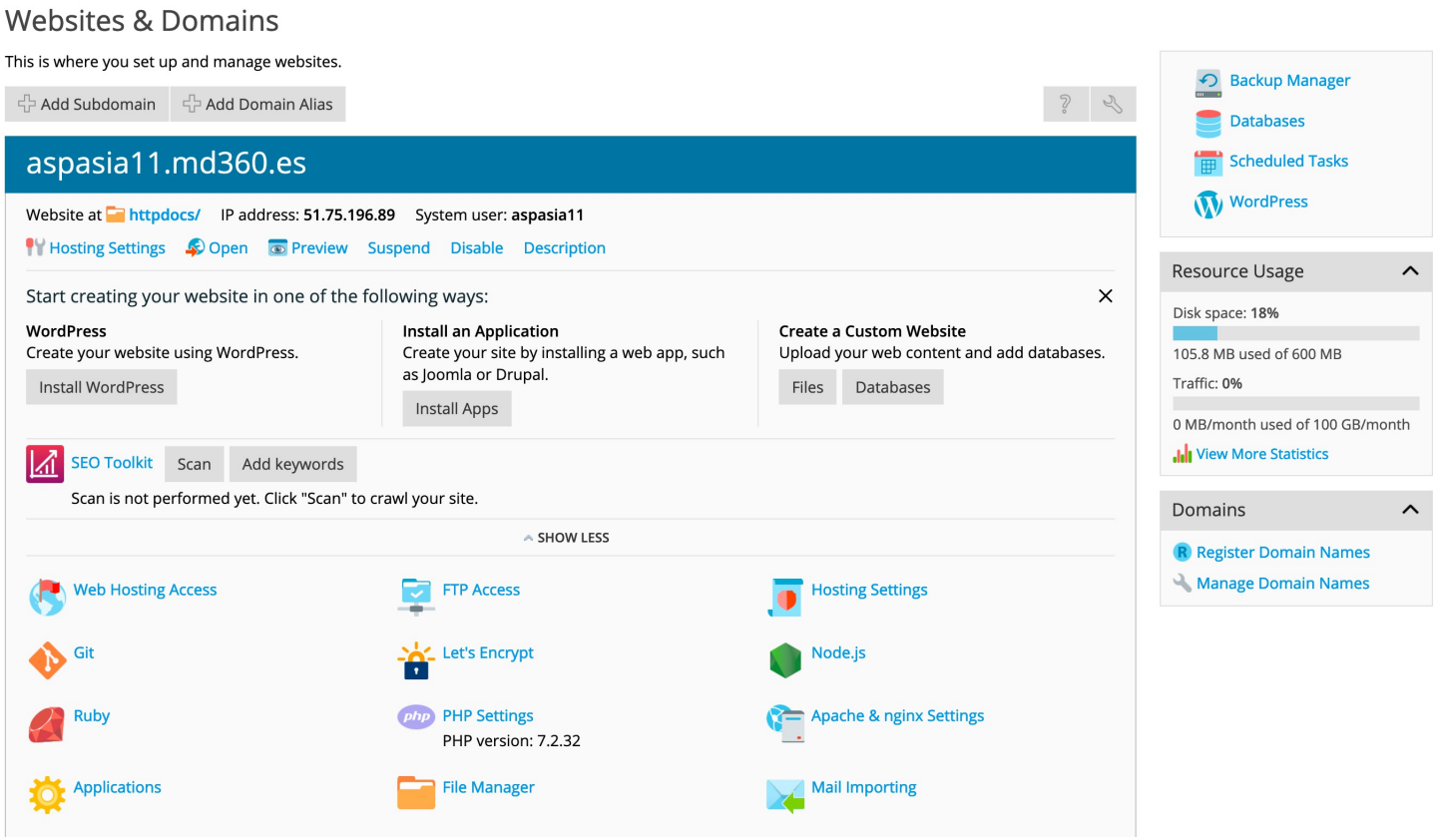
Las tablas tienen atributos claves. Estos atributos son únicos para cada registro. Por ejemplo, un identificador para cada fila.

Una tabla

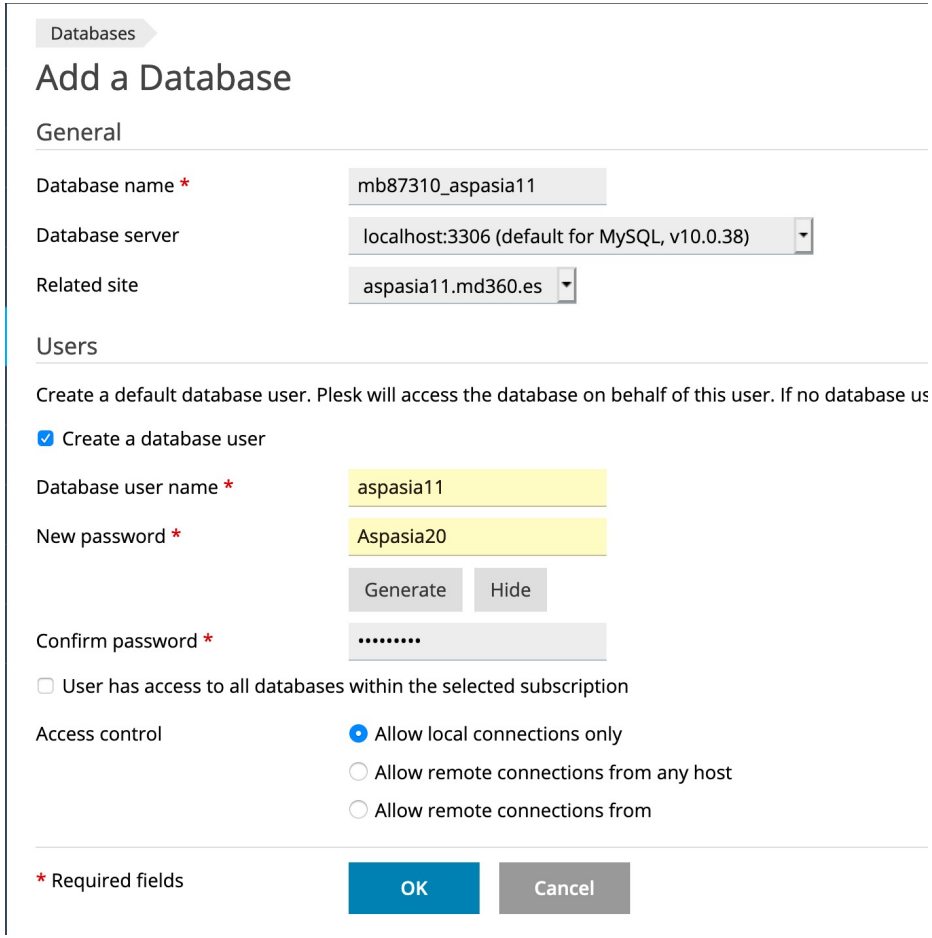
USUARIOS

username
password
valor_cookie
fecha_conexión

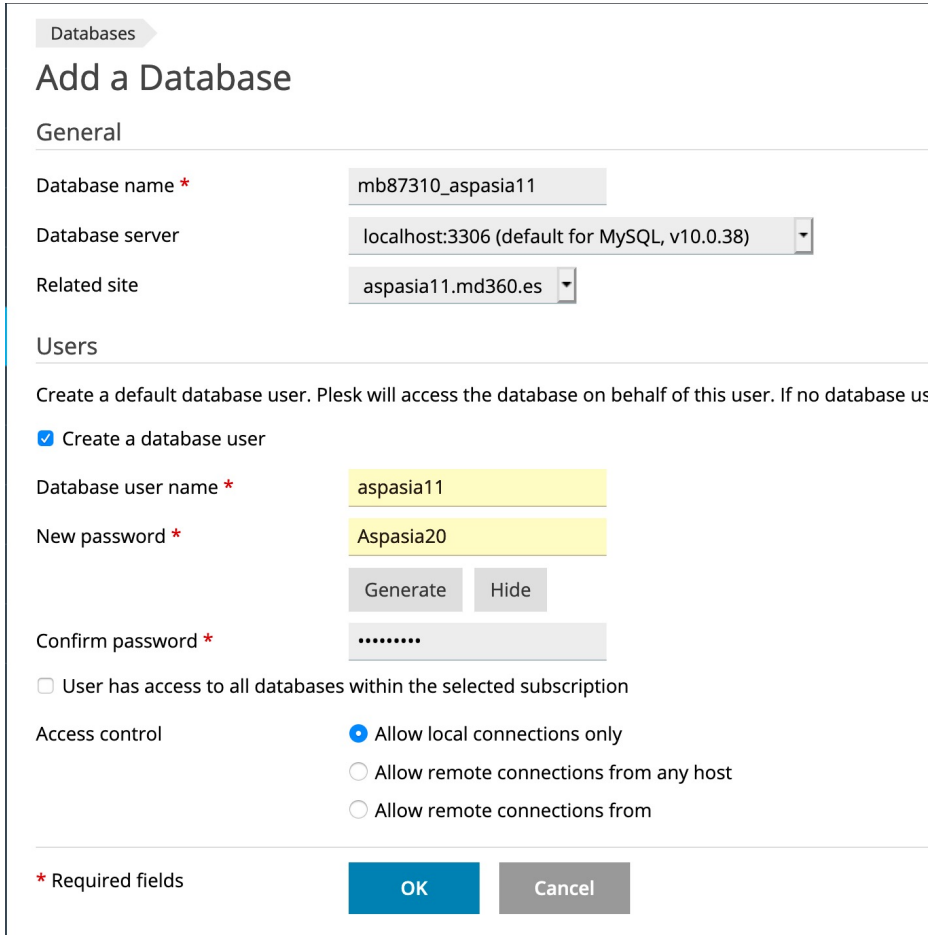
¿La creamos?


Websites & Domains
This is where you set up and manage websites.
Add Subdomain
Add Domain Alias
aspasia11.md360.es
Website at httpdocs/ IP address: 51.75.196.89 System user: aspasia11
Hosting Settings
Open
Preview
Suspend
Disable
Description
Start creating your website in one of the following ways:
WordPress
Create your website using WordPress.
Install WordPress
Install an Application
Create your site by installing a web app, such as Joomla or Drupal.
Install Apps
Create a Custom Website
Upload your web content and add databases.
Files
Databases
SEO Toolkit
Scan
Add keywords
Scan is not performed yet. Click "Scan" to crawl your site.
SHOW LESS
Web Hosting Access
FTP Access
Hosting Settings
Git
Let's Encrypt
Node.js
Ruby
PHP Settings
PHP version: 7.2.32
Apache & nginx Settings
Applications
File Manager
Mail Importing
Backup Manager
Databases
Scheduled Tasks
WordPress
Resource Usage
Disk space: 18%
105.8 MB used of 600 MB
Traffic: 0%
0 MB/month used of 100 GB/month
View More Statistics
Domains
Register Domain Names
Manage Domain Names

Creamos la BBDD



Creamos la BBDD



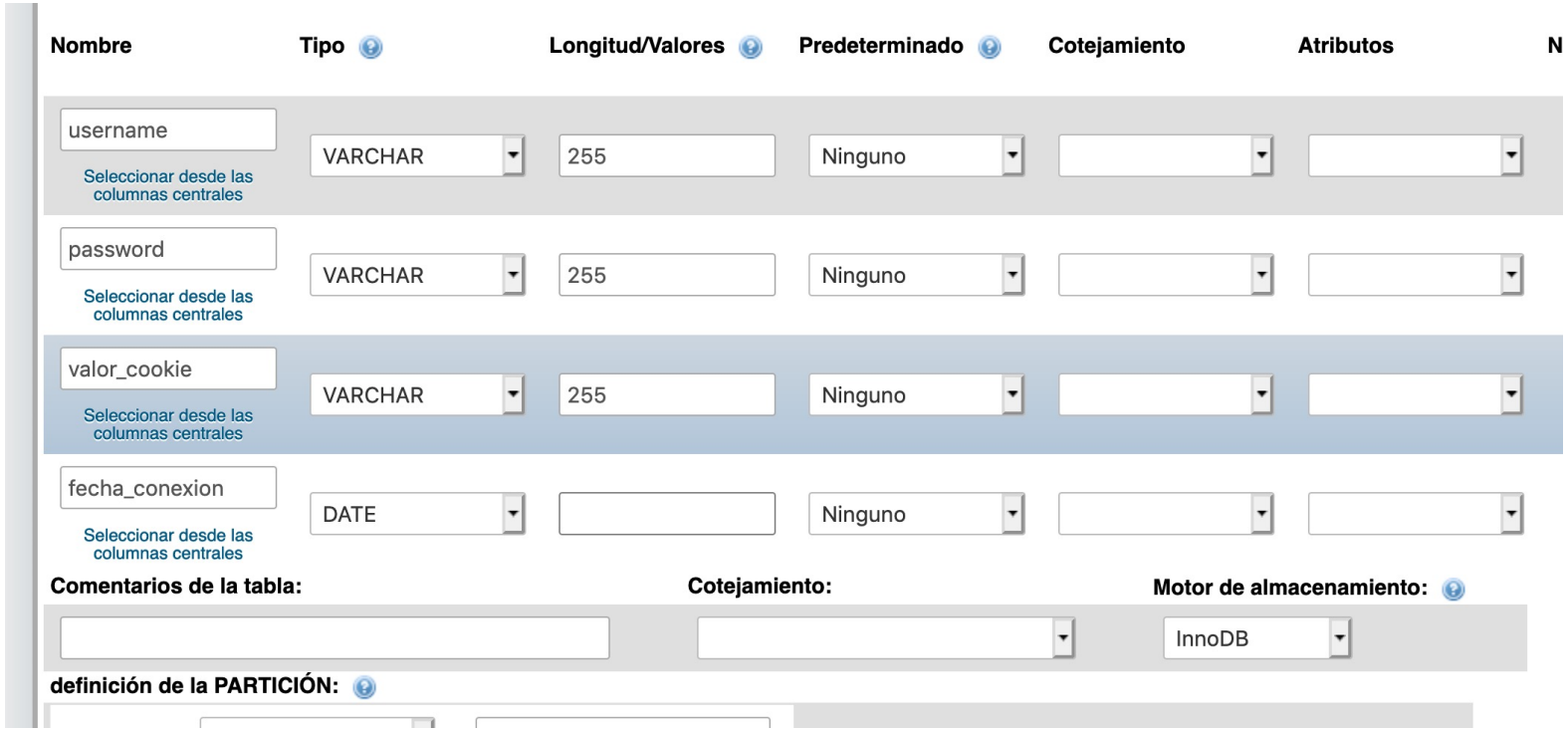
Tipos de datos

INT: número enteros
FLOAT: números decimales
VARCHAR: textos hasta un máximo de 65.535 caracteres, se indexan fácilmente.
TEXT: textos largos, no se indexan fácilmente.
DATE: Fechas

https://disenowebakus.net/tipos-de-datos-mysql.php

Creamos la tabla

Creamos la tabla



Ejemplo de acceso PHP

Accedemos a una tabla para poder realizar la validación de un usuario

https://nascor17.md360.es/modulo2/ejemplos/sesiones/ejemplo4/index.html

Actividad BBDD1

Completar el ejemplo4 modificando el identificación.php para que consulte en BBDD la sesión.

- Realizar la consulta en BBDD del id de sesión y el tiempo:

“SELECT username,fecha_conexion FROM usuarios WHERE valor_cookie=‘”.session_id().”’ “

Actividad BBDD1

Comprobar que la sesión existe y el tiempo de conexión con BBDD

```
SELECT username
FROM usuarios
WHERE valor_cookie='khlsmrshs5vv8j7rhtfmfkvgdm'
    AND date_add(fecha_conexion, INTERVAL 1 HOUR) > now()
```

https://www.w3schools.com/sql/func_mysql_date_add.asp

1 tablas

USUARIOS

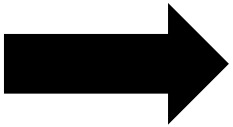
username
password
valor_cookie
fecha_conexión
Nombre
Apellido1
Apellido2
fecha_nacimiento
pais
moneda_país
poblacion_país

1 tablas

username	password	valor_cookie	fecha_conexion	nombre	apellido1	apellido2	fecha_nacimiento	pais	moneda_pais	poblacion_pais
borja.mulleras	Aspasia20			Borja	Mulleras	Vinzia	19/1/72	España	Euro	45M
alex.lopez	Aspasia10			Alex	Lopez	Lopez	30/10/78	España	Euro	45M
oscar.vilalta	Aspasia40			Oscar	Vilalta	Fernandex	20/10/79	España	Euro	45M
enric.colet	Aspasia30			Enric	Colet	Petit		España	Euro	45M

2 tablas

USUARIOS
username
password
valor_cookie
fecha_conexión
Nombre
Apellido1
Apellido2
fecha_nacimiento
pais
moneda_país
poblacion_país



USUARIOS
username
password
valor_cookie
fecha_conexión
Nombre
Apellido1
Apellido2
fecha_nacimiento
id_pais

PAISES
id_pais
pais
moneda_país
poblacion_país

2 tablas

Usuarios

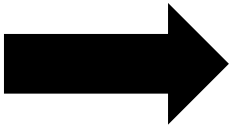
username	password	valor_cookie	fecha_conexion	nombre	apellido1	apellido2	fecha_nacimiento	id_pais
borja.mulleras	Aspasia20			Borja	Mulleras	Vinzia	19/1/72	1
alex.lopez	Aspasia10			Alex	Lopez	Lopez	30/10/78	1
oscar.vilalta	Aspasia40			Oscar	Vilalta	Fernandex	20/10/79	1
enric.colet	Aspasia30			Enric	Colet	Petit		2

Paises

id_pais	pais	moneda	poblacion
1	España	Euro	45M
2	Francia	Euro	67M

3 tablas

USUARIOS
username
password
valor_cookie
fecha_conexión
Nombre
Apellido1
Apellido2
fecha_nacimiento
pais
moneda_país
poblacion_país



USUARIOS
username
password
valor_cookie
fecha_conexión
id_persona

PERSONAS
id_persona
Nombre
Apellido1
Apellido2
fecha_nacimiento
id_pais

PAISES
id_pais
pais
moneda_país
poblacion_país

3 tablas

Usuarios

username	password	valor_cookie	fecha_conexion	Id_persona
borja.mulleras	Aspasia20			1
alex.lopez	Aspasia10			2
oscar.vilalta	Aspasia40			3
enric.colet	Aspasia30			4

Paises

id_pais	pais	moneda	poblacion
1	España	Euro	45M
2	Francia	Euro	67M

Personas

id_persona	nombre	apellido1	apellido2	fecha_nacimiento	id_pais
1	Borja	Mulleras	Vinzia	19/1/72	1
2	Alex	Lopez	Lopez	30/10/78	1
3	Oscar	Vilalta	Fernandex	20/10/79	1
4	Enric	Colet	Petit		2

Entidad

Representa una “cosa”, "objeto" o "concepto" del mundo real con existencia independiente, es decir, se diferencia únicamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad.

Algunos ejemplos:
•Una persona: se diferencia de cualquier otra persona, incluso siendo gemelos.
•Un automóvil: aunque sean de la misma marca, el mismo modelo, etc, tendrán atributos diferentes, por ejemplo, el número de chasis.
•Una casa: aunque sea exactamente igual a otra, aún se diferenciará en su dirección.

Una entidad puede ser un objeto con existencia física como: una persona, un animal, una casa, etc. (entidad concreta); o un objeto con existencia conceptual como: un puesto de trabajo, una asignatura de clases, un nombre, etc. (entidad abstracta).
Una entidad está descrita y se representa por sus características o atributos. Por ejemplo, la entidad *Persona* tiene como características: Nombre, Apellido, Género, Estatura, Peso, Fecha de nacimiento.

3 entidades con sus atributos

USUARIO
username
password
valor_cookie
fecha_conexión

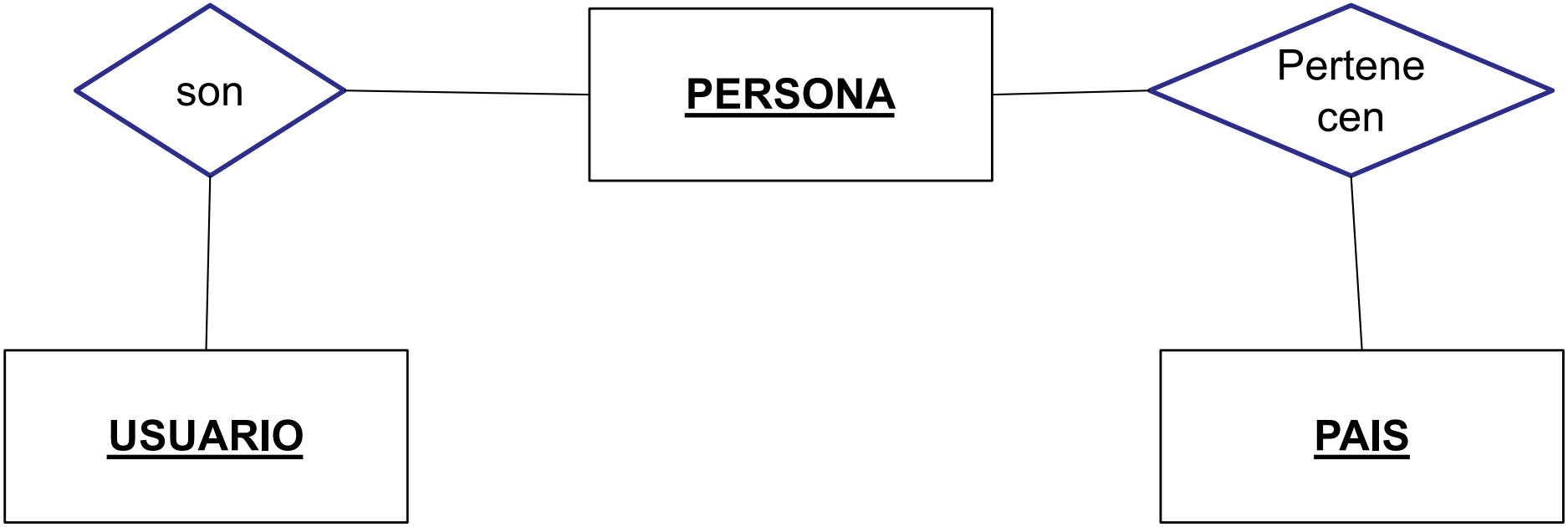
PERSONA
Nombre
Apellido1
Apellido2
fecha_nacimiento

PAIS
pais
moneda_país
poblacion_país

Relaciones

Una relación es una asociación entre una o más entidades con ciertas restricciones que vienen determinadas por las entidades participantes en dicha relación.

Las relaciones se representan gráficamente con un rombo.



Cardinalidad

Dado un conjunto de relaciones en el que participan dos o más conjuntos de entidades, la cardinalidad de la correspondencia indica el número de entidades con las que puede estar relacionada una entidad dada.

Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, las cardinalidades pueden ser:

Uno a Uno: (1:1) Un registro de una entidad A se relaciona con solo un registro en una entidad B. (ejemplo dos entidades, usuarios y personas, un usuario solo puede ser una persona y una persona solo puede tener un usuario).

Cardinalidad

Varios a Uno: (N:1) Una entidad en A se relaciona exclusivamente con una entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A (ejemplo personas-paises).

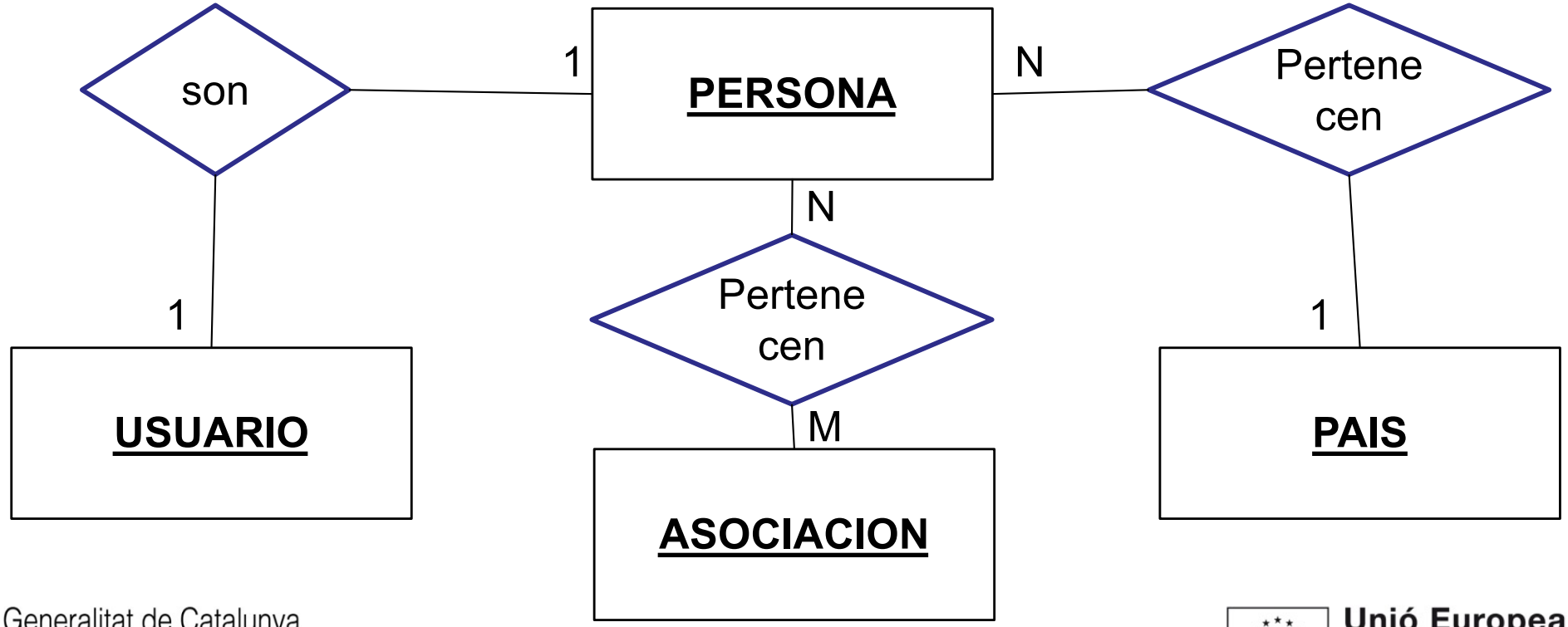
Uno a Varios: (1:N) Un registro en una entidad en A se relaciona con cero o muchos registros en una entidad B. Pero los registros de B solamente se relacionan con un registro en A. (ejemplo: dos entidades, vendedor y ventas, un vendedor puede tener muchas ventas pero una venta solo puede tener un vendedor).

Varios a Varios: (N:M) Una entidad en A se puede relacionar con 0 o con muchas entidades en B y viceversa (ejemplo asociaciones-ciudadanos, donde muchos ciudadanos pueden pertenecer a una misma asociación, y cada ciudadano puede pertenecer a muchas asociaciones distintas).

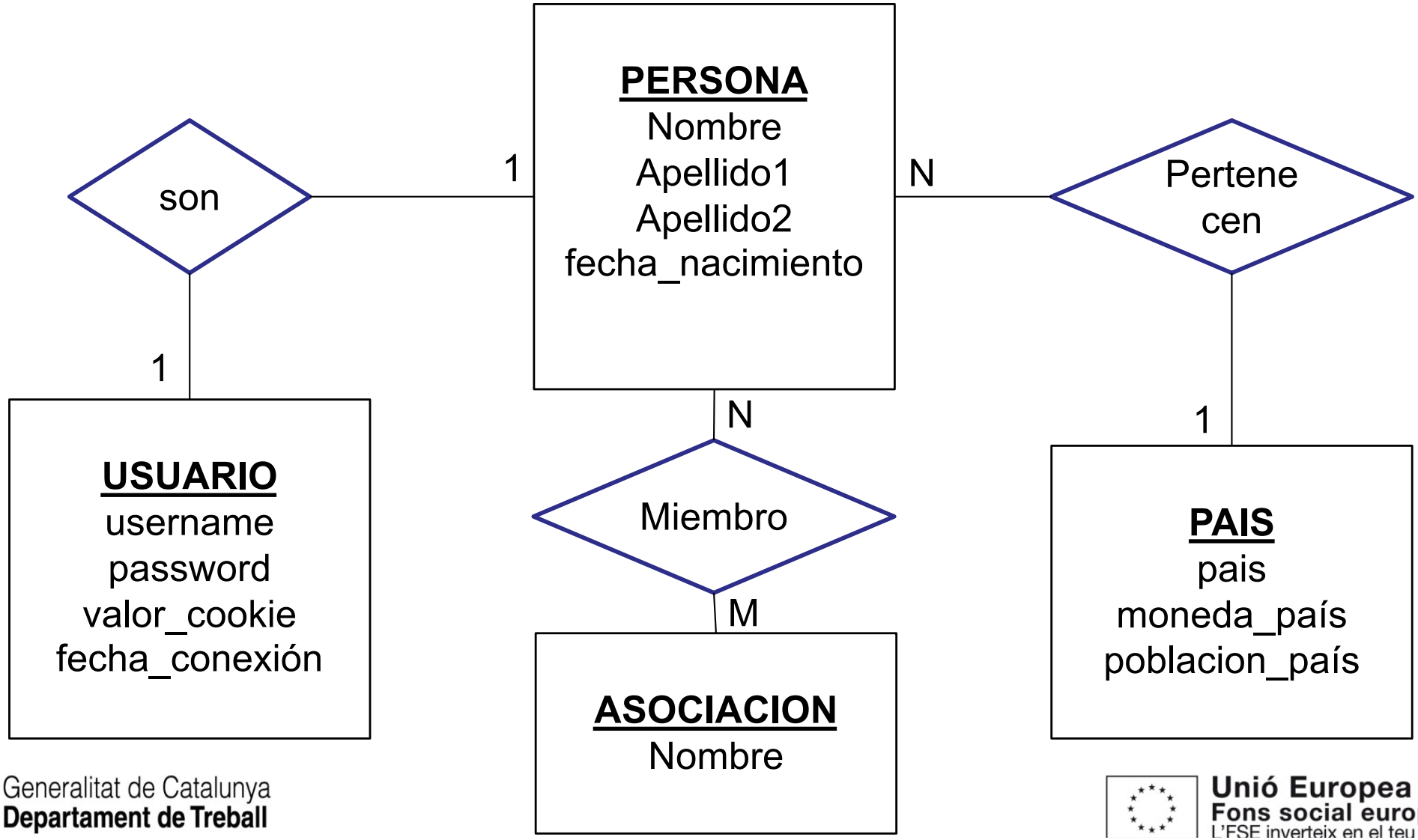
Relaciones

Una relación es una asociación entre una o más entidades con ciertas restricciones que vienen determinadas por las entidades participantes en dicha relación.

Las relaciones se representan gráficamente con un rombo.



Modelo relacional



MODELO DE DATOS

Usuarios

username	password	valor_cookie	fecha_conexion	Id_persona
borja.mulleras	Password20			1
alex.lopez	Password10			2
oscar.vilalta	Password40			3
enric.colet	Password30			4

Paises

id_pais	pais	moneda	poblacion
1	España	Euro	45M
2	Francia	Euro	67M

Personas

id_persona	nombre	apellido1	apellido2	fecha_nacimiento	id_pais
1	Borja	Mulleras	Vinzia	19/1/72	1
2	Alex	Lopez	Lopez	30/10/78	1
3	Oscar	Vilalta	Fernandex	20/10/79	1
4	Enric	Colet	Petit		2

ACTIVIDAD BBDD2

Partiendo de Alumnos, Profesores y Aulas:

A) Realizar el modelo entidad relación entre las 3 entidades y definir sus atributos.

B) Traducir dicho modelo a tablas y crearlas en la BBDD. Añadir datos a dichas tablas.

ACTIVIDAD BBDD3

Definir las entidades para realizar un modelo entidad-relación para localizar los libros en una biblioteca familiar. La idea es que tenemos unos estantes identificados y cada libro se guarda en uno de ellos. Además, queremos poder consultar los libros de un determinado autor. Cada libro pertenece a una persona y también queremos saber los libros que pertenecen a una persona determinada.

A) Realizar el modelo entidad relación entre las 3 entidades y definir sus atributos.

B) Traducir dicho modelo a tablas y crearlas en la BBDD

ACTIVIDAD BBDD4

A partir de uno de los modelos creados en las actividades BBDD2 o BBDD3, crear una pequeña aplicación que muestre el resultado de una consulta realizada a la bbdd siguiendo el modelo MVC mostrados en los ejemplos de clase.

LA NORMALIZACIÓN

RefPed	FecPed	CodArt	DesArt	CantArt	PVPArt
P0001	16/02/2014	A0043	Bolígrafo azul fino	10	0,78
P0001	16/02/2014	A0078	Bolígrafo rojo normal	12	1,05
P0002	18/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0003	23/02/2014	A0075	Lápiz 2B	20	0,55
P0004	25/02/2014	A0012	Goma de borrar	15	0,15
P0004	25/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0004	25/02/2014	A0089	Sacapuntas	50	0,25

Redundancia de datos:

La fecha del pedido se repite en cada línea del pedido

Si un artículo se ha pedido varias veces se repite la descripción del artículo y código

LA NORMALIZACIÓN

RefPed	FecPed	CodArt	DesArt	CantArt	PVPArt
P0001	16/02/2014	A0043	Bolígrafo azul fino	10	0,78
P0001	16/02/2014	A0078	Bolígrafo rojo normal	12	1,05
P0002	18/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0003	23/02/2014	A0075	Lápiz 2B	20	0,55
P0004	25/02/2014	A0012	Goma de borrar	15	0,15
P0004	25/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0004	25/02/2014	A0089	Sacapuntas	50	0,25

Anomalías de inserción:

No puedo añadir un artículo que no haya sido comprado anteriormente

¿Cómo lo inserto?

LA NORMALIZACIÓN

RefPed	FecPed	CodArt	DesArt	CantArt	PVPArt
P0001	16/02/2014	A0043	Bolígrafo azul fino	10	0,78
P0001	16/02/2014	A0078	Bolígrafo rojo normal	12	1,05
P0002	18/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0003	23/02/2014	A0075	Lápiz 2B	20	0,55
P0004	25/02/2014	A0012	Goma de borrar	15	0,15
P0004	25/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0004	25/02/2014	A0089	Sacapuntas	50	0,25

Anomalías de modificación:

Modificar la descripción del artículo o la fecha del pedido

LA NORMALIZACIÓN

RefPed	FecPed	CodArt	DesArt	CantArt	PVPArt
P0001	16/02/2014	A0043	Bolígrafo azul fino	10	0,78
P0001	16/02/2014	A0078	Bolígrafo rojo normal	12	1,05
P0002	18/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0003	23/02/2014	A0075	Lápiz 2B	20	0,55
P0004	25/02/2014	A0012	Goma de borrar	15	0,15
P0004	25/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0004	25/02/2014	A0089	Sacapuntas	50	0,25

Anomalías de borrado:

Si quiero borrar un artículo, lo tengo que borrar de la tabla y desapareceria de los pedidos realizados anteriormente.

LA NORMALIZACIÓN

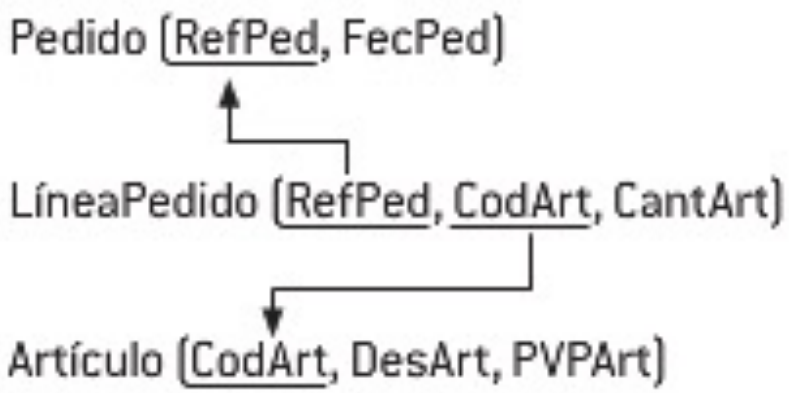
RefPed	FecPed	CodArt	DesArt	CantArt	PVPArt
P0001	16/02/2014	A0043	Bolígrafo azul fino	10	0,78
P0001	16/02/2014	A0078	Bolígrafo rojo normal	12	1,05
P0002	18/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0003	23/02/2014	A0075	Lápiz 2B	20	0,55
P0004	25/02/2014	A0012	Goma de borrar	15	0,15
P0004	25/02/2014	A0043	Bolígrafo azul fino	5	0,78
P0004	25/02/2014	A0089	Sacapuntas	50	0,25

¿Cuál es el problema?

Almacenamos artículos y pedidos en una misma tabla

LA NORMALIZACIÓN

El **proceso de normalización** es un método formal para organizar datos que nos ayuda a determinar qué es lo que está equivocado en un diseño y como corregirlo. Existen hasta 5 formas normales en función de las relaciones establecidas entre los datos.



LA NORMALIZACIÓN
Dependencia funcional

Decimos que un atributo Y de una relación «*depende funcionalmente*» de otro atributo X de la relación si a todo valor de X le corresponde siempre el mismo valor de Y.

Por ejemplo, si un atributo de la relación es el nombre de pila y otro es el NIF, podemos asegurar que a un valor concreto de NIF, corresponde siempre el mismo nombre de pila. Entonces, decimos que «el NIF determina el nombre» o que «nombre depende funcionalmente de NIF».

Fíjate que al revés no es cierto. Para un mismo nombre (por ejemplo Ana) pueden corresponder varios NIFs distintos.

Las dependencias funcionales se representan así:

X -> Y

NIF -> nombre

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

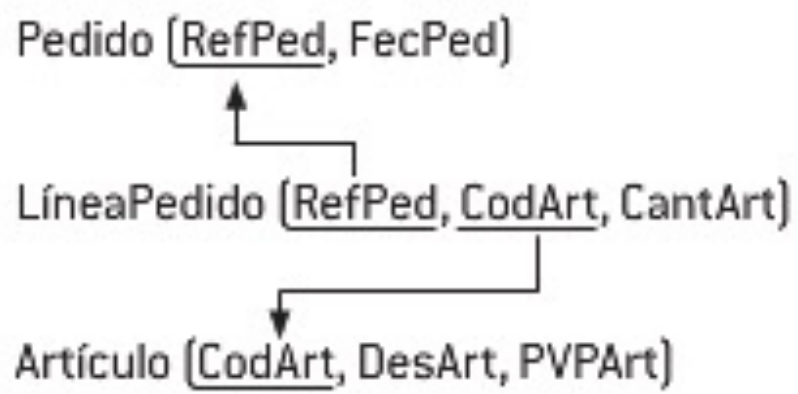
LA NORMALIZACIÓN
Dependencia funcional

Y naturalmente, no tiene porque ser un solo atributo:

Decimos que un *conjunto de atributos* Y de una relación «*depende funcionalmente*» de otro *conjunto de atributos* X de la relación si a todo valor de X le corresponde siempre el mismo valor de Y.

Por ejemplo, si tenemos una tabla con los atributos de la dirección postal de una persona (calle, piso, código postal, localidad…) y su nombre completo (nombre, apellido1, apellido2), podemos ver que el conjunto de atributos que forman la dirección, dependen funcionalmente del conjunto de atributos que forman el nombre.

LA NORMALIZACIÓN
Dependencia funcional



Dependencias funcionales:
Artículo: CodArt -> DesArt y CodArt -> PVPArt
LineaPedido: RefPed,CodArt -> CantArt
Pedido: RefPed -> FecPed

LA NORMALIZACIÓN
1FN

- **Todos los atributos son «atómicos»**. Por ejemplo, en el campo teléfono no tenemos varios teléfonos.

- La tabla contiene una **clave primaria única**. Por ejemplo el NIF para personas, la matrícula para vehículos o un simple id autoincremental. Si no tiene clave, no es 1FN.

- **La clave primaria no contiene atributos nulos**. No podemos tener filas para las que no haya clave (por ejemplo, personas sin NIF o vehículos sin matrícula).

- **No debe existir variación en el número de columnas**. Si algunas filas tienen 8 columnas y otras 3, pues no estamos en 1FN.

LA NORMALIZACIÓN
1FN

- **Los campos no clave deben identificarse por la clave**. Es decir, que los campos no clave dependen funcionalmente de la clave. Esto es prácticamente lo mismo que decir que existe clave primaria.

- **Debe Existir una independencia del orden tanto de las filas como de las columnas**, es decir, si los datos cambian de orden no deben cambiar sus significados. Por ejemplo, si en la columna 1 tenemos el primer apellido y en la columna 2 tenemos el segundo, pues no estamos en 1FN. Igualmente si en la tercera fila tenemos el tercer mejor expediente y en la quinta fila el quinto, no estamos en 1FN.

https://naps.com.mx/blog/ejemplos-de-primera-forma-normal-del-modelo-relacional/

LA NORMALIZACIÓN
2FN

•Una tabla está en 2FN si además de estar en 1FN cumple que **los atributos no clave depende de TODA la clave principal**.

•Por ejemplo, si tenemos una tabla con Personas, identificadas por su NIF y recogemos su empresa y dirección de trabajo, la clave sería NIF-Empresa. Pero nos encontraremos con que una misma persona puede trabajar en varias empresas. Y vemos que la dirección de trabajo no depende de TODA la clave primaria, sino solo de la empresa. Por lo tanto, no estamos en 2FN.

https://naps.com.mx/blog/segunda-forma-normal-en-el-modelo-relacional-2fn/

LA NORMALIZACIÓN
3FN

Una tabla está en 3FN si además de estar en 2FN **no existe ninguna dependencia transitiva entre los atributos que no son clave**.

Supongamos que tenemos una tabla de ganadores de torneos de tenis. En ella figura el nombre del torneo, el año, el nombre del ganador y su nacionalidad. La clave sería Torneo-Año. Pues esta tabla no está en 3FN porque el atributo nacionalidad, que no es de la clave, depende del nombre del ganador (también depende de la clave). Digamos que nacionalidad aporta información sobre el ganador, pero no sobre la clave. Es una dependencia transitiva porque nacionalidad depende de ganador que a su vez depende de Torneo-Año.

https://naps.com.mx/blog/tercera-forma-normal-en-el-modelo-relacional-3fn/

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

LA NORMALIZACIÓN
Ejemplos

https://www.youtube.com/watch?v=bO18omSzeR4

https://www.youtube.com/watch?v=ig_CUc_5wuM

LA NORMALIZACIÓN
Ejercicio BBDD5

A partir de esta tabla determinar la 3FN

CodLibro	Titulo	Autor	Editorial	NombreLector	FechaDev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez Gómez, Juan	15/04/2005
1004	Visual Basic 5	E. Petroustsos	Anaya	Ríos Terán, Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca, René	16/04/2005
1006	Oracle University	Nancy Greenberg y Priya Nathan	Oracle Corp.	García Roque, Luis	20/04/2005
1007	Clipper 5.01	Ramalho	McGraw Hill	Pérez Gómez, Juan	18/04/2005

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Ejercicio BBDD6

A partir de los ejemplos explicados en clase de como insertar datos en una base de datos con un programa php siguiendo el patrón MVC, realizar inserciones en el proyecto del ejercicio BBDD4

SOC Servei d'Ocupació de Catalunya
nascor formación
MINISTERIO DE TRABAJO E INMIGRACIÓN
SERVICIO PÚBLICO DE EMPLEO ESTATAL
Generalitat de Catalunya
Departament de Treball
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Clave primaria

En el diseño de bases de datos relacionales, se llama **clave primaria** o **clave principal** a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria.

Clave primaria

```
ALTER TABLE usuarios
ADD PRIMARY KEY(username);
```

Definimos la clave primaria Username para la tabla usuarios

Clave única o candidata

En el diseño de bases de datos relacionales, se llama **clave única** o **UNIQUE** a un campo o a una combinación de campos que no pueda tener dos datos iguales. Es decir en cada registro, el campo marcado con UNIQUE debe tener un dato diferente. Esto lo convierte en un identificador del registro, ya que no puede haber dos registros que contengan el mismo dato en esa columna.

Clave única

```
ALTER TABLE miagenda
    ADD direccion VARCHAR(255),
    ADD UNIQUE (direccion),
    ADD UNIQUE (telefono2);
```

Definimos como claves únicas dirección y telefono2

Clave primaria vs única

¿Qué pasa con la tabla Alu_asig?

id	id_alumno	id_asignatura
1	1	1
16	1	2
15	2	1
3	2	2
17	3	1
7	3	2

Clave primaria vs única

¿Qué pasa con la tabla Alu_asig?

Id es la clave primaria y única

La combinación de id_alumno y id_asignatura tiene que ser una clave única.

Clave primaria vs única

¿Qué pasa con la tabla Alu_asig?

	#	Nom	Tipus	Col·lació	Atributs	Nul	Per defecte	Comentaris	Extra	Acció
☐	1	id	int(11)			No	*Cap*		AUTO_INCREMENT	Canvia Elimina Més
☐	2	id_alumno	int(11)			No	*Cap*			Canvia Elimina Més
☐	3	id_asignatura	int(11)			No	*Cap*			Canvia Elimina Més

Marca-ho tot *Amb els seleccionats:* Navega Canvia Elimina Primària Única Índex Text compl

Imprimeix Proposa una estructura de taula Mou les columnes Normalitza

Afegeix 1 columna(es) després de id_asignatura **Executa**

Índexs

Acció	Nom de clau	Tipus	Única	Empaquetat	Columna	Cardinalitat	Col·lació	Nul	Comentari
Edita Elimina	PRIMARY	BTREE	Sí	No	id	30	A	No	
Edita Elimina	id_alumno	BTREE	Sí	No	id_alumno	30	A	No	
					id_asignatura	30	A	No	

Clave primaria vs única

¿Qué pasa con la tabla Alu_asig?

```
ALTER TABLE `Alu_asig`
ADD UNIQUE(id_alumno,id_asignatura)
```

Clave primaria vs única

¿Podemos prescindir del ID y dejar como clave única y primaria la combinación de id_alumno y id_asignatura?

Clave foranea

Una **clave foránea** es un campo común y corriente que tiene la particularidad de corresponderse con la clave primaria o una candidata de otra tabla, una columna en una tabla que contiene los mismos valores.

Una **clave externa** denota la relación entre las dos tablas. Se puede crear una **clave externa** en una columna o un grupo de columnas en una tabla y usarla para hacer referencia a una columna o grupo de columnas de otra tabla. Las columnas a las que se hace referencia deben ser una clave primaria o única y no pueden contener valores nulos.

Clave foranea

Estas claves están muy estrechamente ligadas al concepto de **integridad referencial**, debido a que si una **clave foránea** contiene un valor, ese valor se refiere a un registro existente en la tabla relacionada. Las columnas referenciadas ya deben contener los valores clave que se insertan en una nueva fila para la tabla de **clave externa**. Si no están presentes, no se podrá insertar la fila. Además, todos los valores clave en la tabla de **claves externas** deben borrarse antes de eliminar los valores clave en la tabla a la que se hace referencia.

Clave foranea

A continuación un ejemplo en SQL, para crear una **clave foránea** para la tabla *Salarios*, haciendo referencia a la tabla *Empleados* con una clave primaria compuesta para sus columnas *ID* y *Contacto*:

```
ALTER TABLE Salarios
FOREIGN KEY FK1 (ID, Contacto)
REFERENCES Empleados
```

Índices ordinarios

Los índices ordinarios permiten optimizar las consultas que se realizan por el atributo objeto del índice.

Permite valores repetidos

```
ALTER TABLE usuarios
ADD INDEX idx_apellidos (apellidos);
```

Índices fulltext

Los índices de texto completo son del tipo FULLTEXT y pueden contener uno o más campos del tipo CHAR, VARCHAR y TEXT. Un índice de texto completo está diseñado para facilitar y optimizar la búsqueda de palabras clave en tablas que tienen grandes cantidades de información en campos de texto.

```
CREATE FULLTEXT INDEX idx_nombre
ON usuarios(nombre);
```

Ejercicio BBDD7

Crear un modelo entidad-relación del proyecto global pensado para el curso.

Los modelos se veran y discutirán en clase.

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

ALL es el valor predeterminado, especifica que el conjunto de resultados puede incluir filas duplicadas. Por regla general nunca se utiliza.

DISTINCT especifica que el conjunto de resultados sólo puede incluir filas únicas. Es decir, si al realizar una consulta hay registros exactamente iguales que aparecen más de una vez, éstos se eliminan. Muy útil en muchas ocasiones.

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

> (Mayor)
>= (Mayor o igual)
< (Menor)
<= (Menor o igual)
= (Igual)
<> o **!=** (Distinto)

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

IS [NOT] NULL (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)

Se dice que una columna de una fila es NULL si está completamente vacía. Hay que tener en cuenta que si se ha introducido cualquier dato, incluso en un campo alfanumérico si se introduce una cadena en blanco o un cero en un campo numérico, deja de ser NULL.

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones

ORDER BY ListaColumnas [ ASC / DESC ]
```

LIKE: para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: "%" y "_". Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda. Por ejemplo:

El nombre empieza por A: Nombre LIKE 'A%'
El nombre acaba por A: Nombre LIKE '%A'
El nombre contiene la letra A: Nombre LIKE '%A%'
El nombre empieza por A y después contiene un solo carácter cualquiera: Nombre LIKE 'A_'
El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: Nombre LIKE 'A_E%'

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

BETWEEN: para un intervalo de valores. Por ejemplo:

Clientes entre el 30 y el 100: `CodCliente BETWEEN 30 AND 100`

Clientes nacidos entre 1970 y 1979:
`FechaNac BETWEEN '19700101' AND '19791231'`

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

IN( ): para especificar una relación de valores concretos. Por ejemplo:

Ventas de los Clientes 10, 15, 30 y 75: CodCliente IN (10, 15, 30, 75)

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

Es posible combinar varias condiciones simples de los operadores anteriores utilizando los operadores lógicos **OR**, **AND** y **NOT**, así como el uso de paréntesis para controlar la prioridad de los operadores (como en matemáticas). Por ejemplo:

… (Cliente = 100 AND Provincia = 30) OR Ventas > 1000 …

que sería para los clientes de las provincias 100 y 30 o cualquier cliente cuyas ventas superen 1000.

SELECT

```
SELECT [ ALL / DISTINCT ] [ * ] /
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

ASC es el valor predeterminado, especifica que la columna indicada en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica DESC se ordenará de forma descendente (de mayor a menor).

Por ejemplo, para ordenar los resultados de forma ascendente por ciudad, y los que sean de la misma ciudad de forma descendente por nombre, utilizaríamos esta cláusula de ordenación:

```
… ORDER BY Ciudad, Nombre DESC …
```

SELECT

Mostrar todos los datos de los Clientes de nuestra empresa:

```
SELECT * FROM Customers
```

Mostrar apellido, ciudad y región (LastName, city, region) de los empleados de USA (nótese el uso de AS para darle el nombre en español a los campos devueltos):

```
SELECT E.LastName AS Apellido, City AS Ciudad, Region
FROM Employees AS E
WHERE Country = 'USA'
```

- Mostrar los clientes que no sabemos a qué región pertenecen (o sea, que no tienen asociada ninguna región):

```
SELECT * FROM Customers WHERE Region IS NULL
```

SELECT

- Mostrar las distintas regiones de las que tenemos algún cliente, accediendo sólo a la tabla de clientes:

```
SELECT DISTINCT Region FROM Customers WHERE Region IS NOT NULL
```

- Mostrar los clientes que pertenecen a las regiones CA, MT o WA, ordenados por región ascendentemente y por nombre descendentemente.

```
SELECT * FROM Customers
WHERE Region IN('CA', 'MT', 'WA')
ORDER BY Region, CompanyName DESC
```

SELECT

Mostrar los clientes cuyo nombre empieza por la letra “W”:

```
SELECT * FROM Customers WHERE CompanyName LIKE ‘W%’
```

Mostrar los empleados cuyo código está entre el 2 y el 9:

```
SELECT * FROM Employees WHERE EmployeeID BETWEEN 2 AND 9
```

Mostrar los clientes cuya dirección contenga “ki”:

```
SELECT * FROM Customers WHERE Address LIKE ‘%ki%’
```

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT Asignaturas.Nombre,Alumnos.Apellido1,Alumnos.Apellido2,Alumnos.Nombre
FROM Asignaturas, Alu_asig, Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
  ORDER BY
Asignaturas.Nombre,Alumnos.Apellido1,Alumnos.Apellido2,Alumnos.Nombre
```

SELECT

¿Cuantos alumnos hay matrículados por curso?

Nombre ▲ 4	Apellido1	Apellido2	Nombre
Desarrollo de aplicaciones Web	Alonso	de Andrés	Pere
Desarrollo de aplicaciones Web	Alvarez	Casas	Jose
Desarrollo de aplicaciones Web	Alvarez	Fernandez	Bruno
Desarrollo de aplicaciones Web	Fernandez	*NULL*	Virtudes
Desarrollo de aplicaciones Web	Fernández	García	Mariano
Desarrollo de aplicaciones Web	Fernandez	Gomez	Borja
Desarrollo de aplicaciones Web	Garcia	Calabazas	Alejandra
Desarrollo de aplicaciones Web	GARCIA	Casas	Pepe
Desarrollo de aplicaciones Web	Garcia	Gomez	Rafael
Desarrollo de aplicaciones Web	García	Lorca	Federico
Desarrollo de aplicaciones Web	Lopez	Fernandez	Jorge
Desarrollo de aplicaciones Web	Martínez	Machín	Saúl
Desarrollo de aplicaciones Web	Mata	Enrique	Fatima
Desarrollo de aplicaciones Web	Muller	Barbat	Alberto
Desarrollo de aplicaciones Web	Paco	Paco	Paco
Publicación de Páginas Web	Fernandez	*NULL*	Virtudes
Publicación de Páginas Web	Fernandez	Gomez	Borja
Publicación de Páginas Web	Garcia	Gomez	Rafael
Publicación de Páginas Web	Lopez	Fernandez	Jorge
Publicación de Páginas Web	Lopez	Gomez	Rafael
Publicación de Páginas Web	Mata	Enrique	Fatima
Publicación de Páginas Web	Mullerat	Barbado	Albert
Publicación de Páginas Web	Palacio	Rodriguez	Adri Benito

15

8

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT Asignaturas.Nombre,Alumnos.Apellido1,Alumnos.Apellido2,Alumnos.Nombre
FROM Asignaturas,Alu_asig,Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
  ORDER BY
Asignaturas.Nombre,Alumnos.Apellido1,Alumnos.Apellido2,Alumnos.Nombre
```

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT COUNT(*)
FROM Asignaturas, Alu_asig, Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
```

Eliminamos la consulta y el ORDER BY

Resultado: 23

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT COUNT(*)
FROM Asignaturas,Alu_asig,Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
  AND Asignaturas.id=1
```

Resultado: 15

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT COUNT(*)
FROM Asignaturas,Alu_asig,Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
  AND Asignaturas.id=2
```

Resultado: 8

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT Asignaturas.nombre,COUNT(Alu_asig.id_asignatura)
FROM Asignaturas,Alu_asig,Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
GROUP BY Asignaturas.nombre
```

nombre	COUNT(Alu_asig.id_asignatura)
Desarrollo de aplicaciones Web	15
Publicación de Páginas Web	8

SELECT

¿Cuantos alumnos hay matrículados por curso?

```
SELECT Asignaturas.nombre,COUNT(Alu_asig.id_asignatura)
FROM Asignaturas,Alu_asig,Alumnos
WHERE Asignaturas.id=Alu_asig.id_asignatura
  AND Alumnos.id=Alu_asig.id_alumno
GROUP BY Asignaturas.nombre
```

nombre	COUNT(Alu_asig.id_asignatura)
Desarrollo de aplicaciones Web	26
Publicación de Páginas Web	8

ERROR DE INTEGRIDAD DE DATOS

PROBLEMA INTEGRIDAD DE DATOS

SELECT * FROM ALU_ASIG

	id	id_alumno	id_asignatura
Editar Copiar Borrar	1	1	1
Editar Copiar Borrar	16	1	2
Editar Copiar Borrar	15	2	1
Editar Copiar Borrar	3	2	2
Editar Copiar Borrar	17	3	1
Editar Copiar Borrar	7	3	2
Editar Copiar Borrar	30	4	1
Editar Copiar Borrar	40	7	1
Editar Copiar Borrar	12	8	2
Editar Copiar Borrar	19	10	1
Editar Copiar Borrar	38	10	2
Editar Copiar Borrar	11	12	1
Editar Copiar Borrar	21	12	2
Editar Copiar Borrar	5	13	1
Editar Copiar Borrar	6	14	2
Editar Copiar Borrar	14	15	1
Editar Copiar Borrar	13	16	2
Editar Copiar Borrar	23	17	1

SELECT * FROM Alumnos WHERE Alumnos.id=13

id	Nombre	Apellido1	Apellido2	Fecha_nac

Lo mismo pasa con los id_alumno 18, 19, 23, 24, 25, 26, 27, 34, 35, 36

PROBLEMA INTEGRIDAD DE DATOS

¿Por qué?

PROBLEMA INTEGRIDAD DE DATOS

Falta una clave foranea entre id_alumno de Alu_Asig y la clave primaria id de la tabla Alumnos

PROBLEMA INTEGRIDAD DE DATOS

¿Cómo podemos saber qué registros son los que estan mal para arreglarlo?

PROBLEMA INTEGRIDAD DE DATOS

¿Cómo podemos saber qué registros son los que estan mal para arreglarlo?

```
SELECT Alu_asig.id_alumno
FROM Alu_asig
WHERE id_alumno NOT IN (SELECT Alumnos.id
                        FROM Alumnos)
```

id_alumno
13
18
19
23
24
25
26
27
34
35
36

PROBLEMA INTEGRIDAD DE DATOS

¿Cómo borrarlos?

```
DELETE FROM Alu_asig
WHERE Alu_asig.id_alumno in (13,18,19,23,24,25,26,27,34,35,36)
```

O

```
DELETE FROM Alu_asig
WHERE Alu_asig.id_alumno in (SELECT Alu_asig.id_alumno
                             FROM Alu_asig
                             WHERE id_alumno NOT IN (SELECT Alumnos.id
                                                     FROM Alumnos)
                             )
```

id_alumno
13
18
19
23
24
25
26
27
34
35
36

Ejercicio BBDD8

A partir del modelo entidad-relación del ejercicio anterior, crear las tablas con sus relaciones, claves primarias, claves foraneas, etc..

Presentar una exportación del modelo de datos correspondiente.

Inner Joins

Es una unión interna que utiliza cualquiera de las consultas equivalentes dando la intersección central de las dos o tres tablas, es decir, las dos filas que tienen en común entre si, veamos un ejemplo.

```
SELECT *
FROM Tabla1 a
INNER JOIN Tabla2 b ON a.id = b.id
LIMIT 0 , 30
```

+ Opciones

id	letra	id	valores
3	Tres	3	Tres
4	Cuatro	4	Cuatro

Inner Joins

Si observamos podemos utilizar cualquiera de las dos consultas y el resultado sera el mismo.

```
SELECT tabla1.*,tabla2.*
    FROM tabla1,tabla2
where tabla1.id = tabla2.id;
```

```
SELECT *
FROM tabla1 a INNER JOIN tabla2 b on a.id = b.id;
```

```
SELECT *
FROM alu_asig INNER JOIN alumnos ON
    alu_asig.id_alumno=alumnos.id
```

Left Outer Joins

LEFT OUTER JOIN muestra todas las filas de la tabla izquierda (tabla 1) y los registros coincidentes de la tabla derecha (tabla 2). El resultado es NULL en el lado derecho, si no hay coincidencia alguna.

```
select * from tabla1 a LEFT OUTER JOIN
        tabla2 b on a.id = b.id;
```

```
SELECT *
FROM tabla1 a
LEFT OUTER JOIN tabla2 b ON a.id = b.id
LIMIT 0 , 30
```

Left Outer Join

+ Opciones

id	letra	id	valores
1	Uno	*NULL*	*NULL*
2	Dos	*NULL*	*NULL*
3	Tres	3	Tres
4	Cuatro	4	Cuatro

Right Outer Joins

RIGHT JOIN muestra todos los registros de la tabla derecha (tabla 2) y los registros coincidentes de la tabla izquierda (tabla1). El resultado es NULL en el lado izquierdo de la tabla, cuando no hay coincidencia. Veamos un ejemplo.

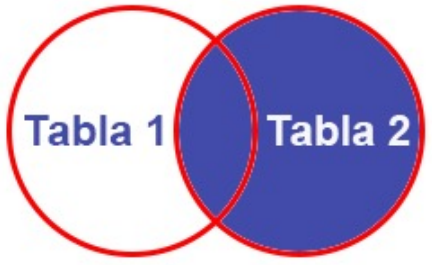
```
select * from tabla1 a RIGHT OUTER JOIN
        tabla2 b on a.id = b.id;
```

```
SELECT *
FROM tabla1 a
RIGHT OUTER JOIN tabla2 b ON a.id = b.id
LIMIT 0 , 30
```

RIGHT OUTER JOIN

+ Opciones

id	letra	id	valores
3	Tres	3	Tres
4	Cuatro	4	Cuatro
NULL	*NULL*	5	Cinco
NULL	*NULL*	6	Seis



Aplicación a nuestra aplicación

```
SELECT *
FROM Alu_asig
RIGHT OUTER JOIN Alumnos  on Alumnos.id = Alu_asig.id_alumno
LEFT OUTER JOIN Asignaturas  on Asignaturas.id = Alu_asig.id_asignatura
```

INSERT

```
INSERT INTO usuarios (nombre, apellidos) VALUES ('Juan','Garcia Pérez');
```

```
INSERT INTO usuarios (nombre, apellidos) VALUES
                        ('Juan','García Pérez'),
                        ('Domingo', 'Sánchez Fernández');
```

REPLACE INTO

REPLACE es como un INSERT pero evitando duplicidades de claves únicas. Puede resultar cómo en determinados casos cuando queremos hacer inserciones incrementales en una tabla.

O en caso de querer modificar un registro que conocemos su id.

```
REPLACE INTO nombre_tabla (columna1, columna2, ...)
                    VALUES (valor1, valor2, ...);
```

REPLACE INTO

```
INSERT INTO coches(matricula, marca, modelo)
VALUES ("112233445566-FF", "Mercedes Benz", "Clase E");

//metemos la misma matricula

REPLACE INTO coches(matricula, marca, modelo)
VALUES ("112233445566-FF", "BMW", "Serie 3");
```

Con este nuevo ejemplo estamos primero insertando un coche con una determinada matrícula, en segundo lugar con el **REPLACE** estaríamos sustituyendo la anterior inserción ya que la matrícula (clave primaria) es la misma.

REPLACE INTO

```
REPLACE INTO Alumnos (id,Nombre,Apellido1,Apellido2,Fecha_nac)
                    VALUES('','Axel','Martinez','Closa','10-06-2000');
```

Y para modificar:

```
REPLACE INTO Alumnos (id,Nombre,Apellido1,Apellido2,Fecha_nac)
                    VALUES(40,'Axel','Martinez','Closa','10-06-1990');
```

REPLACE INTO

```
public function set_alumno($nombre,$apellido1,$apellido2,$fecha_nac){
        $sql = "INSERT INTO Alumnos(Apellido1,Apellido2,Nombre,Fecha_nac)
VALUES ('" . $apellido1 . "', '" . $apellido2 . "', '" . $nombre . "', '" . $fecha_nac . "')";
        if ($this->db->query($sql)) {
           return ($this->db->insert_id);
        }
        else {
            return false;
        }
    }
```

REPLACE INTO

```
public function set_alumno($id,$nombre,$apellido1,$apellido2,$fecha_nac){
        $sql = "REPLACE INTO Alumnos(id,Apellido1,Apellido2,Nombre,Fecha_nac)
VALUES ('".$id."'," . $apellido1 . "', " . $apellido2 . "', " . $nombre . "', " . $fecha_nac .
"')";
        echo $sql;
        if ($this->db->query($sql)) {
           return ($this->db->insert_id);
        }
        else {
            return false;
        }
    }
```

Si $id esta vacio, crea un registro nuevo, si tiene un valor, modifica el registro.

Ejercicio BBDD9

Realizar un analisis funcional de vuestra aplicación

Describir las distintas funciones, pantallas, etc.. Que tendrá la aplicación para poder determinar el modelo, los controladores y las vistas.

SOC Servei d'Ocupació de Catalunya
nascor formación
MINISTERIO DE TRABAJO E INMIGRACIÓN
SERVICIO PÚBLICO DE EMPLEO ESTATAL
Generalitat de Catalunya
Departament de Treball
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Ejercicio BBDD10

En la aplicación de pruebas, realizar una edición sencilla de un formulario. Implementar la inserción y modificación.

UPDATE

UPDATE se utiliza para modificar el contenido de una tabla

```
UPDATE nombre_tabla
SET columna1 = valor1, columna2 = valor2
WHERE columna3 = valor3
```

SET establece los nuevos valores para las columnas indicadas.
WHERE sirve para seleccionar las filas que queremos modificar.

Ojo: Si omitimos la cláusula WHERE, por defecto, modificará los valores en todas las filas de la tabla.

UPDATE

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	RODRIGUEZ
PEDRO	RUIZ	GONZALEZ

Si queremos cambiar el apellido2 'BENITO' por 'RODRIGUEZ' ejecutaremos:

```
UPDATE personas
SET apellido2 = 'RODRIGUEZ'
WHERE nombre = 'ANTONIO'
    AND apellido1 = 'GARCIA'
    AND apellido2 = 'BENITO'
```

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO
PEDRO	RUIZ	GONZALEZ

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***AVG()*** devuelve la **media de valores de una columna numérica**

SELECT AVG (Stock) FROM productos;

La siguiente sentencia muestra el NombreProducto y el Precio de los registros que tienen un Precio por encima de la media:

```
SELECT NombreProducto, Precio
FROM productos
WHERE Precio > (SELECT AVG (Precio) FROM productos);
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***COUNT()*** devuelve el **núm de filas** que cumplen con un criterio:

Número de registros en una tabla

```
SELECT COUNT(*) FROM productos;
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***MAX()*** devuelve el **mayor valor de la columna seleccionada**:

Seleccionamos el producto más caro

```
SELECT MAX(Precio) AS ProductoMasCaro FROM productos;
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***MIN()*** devuelve el **menor valor de la columna seleccionada**:

Seleccionamos el producto más barato

```
SELECT MIN(Precio) AS ProductoMasBarato FROM productos;
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***SUM()*** devuelve la **suma de una columna numérica**:

De la tabla productos vamos a coger el **número total de productos en stock**:

```
SELECT SUM(Stock) AS ProductosTotales FROM productos;
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***UCASE() o LCASE()*** *convierte* el
valor de un campo a mayúsculas o minúsculas

Si queremos obtener todos los **nombres de los productos en mayúsculas**:

```
SELECT ucase(NombreProducto) FROM productos
```

Si queremos obtener todos los **nombres de los productos en minúsculas**:

```
SELECT lcase(NombreProducto) FROM productos
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función **`LENGTH()`** devuelve la longitud de un campo de texto.

Si queremos obtener **nombreProducto** y la **longitud de las descripciones** de los productos de la tabla **productos**:

```
SELECT NombreProducto, LENGTH(Descripcion) as LongitudDescripcion
FROM productos
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función ***NOW()*** devuelve la hora y fecha actuales.

Si queremos devolver el **nombreProducto** y **Precio** de hoy de la tabla **Productos**:

```
SELECT NombreProducto, Precio, Now() AS PrecioProductosHoy
FROM productos
```

FUNCIONES

ID	Nombre	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

La función **DATE_*FORMAT()*** se usa para formatear
cómo se mostrará un campo de tipo decha

Vamos a mostrar un **formato de fecha del ejemplo anterior**:

```
SELECT NombreProducto, Precio,
DATE_FORMAT(Now(), '%d-%m-%Y %H:%i') AS PrecioProductosHoy
FROM productos
```

https://mariadb.com/kb/en/date_format/

FUNCIONES

La función **STR_TO_DATE*()*** se usa para convertir un campo varchar a de tipo decha

Vamos a mostrar a convertir Fecha_nac de nuestra tabla Alumnos:

```
SELECT Fecha_nac,STR_TO_DATE(Fecha_nac,'%d-%m-%Y') FROM Alumnos
```

FUNCIONES

La función **STR_TO_DATE*()*** se usa para convertir
un campo varchar a de tipo decha

Vamos a mostrar a seleccionar Alumnos con menos de 30 años:

```
SELECT Fecha_nac,STR_TO_DATE(Fecha_nac,'%d-%m-%Y')
FROM Alumnos
WHERE STR_TO_DATE(Fecha_nac,'%d-%m-%Y') > now() - INTERVAL 30 YEAR
```

FUNCIONES

La función **STR_TO_DATE*()*** se usa para convertir
un campo varchar a de tipo decha

Vamos a mostrar a la edad de los alumnos a partir de la Fecha_nac que es un varchar:

Para eso usamos la función YEAR que muestra un año
También usamos la Función IF

```
SELECT Nombre, Apellido1, apellido2,Fecha_nac,
YEAR(NOW()) - YEAR(STR_TO_DATE(Fecha_nac,'%d-%m-%Y')) +
IF(DATE_FORMAT(NOW(),'%m-%d') >
DATE_FORMAT(STR_TO_DATE(Fecha_nac,'%d-%m-%Y'),'%m-%d'), 0 , -1 ) AS
EDAD_ACTUAL
FROM Alumnos
```

FUNCIONES

CASE = SWITCH EN SQL

```
SELECT Alumnos.*,
    (CASE DATE_FORMAT(STR_TO_DATE(Fecha_nac,'%d-%m-%Y'),'%c')
        WHEN 1 THEN 'Enero'
        WHEN 2 THEN 'Febrero'
        WHEN 3 THEN 'Marzo'
        WHEN 4 THEN 'Abril'
        WHEN 5 THEN 'Mayo'
        WHEN 6 THEN 'Junio'
        WHEN 7 THEN 'Julio'
        WHEN 8 THEN 'Agosto'
        WHEN 9 THEN 'Septiembre'
        WHEN 10 THEN 'Octubre'
        WHEN 11 THEN 'Noviembre'
        WHEN 12 THEN 'Diciembre'
      ELSE 'No es un mes'
      END) as nombre_mes_nac
FROM Alumnos
```

Ejercicio BBDD11

A partir del analisis funcional realizado en ejercicios anteriores empezar a desarrollar vuestro propyecto.

Ejemplo práctico

Importamos 2 excels en 2 tablas mediante un CSV

TABLE_16
Clau_subfamilia,Nom Categoria

TABLE_17
_subfamilia,IDPadre

Ejemplo práctico

Creamos un listado con id de la categoria, idPadre y nombre categoria

```
SELECT TABLE_16.clau_subfamilia,
            TABLE_17.IDpadre,
             SUBSTR(TABLE_16.nom,4) AS nom
FROM TABLE_16,TABLE_17
WHERE TABLE_16.clau_subfamilia=TABLE_17._subfamilia
```

Ejemplo práctico

Ahora queremos que también muestre
el nombre de la categoria padre

```
select A.clau_subfamilia,
       TABLE_17.IDpadre,
       SUBSTR(A.nom,4) AS NombreHijo,
       SUBSTR(B.nom,4) AS NombrePadre
from TABLE_16 A,
     TABLE_17 ,
     TABLE_16 B
where A.clau_subfamilia=TABLE_17._subfamilia
AND B.clau_subfamilia=TABLE_17.IDpadre
order by 4,3
```

Ejemplo práctico

En la consulta anterior faltan los registros con idpadre=0 porque no tienen nombre en la tabla B. Para que se muestren hacemos:

```
SELECT A.clau_subfamilia,
            TABLE_17.IDpadre,
            SUBSTR(A.nom,4) AS NombreHijo,
            SUBSTR(B.nom,4) AS NombrePadre
from TABLE_16 A INNER JOIN TABLE_17
                        ON A.clau_subfamilia=TABLE_17._subfamilia
       LEFT OUTER JOIN TABLE_16 AS B
                        ON B.clau_subfamilia=TABLE_17.IDpadre
order by 4,3
```

Funciones / Procedimientos

Un procedimiento o función almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado.

Funciones / Procedimientos

```
CREATE FUNCTION hola_mundo()
        RETURNS VARCHAR(100)
RETURN 'Hola Mundo!'
```

```
SELECT hola_mundo()
```

Funciones / Procedimientos

Variables en funciones

```
DELIMITER //
CREATE FUNCTION hola_Mundo2()
                    RETURNS VARCHAR(30)
BEGIN
  DECLARE salida VARCHAR(30) DEFAULT 'Hola mundo';
  SET salida = 'Hola mundo con variables';
  RETURN salida;
END
//
```

Funciones / Procedimientos

Variables en funciones

```
DELIMITER //
CREATE FUNCTION hola_Mundo3(entrada VARCHAR(20))
                RETURNS VARCHAR(20)
BEGIN
  DECLARE salida VARCHAR(20);
  SET salida = entrada;
  RETURN salida;
END
//
      select hola_mundo3("Hola Mundo cruel")
```

Funciones / Procedimientos

Ejemplo practico

A partir de un id de alumno devolvemos el nombre completo

```
DELIMITER //
CREATE FUNCTION nombre_completo(id_alu int)
                        RETURNS varchar(200)
BEGIN
  DECLARE salida VARCHAR(200);
  SELECT
       CONCAT(Alumnos.Apellido1,' ',Alumnos.apellido2,', ',Alumnos.Nombre)
       INTO salida
  FROM Alumnos
  WHERE Alumnos.id=id_alu;
  RETURN salida;
END
```

Funciones / Procedimientos

Ejemplo practico

A partir de un id de alumno devolvemos el nombre completo

SELECT nombre_completo(id) FROM Alumnos

nombre_completo(id)
Mata Enrique, Fátima
Fernández Lorencio, Virtudes
Alvarez Fernandez, Borja
Fernandez Gomez, Borja
Garcia Gomez, Rafael
Lopez Gomez, Rafael
Palacio Rodríguez, Adri Benito
García Lorca, Federico
Fernández García, Mariano
Martínez Machín, Saúl
Alonso de Andrés, Pere
Mulleras Vinzia, Borja
Mulleras Humet, Alejandra
Matas Gou, Borja
Caze Casanovas, Fatima

Funciones / Procedimientos

Ejemplo práctico

Ahora creamos una función que devuelva el nombre del curso a partir del id del curso

```
DELIMITER //
CREATE FUNCTION nombre_curso(id_curso int) RETURNS varchar(200)
BEGIN
  DECLARE salida VARCHAR(200);
  SELECT
        Asignaturas.Nombre INTO salida
  FROM Asignaturas
  WHERE Asignaturas.id=id_curso;
  RETURN salida;
END
//
```

Funciones / Procedimientos

Ejemplo practico
Consultamos nombre curso

SELECT nombre_curso(id) FROM Asignaturas

nombre_curso(id)
Desarrollo de aplicaciones Web
Desarrollo de Páginas Web

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Funciones / Procedimientos

Ejemplo practico
Consultar relación asignaturas alumnos

```
SELECT nombre_curso(id_asignatura),
nombre_completo(id_alumno)
 FROM Alu_asig
ORDER BY 1,2
```

nombre_curso(id_asignatura)	nombre_completo(id_alumno)
Desarrollo de aplicaciones Web	Alonso de Andrés, Pere
Desarrollo de aplicaciones Web	Caze Casanovas, Fatima
Desarrollo de aplicaciones Web	Fernández García, Mariano
Desarrollo de aplicaciones Web	Fernandez Gomez, Borja
Desarrollo de aplicaciones Web	Fernández Lorencio, Virtudes
Desarrollo de aplicaciones Web	Garcia Gomez, Rafael
Desarrollo de aplicaciones Web	García Lorca, Federico
Desarrollo de aplicaciones Web	Martínez Machín, Saúl
Desarrollo de aplicaciones Web	Matas Gou, Borja
Desarrollo de aplicaciones Web	Mulleras Vinzia, Borja
Desarrollo de Páginas Web	Fernandez Gomez, Borja
Desarrollo de Páginas Web	Fernández Lorencio, Virtudes
Desarrollo de Páginas Web	Garcia Gomez, Rafael
Desarrollo de Páginas Web	Lopez Gomez, Rafael
Desarrollo de Páginas Web	Palacio Rodríguez, Adri Benito

Funciones / Procedimientos

SENTENCIA IF

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN
statement_list] ...
    [ELSE statement_list]
END IF
```

Funciones / Procedimientos

SENTENCIA IF

```
DELIMITER //
CREATE FUNCTION `tipo_sueldo`(sueldo INT)
RETURNS varchar(10)
BEGIN
DECLARE a VARCHAR(10);

IF sueldo>1500 then
     RETURN 'ALTO';
ELSE
     RETURN 'BAJO';
END IF;
END
//
```

Funciones / Procedimientos

SENTENCIA CASE

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

Funciones / Procedimientos

SENTENCIA CASE

```
DELIMITER //
CREATE FUNCTION formato_mes_esp(fecha varchar(20))
RETURNS varchar(30)
BEGIN
DECLARE num_mes int;
DECLARE dia int;
DECLARE anyo int;
DECLARE mes varchar(30);
SELECT DATE_FORMAT(STR_TO_DATE(fecha,'%d-%m-%Y'),'%c') into
num_mes;
SELECT DATE_FORMAT(STR_TO_DATE(fecha,'%d-%m-%Y'),'%d') into dia;
SELECT DATE_FORMAT(STR_TO_DATE(fecha,'%d-%m-%Y'),'%Y') into anyo;
```

Funciones / Procedimientos

SENTENCIA CASE

```
CASE
        WHEN num_mes=1 THEN set mes='Enero';
        WHEN num_mes=2 THEN set mes='Febrero';
        WHEN num_mes=3 THEN set mes='Marzo';
        WHEN num_mes=4 THEN set mes='Abril';
        WHEN num_mes=5 THEN set mes='Mayo';
        WHEN num_mes=6 THEN set mes='Junio';
        WHEN num_mes=7 THEN set mes='Julio';
        WHEN num_mes=8 THEN set mes='Agosto';
        WHEN num_mes=9 THEN set mes='Septiembre';
        WHEN num_mes=10 THEN set mes='Octubre';
        WHEN num_mes=11 THEN set mes='Noviembre';
        WHEN num_mes=12 THEN set mes='Diciembre';
END CASE;
RETURN concat(dia,' de ',mes,' de ',anyo);
END
//
```

Funciones / Procedimientos

CURSORES

Los cursores le permiten iterar los resultados de la consulta uno por línea. DECLARE comando DECLARE se usa para iniciar el cursor y asociarlo con una consulta SQL específica:

DECLARE student **CURSOR FOR** SELECT name FROM student;

Digamos que vendemos productos de algunos tipos. Queremos contar cuántos productos de cada tipo existen.

Funciones / Procedimientos

CREAMOS TABLAS

```
CREATE TABLE product
  ( id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    type VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL );

CREATE TABLE product_type
  ( name VARCHAR(50) NOT NULL PRIMARY KEY );

CREATE TABLE product_type_count
  ( type VARCHAR(50) NOT NULL PRIMARY KEY,
   count INT(10) UNSIGNED NOT NULL DEFAULT 0 );
```

Funciones / Procedimientos

INSERTAMOS VALORES

```
INSERT INTO product_type (name)
 VALUES ('dress'),
                ('food');

INSERT INTO product (type, name)
VALUES ('dress', 'T-shirt'),
                ('dress', 'Trousers'),
                ('food', 'Apple'),
                ('food', 'Tomatoes'),
                ('food', 'Meat');
```

Funciones / Procedimientos

DEFINIMOS EL PROCEDIMIENTO

```
DELIMITER //
 CREATE PROCEDURE product_count()
   BEGIN
       DECLARE p_type VARCHAR(255);
       DECLARE p_count INT(10) UNSIGNED;
       DECLARE done INT DEFAULT 0;

       DECLARE product CURSOR FOR
            SELECT type, COUNT(*)
            FROM product
            GROUP BY type;

       DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET
done = 1;
```

Funciones / Procedimientos

DEFINIMOS EL PROCEDIMIENTO

```
TRUNCATE product_type_count;

OPEN product;
  REPEAT FETCH product INTO p_type, p_count;
    IF NOT done THEN
       INSERT INTO product_type_count
      SET type = p_type, count = p_count;
    END IF;
  UNTIL done END REPEAT;
CLOSE product;
END
//
```

SOC Servei d'Ocupació de Catalunya

nascor formación

MINISTERIO DE TRABAJO E INMIGRACIÓN

SERVICIO PÚBLICO DE EMPLEO ESTATAL

Certificado de profesionalidad IFCD0210

Generalitat de Catalunya
Departament de Treball

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur