Bases de Datos Relacionales parte 2

Fundación ONCE | Inserta | Cofinanciado por la Unión Europea | Ministerio de Trabajo y Economía Social | Fondos Europeos

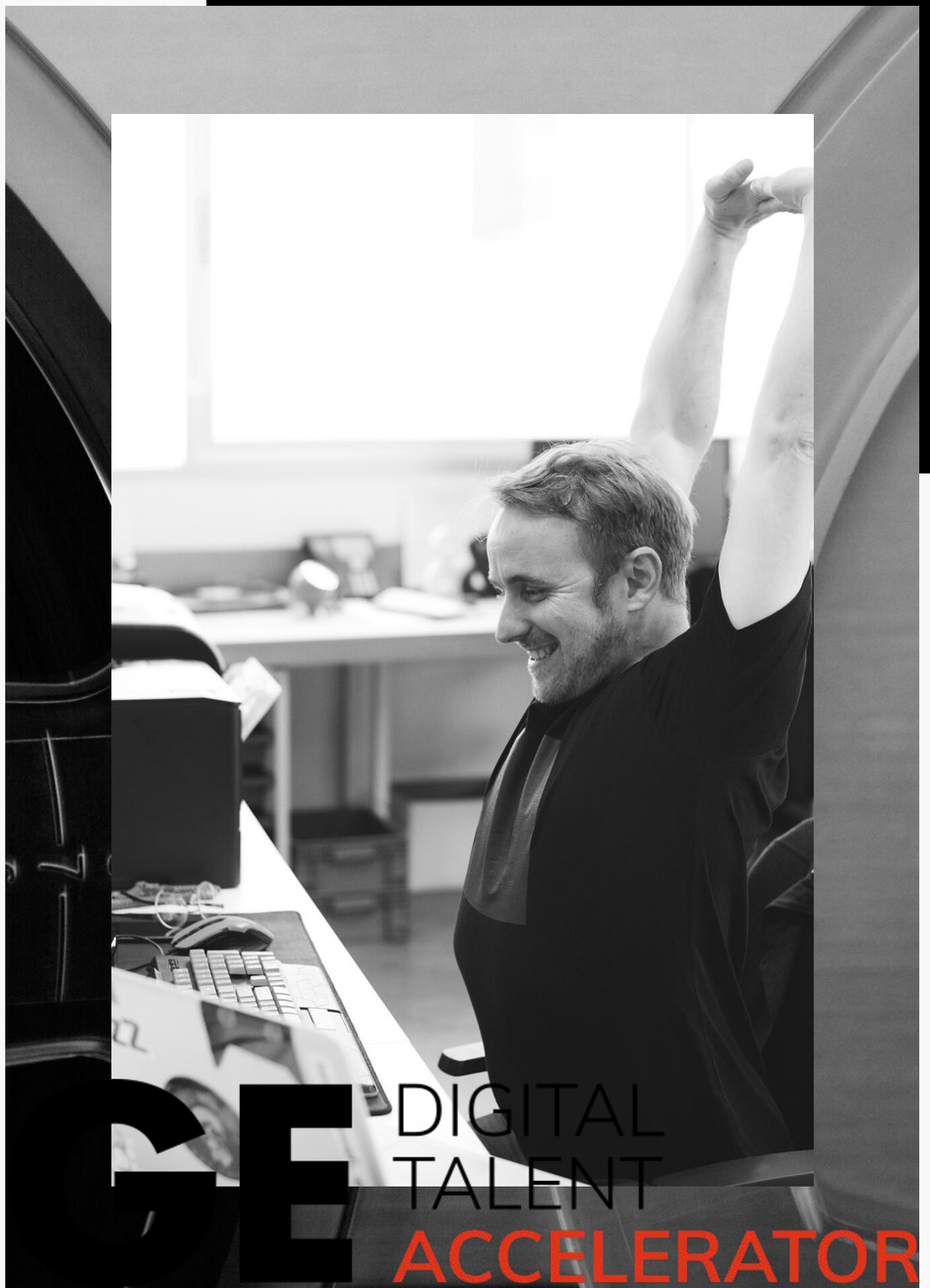
THE BRIDGE
DIGITAL TALENT ACCELERATOR

Índice

Relaciones

Dibuja tu primer diagrama

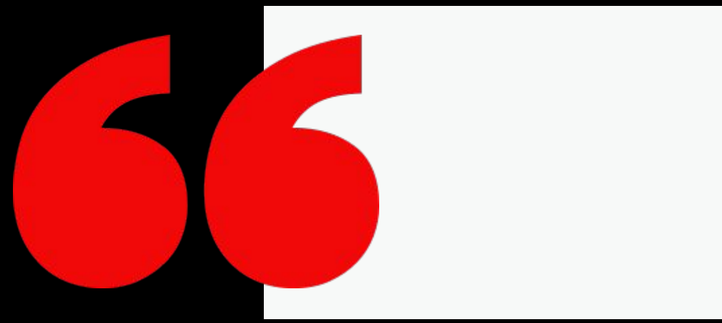
SQL Joins



THE BRIDGE DIGITAL TALENT ACCELERATOR

“En una clínica veterinaria, un/a veterinari@ se le asigna una mascota para que la atienda. Esta mascota tiene un dueño, que necesitamos conocer para avisarlo en casos especiales”.

“En una librería Online se venden libros. Cada libro pertenece a un departamento. Los clientes exploran nuestro catálogo de libros, selecciona libros y los colocan en la cesta de la compra. Al finalizar la compra, la cesta de la compra se convierte es un pedido efectivo que se envía al cliente.”



“Se desea diseñar la base de datos de un Instituto. En la base de datos se desea guardar los datos de los profesores del Instituto (DNI, nombre, dirección y teléfono).

Los profesores imparten módulos, y cada módulo tiene un código y un nombre.

Cada alumno puede cursar uno o varios módulos.

De cada alumno se desea guardar el nº de expediente, nombre, apellidos y fecha de nacimiento. Los profesores pueden impartir varios módulos, pero un módulo sólo puede ser impartido por un profesor. Cada curso tiene un grupo de alumnos, uno de los cuales es el delegado del grupo”

Creamos nuestro primer diagrama con Workbench

“En una red social un usuario puede crear varias publicaciones. Las publicaciones tienen comentarios y esos comentarios los hacen los usuarios.”

Creando otra tabla

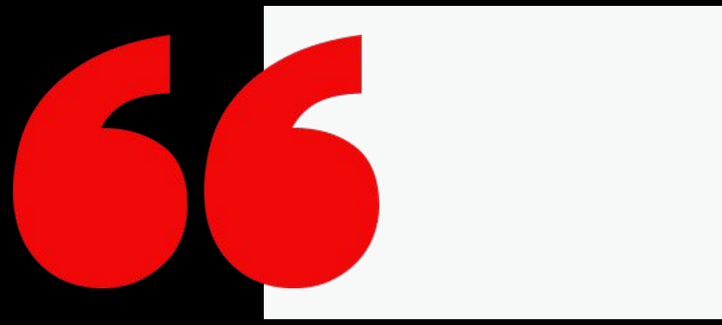
- Creamos una nueva tabla, la tabla Posts.
- Y como novedad añadimos una **foreign key**. La **foreign key** que queremos es el "user_id" y tenemos que especificar la referencia y es la tabla del usuario y su **primary key** "id".

```
CREATE TABLE posts(
    id INT AUTO_INCREMENT,
    user_id INT,
    title VARCHAR(100),
    body VARCHAR(255),
    publish_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Insert posts

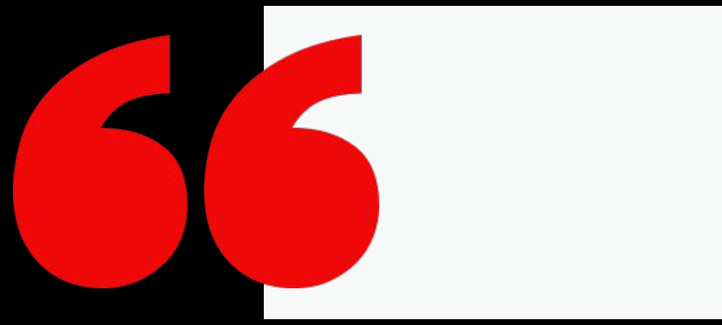
- Insertamos datos en nuestra nueva tabla(el user_id, el title y el body).

```
INSERT INTO posts(user_id, title, body)
VALUES (1, 'Post One', 'This is post
one'),
(3, 'Post Two', 'This is post two'),
(1, 'Post Three', 'This is post three'),
(2, 'Post Four', 'This is post four'),
(1, 'Post Five','This is post five'),
(2, 'Post Six', 'This is post six'),
(1, 'Post Seven', 'This is post seven'),
(3, 'Post Eight', 'This is post eight');
```



SQL Joins

Unir varias tablas en la misma consulta SQL. Una cláusula JOIN se usa para combinar filas de dos o más tablas, según una columna relacionada entre ellas.



Inner Join

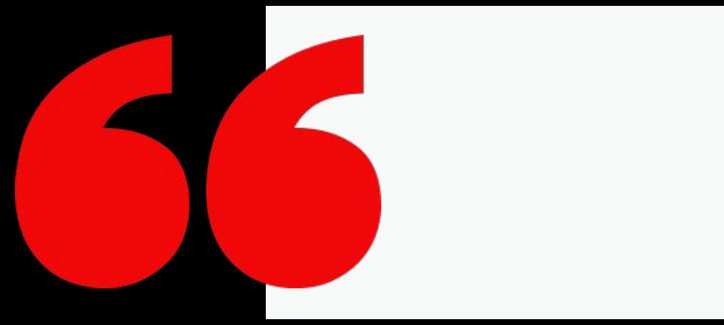
INNER JOIN selecciona todas las filas de ambas tablas participantes siempre que haya una coincidencia entre las columnas.

INNER JOIN une dos tablas de acuerdo con la coincidencia de ciertos criterios utilizando un operador de comparación.

Inner Join

- La idea aquí es obtener el nombre de usuario, el apellido junto con el título de sus publicaciones y la fecha de publicación.
- Entonces usamos INNER JOIN con la tabla de publicaciones y para saber dónde unir estas dos tablas, usamos la primary key en la tabla de usuarios hacia la foreign key en la tabla de publicaciones.

```sql
SELECT
  users.first_name,
  users.last_name,
  posts.title,
  posts.publish_date
FROM users
INNER JOIN posts
ON users.id = posts.user_id;
```



Compliquemos esto un poco más ...

¿Un usuario puede hacer comentarios?

¿Y una publicación puede tener comentarios?

Relaciones parte 2

- Creamos la tabla de comentarios.
- Y como novedad añadimos dos foreign key. La **foreign key** que queremos es el **"user_id"** y tenemos que especificar la referencia y es la tabla de usuarios y su **primary key "id"**. Y la **foreign key del "post_id"** y tenemos que especificar la referencia y es la tabla de **publicaciones y su primary key "id"**.

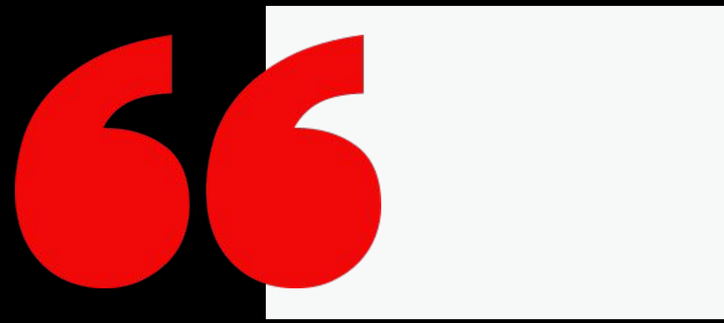
```
CREATE TABLE comments(
    id INT AUTO_INCREMENT,
    post_id INT,
    user_id INT,
    body TEXT,
    PRIMARY KEY(id),
    FOREIGN KEY(user_id) REFERENCES users(id),
    FOREIGN KEY(post_id) REFERENCES posts(id)
);
```

Insertar comentarios

- Insertamos datos en nuestra nueva tabla.
- Insertaremos post_id, user_id y el body.

```
INSERT INTO comments(post_id,
user_id, body) VALUES
(1, 3, 'This is comment one'),
(2, 1, 'This is comment two'),
(5, 3, 'This is comment three'),
(2, 2, 'This is comment four'),
(1, 2, 'This is comment five');
```

Left Join & Right Join



La palabra clave LEFT JOIN devuelve todos los registros de la tabla izquierda (tabla1) y los registros coincidentes de la tabla derecha (tabla2). El resultado es NULO desde el lado derecho, si no hay coincidencia.

La palabra clave RIGHT JOIN devuelve todos los registros de la tabla derecha (tabla2) y los registros coincidentes de la tabla izquierda (tabla1). El resultado es NULL desde el lado izquierdo, cuando no hay coincidencia.

Left Join & Right Join

```sql
SELECT
comments.body,
posts.title
FROM comments
LEFT JOIN posts
ON posts.id = comments.post_id;
```

```sql
SELECT
comments.body,
posts.title
FROM comments
RIGHT JOIN posts
ON posts.id = comments.post_id;
```

Relacionar varias tablas

- La idea aquí es **obtener todos los comentarios** con el **título de la publicación** en la que se encuentra ese comentario y también estamos obteniendo la **persona que dejó el comentario.**

```
SELECT
comments.body,
posts.title,
users.first_name,
users.last_name
FROM comments
INNER JOIN posts ON posts.id = comments.post_id
INNER JOIN users ON users.id = comments.user_id;
```