# TIPOS DE RELACIONES

/\*

Relación 1:1 (uno a uno)

Relación que indica que un registro en la tabla A se relaciona

con un sólo registro en la tabla B y viceversa.

\*/

-- El campo user\_id de la tabla "dni" es clave foránea de la clave primaria user\_id de la tabla "users"

-- (Un usuario sólo puede tener un DNI. Un DNI sólo puede estar asociado a un usuario)

CREATE TABLE dni(

dni\_id int AUTO\_INCREMENT PRIMARY KEY,

dni\_number int NOT NULL,

user\_id int,

UNIQUE(dni\_id),

FOREIGN KEY(user\_id) REFERENCES users(user\_id)

);

/\*

Relación 1:N (uno a muchos)

Relación que indica que un registro en la tabla A puede tener varios registros relacionados en la tabla B, pero un registro en la tabla B se relaciona con un sólo registro en la tabla A.

\*/

CREATE TABLE companies(

company\_id int AUTO\_INCREMENT PRIMARY KEY,

name varchar(100) NOT NULL

);

-- El campo company\_id de la tabla "users" es clave foránea de la clave primaria company\_id de la tabla "companies"

-- (Un empleado (usuario) sólo puede tener una empresa, pero una empresa puede tener muchos empleados (usuarios))

ALTER TABLE users

ADD CONSTRAINT fk\_companies

FOREIGN KEY(company\_id) REFERENCES companies(company\_id)

/\*

Relación N:M (muchos a muchos)

Relación que indica que un un registro en la tabla A puede relacionarse

con varios registros en la tabla B y viceversa.

Requiere una tabla intermedia o de unión para establecer la relación.

\*/

CREATE TABLE languages(

language\_id int AUTO\_INCREMENT PRIMARY KEY,

name varchar(100) NOT NULL

);

-- El campo user\_id y language\_id de la tabla intermedia "users\_languages" es clave foránea de las

-- claves primarias user\_id de la tabla "users" y de language\_id de la tabla "languages"

-- Un usuario puede conoces muchos lenguajes. Un lenguaje puede ser conocido por muchos usuarios.

CREATE TABLE users\_languages(

users\_language\_id int AUTO\_INCREMENT PRIMARY KEY,

user\_id int,

language\_id int,

FOREIGN KEY(user\_id) REFERENCES users(user\_id),

FOREIGN KEY(language\_id) REFERENCES languages(language\_id),

UNIQUE (user\_id, language\_id)

);

/\*

Relación de Auto-Referencia

Relación que indica que un un registro en la tabla A puede

relacionarse con otro registro de la tabla A.

\*/

/\*

INSERT y UPDATE para trabajar con JOIN

\*/

/\*

1:1

Lección 16.1: https://youtu.be/OuJerKzV5T0?t=14994

\*/

-- "dni" (Relación 1:1)

INSERT INTO dni (dni\_number, user\_id) VALUES (11111111, 1);

INSERT INTO dni (dni\_number, user\_id) VALUES (22222222, 2);

INSERT INTO dni (dni\_number, user\_id) VALUES (33333333, 3);

INSERT INTO dni (dni\_number) VALUES (44444444);

/\*

1:N

Lección 16.2: https://youtu.be/OuJerKzV5T0?t=15203

\*/

-- "companies" y "users" (Relación 1:N)

INSERT INTO companies (name) VALUES ('MoureDev');

INSERT INTO companies (name) VALUES ('Apple');

INSERT INTO companies (name) VALUES ('Google');

UPDATE users SET company\_id = 1 WHERE user\_id = 1;

UPDATE users SET company\_id = 2 WHERE user\_id = 3;

UPDATE users SET company\_id = 3 WHERE user\_id = 4;

UPDATE users SET company\_id = 1 WHERE user\_id = 7;

/\*

N:M

Lección 16.3: https://youtu.be/OuJerKzV5T0?t=15474

\*/

-- "languages" y "users\_languages" (Relación N:M)

INSERT INTO languages (name) VALUES ('Swift');

INSERT INTO languages (name) VALUES ('Kotlin');

INSERT INTO languages (name) VALUES ('JavaScript');

INSERT INTO languages (name) VALUES ('Java');

INSERT INTO languages (name) VALUES ('Python');

INSERT INTO languages (name) VALUES ('C#');

INSERT INTO languages (name) VALUES ('COBOL');

INSERT INTO users\_languages (user\_id, language\_id) VALUES (1, 1);

INSERT INTO users\_languages (user\_id, language\_id) VALUES (1, 2);

INSERT INTO users\_languages (user\_id, language\_id) VALUES (1, 5);

INSERT INTO users\_languages (user\_id, language\_id) VALUES (2, 3);

INSERT INTO users\_languages (user\_id, language\_id) VALUES (2, 5);

# INNER JOIN (JOIN)

-- Realiza un JOIN de manera incorrecta, ya que no existe un campo de relación

SELECT \* FROM users

INNER JOIN dni;

-- Obtiene los datos de los usuarios que tienen un dni

SELECT \* FROM users

INNER JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene los datos de los usuarios que tienen un dni (JOIN es lo mismo que INNER JOIN)

SELECT \* FROM users

JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene el nombre y el dni de los usuarios que tienen un dni y los ordena por edad

SELECT name, dni\_number FROM users

JOIN dni

ON users.user\_id = dni.user\_id

ORDER BY age ASC;

-- Obtiene los datos de los usuarios que tienen empresa

SELECT \* FROM users

JOIN companies

ON users.company\_id = companies.company\_id;

-- Obtiene los datos de las empresas que tienen usuarios

SELECT \* FROM companies

JOIN users

ON users.company\_id = companies.company\_id;

-- Obtiene el nombre de las empresas junto al nombre de sus usuarios

SELECT companies.name, users.name FROM companies

JOIN users

ON companies.company\_id = users.company\_id;

-- Obtiene los nombres de usuarios junto a los lenguajes que conocen

SELECT users.name, languages.name

FROM users\_languages

JOIN users ON users\_languages.user\_id=users.user\_id

JOIN languages ON users\_languages.language\_id=languages.language\_id;

-- Obtiene los nombres de usuarios junto a los lenguajes que conocen (utilizando otro orden de relación entre tablas)

SELECT users.name, languages.name

FROM users

JOIN users\_languages ON users.user\_id=users\_languages.user\_id

JOIN languages ON users\_languages.language\_id=languages.language\_id;

# LEFT JOIN

-- Obtiene los datos de todos los usuarios junto a su dni (lo tenga o no)

SELECT \* FROM users

LEFT JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene el nombre de todos los usuarios junto a su dni (lo tenga o no)

SELECT name, dni\_number FROM users

LEFT JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene todos los dni junto al nombre de su usuario (lo tenga o no)

SELECT name, dni\_number FROM dni

LEFT JOIN users

ON users.user\_id = dni.user\_id;

-- Obtiene el nombre de todos los usuarios junto a sus lenguajes (los tenga o no)

SELECT users.name, languages.name

FROM users

LEFT JOIN users\_languages ON users.user\_id=users\_languages.user\_id

LEFT JOIN languages ON users\_languages.language\_id=languages.language\_id;

RIGHT JOIN

-- Obtiene todos los dni junto a su usuario (lo tenga o no)

SELECT \* FROM users

RIGHT JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene todos los dni junto al nombre de su usuario (lo tenga o no)

SELECT name, dni\_number FROM users

RIGHT JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene el nombre de todos los usuarios junto a su dni (lo tenga o no)

SELECT name, dni\_number FROM dni

RIGHT JOIN users

ON users.user\_id = dni.user\_id;

-- Obtiene el nombre de todos los lenguajes junto a sus usuarios (los tenga o no)

SELECT users.name, languages.name

FROM users

RIGHT JOIN users\_languages ON users.user\_id=users\_languages.user\_id

RIGHT JOIN languages ON users\_languages.language\_id=languages.language\_id;

UNION (FULL JOIN)

-- UNION elimina duplicados

-- Obtiene todos los id de usuarios de las tablas dni y usuarios (exista o no relación)

SELECT users.user\_id AS u\_user\_id, dni.user\_id AS d\_user\_id

FROM users

LEFT JOIN dni

ON users.user\_id = dni.user\_id

UNION

SELECT users.user\_id AS user\_id, dni.user\_id AS d\_user\_id

FROM users

RIGHT JOIN dni

ON users.user\_id = dni.user\_id;

-- Obtiene todos los datos de las tablas dni y usuarios (exista o no relación)

SELECT \*

FROM users

LEFT JOIN dni

ON users.user\_id = dni.user\_id

UNION

SELECT \*

FROM users

RIGHT JOIN dni

ON users.user\_id = dni.user\_id;

-- UNION ALL mantiene duplicados