**Listar las bases de datos** que haya:

**Crea base de datos**:

Nos **conecta a una base de datos**: en este caso a la que hemos llamado second\_database.

Una vez dentro de una base de datos **lista las tablas** de esa base de datos.

**Crea tabla**:

Se ven más **detalles** de una **tabla**:

**Agrega columna y tipo**:

**Eliminar tablas**:

ALTER TABLE table\_name DROP COLUMN column\_name;

Un tipo de dato común es varchar. Es un carácter variable, por lo que hay que darle una dimensión. Varchar(30) por ejemplo. 30 es el número de caracteres máximo.

**Cambiar nombre** a una **columna**:

**Insertar datos**:

INSERT INTO table\_name(column\_1, column\_2) VALUES(value1, value2);

-> ‘Samus’ va con comillas porque es un varchar.

**Ver** los **datos** de una **tabla**:

SELECT columns FROM table\_name;

**Eliminar filas**:

DELETE FROM table\_name WHERE condition;

**Eliminar tabla**:

DROP TABLE table\_name;

**Modificar nombre de una base de datos**:

ALTER DATABASE database\_name RENAME TO new\_database\_name;

drop database second\_database; -> eliminar una base de datos.

Al crear una columna, si le pones de tipo de dato serial = INT with a NOT NULL constrain. Y le pone autoincremento.

alter table more\_info add column weight numeric(4, 1); -> con esto le decimos que el número puede tener 4 digitos y uno de ellos va a los decimales.

alter table characters add column name varchar(30) not null; añadimos la característica de NOT NULL

añadir varios datos a la vez:

INSERT INTO characters(name, homeland, favorite\_color)

VALUES('Mario', 'Mushroom Kingdom', 'Red'),

('Luigi', 'Mushroom Kingdom', 'Green'),

('Peach', 'Mushroom Kingdom', 'Pink');

Si pones NULL en algún dato, sale vacío.

Modificar datos de una fila:

UPDATE table\_name SET column\_name=new\_value WHERE condition;

Ordenar:

SELECT columns FROM table\_name ORDER BY column\_name;

En columns se pondria \* para elegir todas.

Como poner una llave:

ALTER TABLE table\_name ADD PRIMARY KEY(column\_name);

Las restricciones como la llave maestra se pueden quitar:

ALTER TABLE table\_name DROP CONSTRAINT constraint\_name;

Para relacionar tablas se necesita una llave de acceso(Foreing Key):

ALTER TABLE table\_name ADD COLUMN column\_name DATATYPE REFERENCES referenced\_table\_name(referenced\_column\_name);

Añadir restriccion:

ALTER TABLE table\_name ALTER COLUMN column\_name SET NOT NULL;

Insertar filas en una table que tiene relacionada otra:

INSERT INTO more\_info(birthday, height, weight, character\_id) VALUES('1985-10-18', 173, 52.2, 3);

El 3 es el id que corresponde a alguien en la otra tabla.

alter table more\_info rename COLUMN height to height\_in\_cm; -> Renombrar columna

Crear tablas con columnas:

CREATE TABLE table\_name(column\_name DATATYPE CONSTRAINTS);

alter table sounds add column filename varchar(40) not null unique; -> añadir una table con varios atributos.

La Foreing-Key puede ser “one to many” por ejemplo un personaje que puede tener varios sonidos:

ALTER TABLE table\_name ADD COLUMN column\_name DATATYPE CONSTRAINT REFERENCES referenced\_table\_name(referenced\_column\_name);

"Many-to-many" relationships usually use a **junction** table to link two tables together, forming two "one-to-many" relationships.

En estos casos es necesario crear una tabla nueva que una dos tablas.

The foreign keys you set before were added when you created the column. You can set an existing column as a foreign key like this:

ALTER TABLE table\_name ADD FOREIGN KEY(column\_name) REFERENCES referenced\_table(referenced\_column);

Set the character\_id column you just added as a foreign key that references the character\_id from the characters table.

ALTER TABLE character\_actions ADD FOREIGN KEY(character\_id) REFERENCES characters(character\_id);

Crear primary key para dos columnas:

Every table should have a primary key. Your previous tables had a single column as a primary key. This one will be different. You can create a primary key from two columns, known as a **composite** primary key. Here's an example:

ALTER TABLE table\_name ADD PRIMARY KEY(column1, column2);

Use character\_id and action\_id to create a composite primary key for this table.

You can see the character\_id there so you just need to find the matching id in the characters table to find out who it's for. Or... You added that as a foreign key, that means you can get all the data from both tables with a JOIN command:

SELECT columns FROM table\_1 FULL JOIN table\_2 ON table\_1.primary\_key\_column = table\_2.foreign\_key\_column;

This shows the "one-to-many" relationship. You can see that some of the characters have more than one row because they have **many** sounds. How can you see all the info from the characters, actions, and character\_actions tables? Here's an example that joins three tables:

SELECT columns FROM junction\_table

FULL JOIN table\_1 ON junction\_table.foreign\_key\_column = table\_1.primary\_key\_column

FULL JOIN table\_2 ON junction\_table.foreign\_key\_column = table\_2.primary\_key\_column;

SELECT \* FROM character\_actions FULL JOIN characters ON character\_actions.character\_id = characters.character\_id FULL JOIN actions ON character\_actions.action\_id = actions.action\_id;