

PostgreSQL

Universidad Autónoma de Madrid  
Máster en Bioinformática y Biología Computacional  
Bases de Datos Relacionales

Elena Aguayo Jara

30/11/2022

# INDICE

|   |    |
|---|----|
| 1. Introducción.....  | 3  |
| 2. Modelo Entidad-Relación .....  | 4  |
| 3. Modelo relacional.....   | 5  |
| 4. Create Tables .....  | 5  |
| 5. Querys.....  | 6  |
| a. Artistas procedentes de Murcia.....  | 6  |
| b. Número de discos publicados por cada artista ordenados de mayor a menor..... | 7  |
| c. Añadir columna 'edad' a la tabla cliente_1 y aplicar not null.....           | 7  |
| d. Concierto al que acude 'Elena Pardo' .....                                   | 8  |
| e. Enlace web donde aparece 'wizink' .....                                      | 8  |
| f. Bandas de música creadas antes del año 2000 .....                            | 8  |
| g. Bandas de música con mujeres integrantes y el número.....                    | 8  |
| h. Banda de música con id=2.....  | 9  |
| i. Número de conciertos en el Wizink Center .....                               | 9  |
| j. Número de grupos que tocan música Pop.....                                   | 9  |
| k. Suma del número de mujeres en total .....                                    | 10 |
| l. Banda de música con la entrada más barata.....                               | 10 |
| m. Agrupar cuántos grupos tocan cada estilo de música .....                     | 10 |
| n. Número de conciertos en cada ciudad ordenados en sentido ascendente.....     | 10 |
| o. Grupos de música con 2 discos publicados.....                                | 11 |
| p. Precio de las entradas en Madrid ordenadas en sentido ascendente.....        | 11 |
| 6. Trigger .....  | 11 |
| 7. Referencias .....  | 13 |

## **1. Introducción**

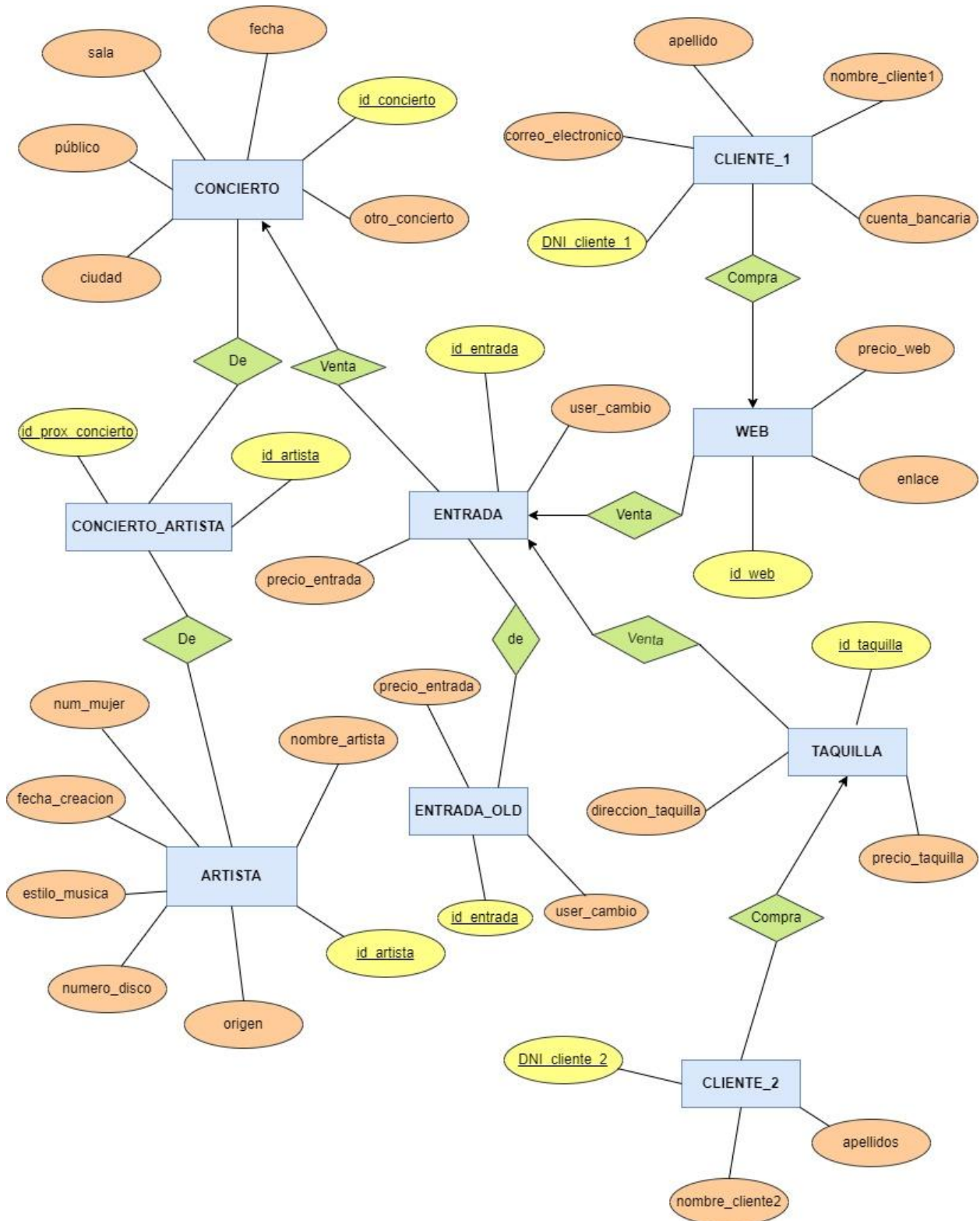
Se propone realizar una base de datos en lenguaje SQL mediante el gestor PostgreSQL. En este caso, la base de datos recoge los conciertos de música que se van a realizar en ciertas ciudades para poder tener organizadas las fechas, salas de concierto y, por supuesto, qué artista o bandas de música realizan esos conciertos. Además, sirve de consulta para obtener información acerca de estos artistas ya que recoge el número de discos publicados, lugar de procedencia y desde qué fecha están activos; y, como dato adicional, también se puede consultar el número de mujeres integrantes de cada banda o cuántas hay como artistas en solitario, de este modo se puede ver reflejada la escasa cantidad de mujeres que consiguen abrirse paso en el mundo de la música.

Por otro lado, esta base de datos también guarda la información de aquellas personas que deciden comprar las entradas de los conciertos en taquilla o por diversas páginas web, quedando siempre reflejado el precio de las entradas en los distintos puntos de venta.

A continuación, se encuentra el proceso de creación de la base de datos de forma detallada, recogiendo el modelo entidad-relación, modelo relacional, queries con sus resultados y el trigger creado.

## 2. Modelo Entidad-Relación

El modelo Entidad-Relación se encuentra aquí representado a, realizado mediante el programa *diagrams*. La relación entre las entidades Concierto-Artista es de muchos a muchos, el resto son de muchos a uno. Se encuentra representada la entidad entrada\_old, que un principio se encuentra vacía hasta que se ejecuta el trigger posteriormente explicado.



### 3. Modelo relacional

Concierto (id\_prox\_concierto, fecha, ciudad, sala, publico, otro\_concierto).

Artista (id\_artista, id\_prox\_concierto ↑, nombre\_artista, estilo\_musica, fecha\_creacion, numero\_disco, origen, num\_mujer).

Concierto\_Artista (id\_prox\_concierto ↑, id\_artista ↑).

Entrada (id\_entrada, id\_prox\_concierto ↑, precio\_entrada, user\_cambio).

Entrada\_old ((id\_entrada ↑, id\_prox\_concierto ↑, precio\_entrada, user\_cambio).

Web (id\_web, id\_prox\_entrada ↑, id\_artista ↑, enlace, precio\_web).

Taquilla (id\_taquilla, id\_prox\_concierto ↑, id\_artista ↑, direccion\_taquilla, precio\_taquilla).

Cliente\_1 (DNI\_cliente\_1, id\_web ↑, id\_prox\_concierto ↑, id\_artista ↑, nombre\_cliente1, apellidos, correo\_electronico, cuenta\_bancaria).

Cliente\_2 (DNI\_cliente\_2, id\_taquilla ↑, id\_prox\_concierto ↑, id\_artista ↑, DNI, nombre\_cliente2, apellidos).

### 4. Create Tables

Las create tables necesarias para cargar la base de datos son las siguientes mostradas a continuación. Se encuentran también en el archivo .txt adjutando en la carpeta, junto a los inserts, queries y el trigger.

```
CREATE TABLE Concierto
(
    id_prox_concierto integer PRIMARY KEY, -- Identificador único
    fecha timestamp, -- Fecha y hora a la que es el concierto
    ciudad varchar(20), -- Ciudad donde es el concierto
    sala varchar(20), -- Sala donde se realiza el concierto
    publico integer, -- Aforo de la sala
    otro_concierto varchar(20) -- Próximos conciertos
);

CREATE TABLE Artista
(
    id_artista integer PRIMARY KEY, -- Identificador único
    id_prox_concierto integer REFERENCES Concierto, -- Referencia a la primary key (id_concierto) de la tabla Concierto
    nombre_artista varchar(20), -- Nombre del artista o banda de música
    estilo_musica varchar(20), -- Estilo de música del artista
    fecha_creacion integer, -- Fecha en la que se formó la banda de música
    numero_disco integer, -- Número de discos publicados por el artista o banda de música
    origen varchar(20), -- Origen del artista o banda
    num_mujer integer -- Número de mujeres integrantes de cada banda de música o artistas en solitario que son mujeres
);

CREATE TABLE Concierto_Artista
(
    id_prox_concierto integer REFERENCES Concierto, -- Referencia a la primary key (id_concierto) de la tabla Concierto
    id_artista integer REFERENCES Artista, -- Referencia a la primary key (id_artista) de la tabla Artista
    PRIMARY KEY (id_prox_concierto, id_artista) -- Ambas claves foráneas se convierte en claves primarias
);

CREATE TABLE Entrada
(
    id_entrada integer PRIMARY KEY, -- Identificador único
    id_prox_concierto integer REFERENCES Concierto, -- Referencia a la primary key (id_concierto) de la tabla Concierto
    precio_entrada float(10) NOT NULL, -- Precio general de la entrada
    user_cambio varchar(20) -- Usuario que realiza el cambio de precio al ejecutar el trigger
);
```

```

CREATE TABLE Web
(
    id_web integer PRIMARY KEY,
    id_prox_concierto integer REFERENCES Concierto,
    id_artista integer REFERENCES Artista,
    enlace varchar(50),
    precio_web integer NOT NULL
);
-- Identificador único
-- Referencia a la primary key (id_concierto) de la tabla Concierto
-- Referencia a la primary key (id_artista) de la tabla Artista
-- Enlace a la página web de venta de entradas
-- Precio de la entrada en la página web

CREATE TABLE Taquilla
(
    id_taquilla integer PRIMARY KEY,
    id_prox_concierto integer REFERENCES Concierto,
    id_artista integer REFERENCES Artista,
    direccion_taquilla varchar(50),
    precio_taquilla integer NOT NULL
);
-- Identificador único
-- Referencia a la primary key (id_concierto) de la tabla Concierto
-- Referencia a la primary key (id_artista) de la tabla Artista
-- Dirección donde se encuentra la taquilla
-- Precio de la entrada en taquilla

CREATE TABLE Cliente_1
(
    DNI_cliente_1 varchar(50) PRIMARY KEY,
    id_web integer REFERENCES Web,
    id_prox_concierto integer REFERENCES Concierto,
    id_artista integer REFERENCES Artista,
    nombre_cliente1 varchar(20) NOT NULL,
    apellido varchar(20) NOT NULL,
    correo_electronico varchar(50) NOT NULL,
    cuenta_bancaria varchar(50) NOT NULL
);
-- Identificador único (DNI del comprador de entrada via web)
-- Referencia a la primary key (id_web) de la tabla Web
-- Referencia a la primary key (id_concierto) de la tabla Concierto
-- Referencia a la primary key (id_artista) de la tabla Artista
-- Nombre del comprador de la entrada via web
-- Apellido del comprador de la entrada via web
-- Correo electrónico del comprador de la entrada via web
-- Cuenta bancaria del comprador de la entrada via web

CREATE TABLE Cliente_2
(
    DNI_cliente_2 varchar(50) PRIMARY KEY,
    id_taquilla integer REFERENCES Taquilla,
    id_prox_concierto integer REFERENCES Concierto,
    id_artista integer REFERENCES Artista,
    nombre_cliente2 varchar(20) NOT NULL,
    apellido varchar(20) NOT NULL
);
-- Identificador único (DNI del comprador de entrada en taquilla)
-- Referencia a la primary key (id_taquilla) de la tabla Taquilla
-- Referencia a la primary key (id_concierto) de la tabla Concierto
-- Referencia a la primary key (id_artista) de la tabla Artista
-- Nombre del comprador de la entrada en taquilla
-- Apellido del comprador de la entrada en taquilla

CREATE TABLE entrada_old
-- Esta entidad guarda el precio de la entrada (precio_entrada) antes de que el trigger modifique el precio
(
    id_entrada integer PRIMARY KEY,
    id_prox_concierto integer REFERENCES Concierto,
    precio_entrada float(10),
    user_cambio varchar(20)
);

```



## 5. Querys

### a. Artistas procedentes de Murcia

```

SELECT nombre_ARTISTA, origen AS Ciudad_de_origen
FROM artista
WHERE origen LIKE 'Murcia';

```

|   | nombre_artista<br>character varying (20)  | ciudad_de_origen<br>character varying (20)  |
|---|--|--|
| 1 | Varry Brava  | Murcia   |
| 2 | Claim  | Murcia   |

**b. Número de discos publicados por cada artista ordenados de mayor a menor**

```
SELECT nombre_artista AS Banda_de_música, numero_disco AS
Número_de_discos

FROM artista

WHERE numero_disco>0

ORDER BY Número_de_discos DESC;
```

|    | banda_de_música<br>character varying (20) | número_de_discos<br>integer |
|----|---|-----------------------------|
| 1  | Sum41                                     | 10                          |
| 2  | Arctic Monkeys                            | 8                           |
| 3  | Simple Plan                               | 8                           |
| 4  | My Chemical Romance                       | 6                           |
| 5  | Bastille                                  | 6                           |
| 6  | Of Monsters & Men                         | 5                           |
| 7  | Varry Brava                               | 5                           |
| 8  | PVRIS                                     | 3                           |
| 9  | Maneskin                                  | 2                           |
| 10 | Claim                                     | 2                           |
| 11 | Yonaka                                    | 2                           |

**c. Añadir columna 'edad' a la tabla cliente\_1 y aplicar not null**

- Primero escribir esto:

```
ALTER TABLE cliente_1
```

```
Add Column edad varchar(10)
```

- Después, añadir que los compradores deben tener más de 18 años:

```
UPDATE cliente_1 set edad = 'más de 18'
```

- Por último:

```
ALTER TABLE cliente_1
```

```
ALTER Column edad set not null
```

- Para visualizar:

```
SELECT * from cliente_1
```

|    | dni_cliente_1<br>[PK] character varying (50) | id_web<br>integer | id_prox_concierto<br>integer | id_artista<br>integer | nombre_cliente1<br>character varying (20) | apellido<br>character varying (20) | correo_electronico<br>character varying (50) | cuenta_bancaria<br>character varying (50) | edad<br>character varying (10) |
|----|--|-------------------|------------------------------|-----------------------|---|------------------------------------|--|---|--------------------------------|
| 1  | 48623545G                                    | 1                 | 1                            | 1                     | Elena                                     | Aguayo                             | elena@gmail.com                              | ES123456789                               | más de 18                      |
| 2  | 38093345E                                    | 2                 | 2                            | 2                     | Modesto                                   | Redrejo                            | modesto@uam.es                               | ES356756789                               | más de 18                      |
| 3  | 49073345A                                    | 3                 | 3                            | 3                     | Jose                                      | Dorronsoro                         | dorronsoro@outlook....                       | ES123456854                               | más de 18                      |
| 4  | 48647854F                                    | 4                 | 4                            | 4                     | Estrella                                  | Pulido                             | estrella@gmail.com                           | ES123454790                               | más de 18                      |
| 5  | 48643900Y                                    | 5                 | 5                            | 5                     | Irene                                     | Ortin                              | ortin@gmail.com                              | ES123456432                               | más de 18                      |
| 6  | 48648965T                                    | 6                 | 6                            | 6                     | Laura                                     | Sempere                            | laura@gmail.com                              | ES123457538                               | más de 18                      |
| 7  | 48643354M                                    | 7                 | 7                            | 7                     | Victoriano                                | Mulero                             | mulero@umu.es                                | ES128966789                               | más de 18                      |
| 8  | 43343345N                                    | 8                 | 8                            | 8                     | Carlos                                    | Barba                              | carlos@live.com                              | ES124796789                               | más de 18                      |
| 9  | 48886345W                                    | 9                 | 9                            | 9                     | María                                     | Lanzarote                          | lanzarote@gmail.com                          | ES098556789                               | más de 18                      |
| 10 | 48649085Q                                    | 10                | 10                           | 10                    | Alba                                      | Baptista                           | alba@gmail.com                               | ES463256789                               | más de 18                      |
| 11 | 48643657Z                                    | 11                | 11                           | 11                    | Victoria                                  | Godoy                              | godoy@outlook.es                             | ES8876578943                              | más de 18                      |

#### d. Concierto al que acude 'Elena Pardo'

```
SELECT nombre_cliente2, apellido, nombre_artista, ciudad, sala, fecha, precio_taquilla
FROM cliente_2 NATURAL JOIN artista NATURAL JOIN concierto NATURAL JOIN taquilla
WHERE cliente_2.nombre_cliente2='Elena';
```

|   | nombre_cliente2<br>character varying (20) 🔒 | apellido<br>character varying (20) 🔒 | nombre_artista<br>character varying (20) 🔒 | ciudad<br>character varying (20) 🔒 | sala<br>character varying (20) 🔒 | fecha<br>timestamp without time zone 🔒 | precio_taquilla<br>integer 🔒 |
|---|---|--------------------------------------|--|------------------------------------|----------------------------------|--|------------------------------|
| 1 | Elena                                       | Pardo                                | Simple Plan                                | Madrid                             | Wizink Center                    | 2022-09-26 19:00:00                    | 50                           |

#### e. Enlace web donde aparece 'wizink'

```
SELECT enlace
FROM web
WHERE enlace SIMILAR TO '%(W|w)izink%';
```

|   | enlace<br>character varying (50) 🔒                        |
|---|---|
| 1 | <a href="https://www.wizink.es">https://www.wizink.es</a> |
| 2 | <a href="https://www.wizink.es">https://www.wizink.es</a> |
| 3 | <a href="https://www.wizink.es">https://www.wizink.es</a> |

#### f. Bandas de música creadas antes del año 2000

```
SELECT fecha_creacion, nombre_artista
FROM artista
WHERE fecha_creacion < 2000;
```

|   | fecha_creacion<br>integer 🔒 | nombre_artista<br>character varying (20) 🔒 |
|---|-----------------------------|--|
| 1 | 1999                        | Simple Plan                                |
| 2 | 1996                        | Sum41                                      |

#### g. Bandas de música con mujeres integrantes y el número

```
CREATE VIEW total_mujeres AS
SELECT nombre_artista, count(num_mujer)
FROM artista
WHERE num_mujer > 0
GROUP BY nombre_artista;
```

|   | nombre_artista<br>character varying (20) 🔒 | count<br>bigint 🔒 |
|---|--|-------------------|
| 1 | Of Monsters & Men                          | 1                 |
| 2 | Yonaka                                     | 1                 |
| 3 | Maneskin                                   | 1                 |
| 4 | PVRIS                                      | 1                 |



#### h. Banda de música con id=2

```
SELECT nombre_artista AS banda_de_música, ciudad AS ciudad, sala as sala_de_conciertos,  
precio_entrada AS precio_€  
  
FROM artista NATURAL INNER JOIN concierto NATURAL INNER JOIN entrada  
  
WHERE id_prox_concierto=2;
```

|   | banda_de_música<br>character varying (20) 🔒 | ciudad<br>character varying (20) 🔒 | sala_de_conciertos<br>character varying (20) 🔒 | precio_€<br>real 🔒 |
|---|---|------------------------------------|--|--------------------|
| 1 | Sum41                                       | Madrid                             | Wizink Center                                  | 45                 |

#### i. Número de conciertos en el Wizink Center

```
CREATE VIEW sala AS  
  
SELECT sala, count(*) AS número_de_conciertos  
  
FROM concierto  
  
WHERE concierto.sala='Wizink Center'  
  
GROUP BY concierto.sala;
```

|   | sala<br>character varying (20) 🔒 | número_de_conciertos<br>bigint 🔒 |
|---|----------------------------------|----------------------------------|
| 1 | Wizink Center                    | 3                                |

#### j. Número de grupos que tocan música Pop

```
CREATE VIEW pop AS  
  
SELECT estilo_musica, count(*) AS número_de_grupos  
  
FROM artista  
  
WHERE estilo_musica='Pop'  
  
GROUP BY artista.estilo_musica  
  
HAVING count(*) >0  
  
ORDER BY estilo_musica;
```

|   | estilo_musica<br>character varying (20) 🔒 | número_de_grupos<br>bigint 🔒 |
|---|---|------------------------------|
| 1 | Pop                                       | 2                            |

**k. Suma del número de mujeres en total**

```
SELECT sum (num_mujer)
FROM artista
```

|   | sum<br>bigint | 🔒 |
|---|---------------|---|
| 1 | 4             |   |

**l. Banda de música con la entrada más barata**

```
SELECT nombre_artista
FROM artista NATURAL JOIN entrada
WHERE precio_entrada IN (SELECT min(precio_entrada)
FROM entrada)
```

|   | nombre_artista<br>character varying (20) | 🔒 |
|---|--|---|
| 1 | Bastille                                 |   |

**m. Agrupar cuántos grupos tocan cada estilo de música**

```
SELECT estilo_musica, count(nombre_artista)
FROM artista
GROUP BY estilo_musica;
```

|   | estilo_musica<br>character varying (20) | count<br>bigint | 🔒 |
|---|---|-----------------|---|
| 1 | Pop Rock Electrónico                    | 1               |   |
| 2 | Indie                                   | 2               |   |
| 3 | Pop                                     | 2               |   |
| 4 | Rock                                    | 3               |   |
| 5 | Punk                                    | 3               |   |

**n. Número de conciertos en cada ciudad ordenados en sentido ascendente**

```
SELECT ciudad, count(id_prox_concierto)
FROM concierto
WHERE id_prox_concierto > 0
GROUP BY ciudad
ORDER BY count asc
```

|   | ciudad<br>character varying (20) | count<br>bigint | 🔒 |
|---|----------------------------------|-----------------|---|
| 1 | Munich                           | 1               |   |
| 2 | Murcia                           | 2               |   |
| 3 | Barcelona                        | 4               |   |
| 4 | Madrid                           | 4               |   |

#### o. Grupos de música con 2 discos publicados

```
SELECT nombre_artista, numero_disco
FROM artista
WHERE numero_disco=2
GROUP BY numero_disco,
nombre_artista;
```

|   | nombre_artista<br>character varying (20) | numero_disco<br>integer |
|---|--|-------------------------|
| 1 | Claim                                    | 2                       |
| 2 | Maneskin                                 | 2                       |
| 3 | Yonaka                                   | 2                       |

#### p. Precio de las entradas en Madrid ordenadas en sentido ascendente

```
SELECT ciudad, precio_entrada
FROM entrada, concierto
WHERE ciudad='Madrid'
GROUP BY ciudad, precio_entrada
HAVING precio_entrada > 55
ORDER BY precio_entrada ASC;
```

|   | ciudad<br>character varying (20) | precio_entrada<br>real |
|---|----------------------------------|------------------------|
| 1 | Madrid                           | 60                     |
| 2 | Madrid                           | 65                     |
| 3 | Madrid                           | 75                     |

## 6. Trigger

Este trigger consiste en actualizar el valor de un atributo (precio\_entrada) de la entidad entrada, guardando previamente el precio sin modificar en otra entidad para que quede registrado el cambio. Además, en el atributo user\_cambio se puede introducir el nombre de la persona que realiza la modificación.

Para ello hay que crear previamente la entidad donde se van a guardar los datos sin modificar, esta entidad se llama entrada\_old y tiene los mismos atributos que la entidad entrada. En el modelo E-R esta entidad se encuentra también representada y relacionada con la entidad entrada.

Los pasos para crear el trigger son los siguientes:

- 1) Se crea la función llamada change\_price() que se encarga de guardar la información ANTES de la modificación (indicado con 'BEGIN') e inserta la información en entrada\_old.

```
CREATE FUNCTION change_price() RETURNS TRIGGER
```

```
as
```

```
$$
```

```

BEGIN

INSERT INTO entrada_old
VALUES(old.id_entrada,old.id_prox_concierto,old.precio_entrada,old.user_cambio);

return new;

END

$$

Language plpgsql

```

**2) A continuación, se crea el trigger *tr\_update* que modifica la información de cada columna.**

```

CREATE TRIGGER tr_update BEFORE UPDATE ON entrada

for each row

execute procedure change_price();

```

**3) Se introduce la información a modificar: id\_entrada, id\_prox\_concierto, precio\_entrada y user\_cambio. En este caso, solo se ha modificado el precio de la entrada y el resto de los valores se han dejado igual. Además, se introduce el nombre de la persona que modifica el precio, en este caso, Elena.**

```

Update entrada set

id_entrada=11,

id_prox_concierto=11,

precio_entrada=80,

user_cambio='Elena'

WHERE precio_entrada=65;

```

**4) Visualizamos los cambios de esta manera:**

```

select * from entrada

select * from entrada_old

```

## 7. Referencias

Temario de la asignatura

Binaria,, T. [@Tecnobinaria]. (s/f). ► *[ Curso de PostgreSQL ]* ◀ *APRENDE a USAR esta BASE de DATOS desde CERO*. Youtube. Recuperado el 18 de noviembre de 2022, de <https://www.youtube.com/playlist?list=PL8gxzfBmzgex2nuVanqvxoTXTPovVSwi2>