.UBAfiuba
FACULTAD DE INGENIERÍA

# Trabajo Práctico

José Luis Cabrera
Adrián Sergio Bernardi
Kailásh Aquista

[75.15- 95.05- TA044]
Base de Datos
2025 1C
*Equipo 4*

| *Integrantes* | | |
|---|---|---|
| *Nombre* | *Padrón* | *E-Mail* |
| Trezeguet, Santiago Bautista | 110343 | strezeguet@fi.uba.ar |
| Taylor, Alan | 110034 | ataylor@fi.uba.ar |
| Polizzi, Estefano | 110075 | epolizzi@fi.uba.ar |
| Pesa, Leandro Rodrigo | 110076 | lpesa@fi.uba.ar |
| Thames Alderete, Agustina | 111287 | athames@fi.uba.ar |

# Índice

**Introducción** **2**
**Diagrama Modelo Entidad-Relación** **3**
**Pasaje a Modelo Relacional** **5**
**Archivo SQL con creación de tablas** **8**
**Archivo SQL con permisos para roles** **13**
**Archivo SQL con ejemplos de inserción** **15**
**Archivo SQL con consultas solicitadas** **24**

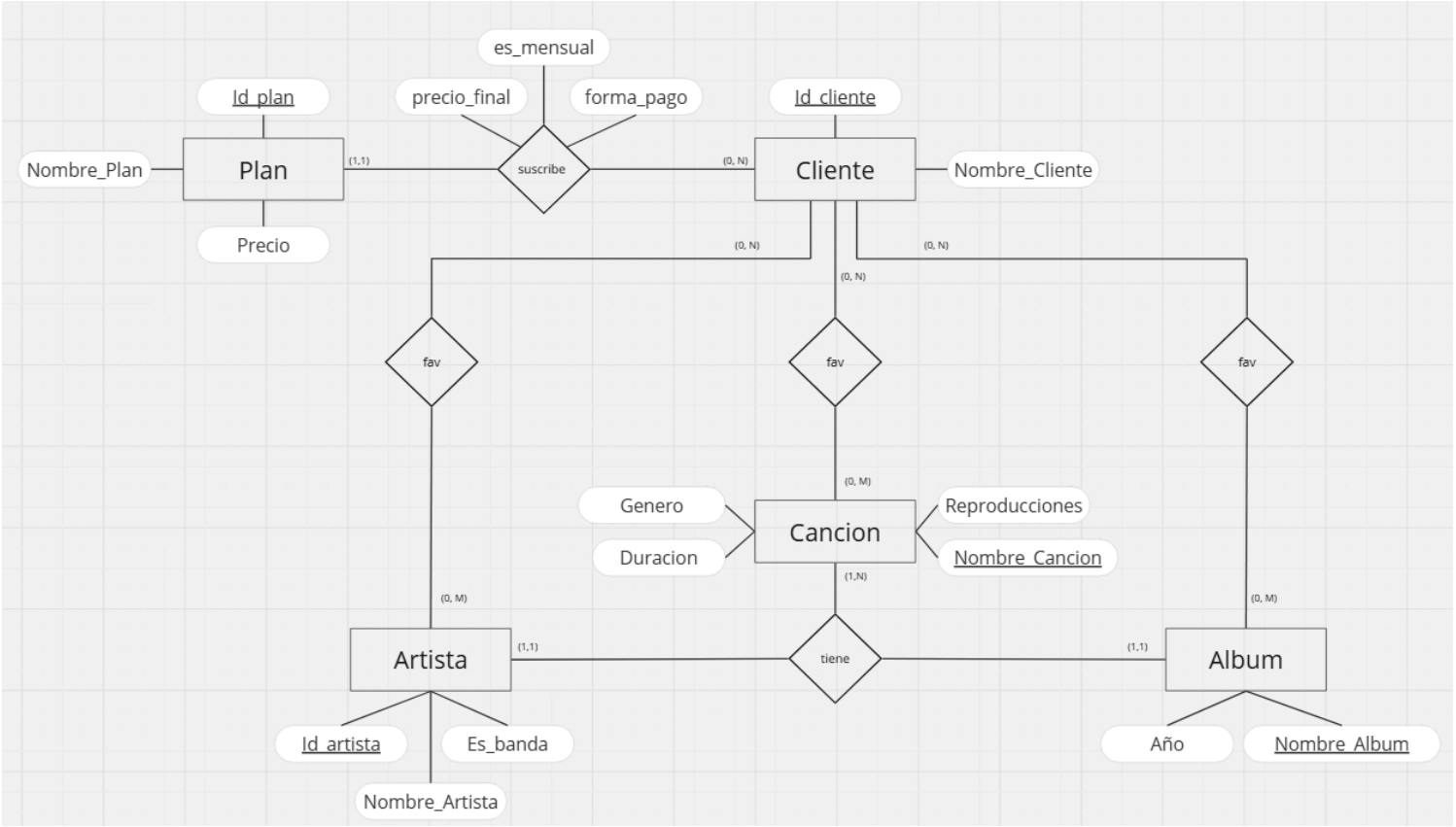
# Introducción

Este trabajo práctico tiene como objetivo diseñar un modelo de base de datos para una aplicación de música. La idea es representar y organizar la información necesaria para registrar usuarios, gestionar roles como clientes o publicadores, manejar canciones, álbumes, géneros musicales y planes de suscripción.

A lo largo del trabajo se desarrollaron las distintas partes necesarias para construir un esquema relacional que permita almacenar los datos y realizar consultas sobre ellos. También se tuvieron en cuenta restricciones planteadas, como la validación de géneros musicales o la lógica de los planes, con precio, pago o gratuito.

El enfoque del trabajo está orientado a aplicar los conceptos vistos en la materia y poder plasmarlos en una solución completa que cumpla con los requerimientos.

# Diagrama Modelo Entidad-Relación




El modelo entidad-relación propuesto fue diseñado teniendo en cuenta los requerimientos. Se consideraron tanto las entidades fundamentales del dominio como las relaciones necesarias para representar la lógica del negocio y permitir una persistencia adecuada de los datos.

**Entidades**

- **Cliente**: Representa a los usuarios consumidores de la plataforma, quienes pueden buscar, reproducir y marcar canciones, álbumes o artistas como favoritos, además de suscribirse a planes pagos u optar por el gratuito.
- **Plan**: Representa los diferentes tipos de planes disponibles (gratuito o pagos), incluyendo su precio y nombre. Se incluye la posibilidad de distinguir entre planes mensuales y anuales.
- **Artista**: Modela a los publicadores del sistema, que pueden ser bandas o solistas. Son quienes publican canciones y crean álbumes.
- **Álbum**: Entidad que agrupa canciones, y está asociado a un artista.
- **Canción**: Representa las pistas musicales individuales, incluyendo atributos como nombre, duración, género y cantidad de reproducciones.

**Relaciones**

- **Suscribe (Cliente–Plan)**: Modela las suscripciones de los clientes a los distintos planes. Se incorporaron atributos como forma_pago, precio_final y es_mensual para distinguir entre mensualidades y anualidades, y para reflejar los diferentes medios de pago (tarjeta, transferencia, etc.).
- **Fav (Cliente–Canción /Cliente–Artista /Cliente–Álbum)**: Permite registrar las entidades marcadas como favoritas por los clientes. Esta relación es clave para luego obtener estadísticas de popularidad.

- **Tiene(Artista-Canción-Álbum)**: La relación ternaria **"tiene"** entre Artista, Canción y Álbum se representa de esta manera ya que una canción es publicada por un artista y forma parte de un álbum específico al mismo tiempo. Esta relación permite modelar correctamente la asociación conjunta entre los tres elementos, cumpliendo con la exigencia de que los artistas (publicadores) puedan subir canciones dentro de álbumes. Además, evita ambigüedades que surgirían si se modelara con relaciones binarias separadas.

**Diseño y cardinalidades**

Se establecieron cardinalidades y participaciones basadas en las reglas del dominio:

- Un **cliente puede suscribirse a un único plan**, y un plan puede tener muchos suscriptores.
- Las relaciones de "favorito" permiten múltiples marcas de favoritos por cliente, pero una misma canción/artista/álbum también puede ser marcada por muchos usuarios.
- Cada **canción pertenece a un único álbum y es publicada por un único artista**, lo cual respeta las reglas impuestas por el glosario del trabajo.
- Cada **álbum** pertenece a un único artista, cumpliendo con la definición de que el artista es quien lo publica.

El modelo permite:

- Registrar usuarios con el rol de cliente y asociar suscripciones.
- Asociar canciones, álbumes y artistas como favoritos.
- Representar estadísticas de uso, como reproducciones y favoritos.
- Diferenciar entre suscripciones gratuitas y pagas, incluyendo el tratamiento de formas de pago y periodicidad.
- Restringir la creación de géneros musicales a un conjunto predefinido (se representa en el Archivo SQL).
- Asociar artistas (como publicadores) con la publicación de canciones y álbumes.

Este esquema proporciona una base sólida para construir el modelo relacional, realizar consultas SQL y garantizar una estructura coherente y escalable para una aplicación de música.

# Pasaje a Modelo Relacional

Cliente(id_cliente, nombre_cliente)
PK(id_cliente)

Plan(id_plan, nombre_plan, precio)
PK(id_plan)

Suscribe(id_plan, id_cliente, forma_pago, es_mensual, precio_final)
PK(id_cliente, id_plan)
FK: id_cliente -> Cliente(id_cliente)
FK: id_plan -> Plan(id_plan)

Artista(id_artista, nombre_artista, es_banda)
PK(id_artista)

Cancion(nombre_cancion, id_artista, nombre_album, duracion, genero, reproducciones)
PK(nombre_cancion, id_artista)
FK: id_artista -> Artista(id_artista)
FK: nombre_album, id_artista -> Album(nombre_album, id_artista)

Album(nombre_album, id_artista, año)
PK(nombre_album, id_artista)
FK: id_artista -> Artista(id_artista)

Fav_cancion(id_cliente, nombre_cancion, id_artista)
PK(id_cliente, nombre_cancion, id_artista)
FK: id_cliente -> Cliente(id_cliente)
FK: nombre_cancion, id_artista -> Cancion(nombre_cancion, id_artista)

Fav_artista(id_cliente, id_artista)
PK(id_cliente, id_artista)
FK: id_cliente -> Cliente(id_cliente)
FK: id_artista -> Artista(id_artista)

Fav_album(id_cliente, nombre_album, id_artista)
PK(id_cliente, nombre_album, id_artista)
FK: id_cliente -> Cliente(id_cliente)
Fk: nombre_album, id_artista -> Album(nombre_album, id_artista)

El pasaje del Modelo Entidad-Relación (MER) al Modelo Relacional (MR) se realizó respetando los lineamientos del curso, asegurando la correcta representación del dominio propuesto y manteniendo la integridad de los datos. Cada entidad y relación del MER fue transformada en una o más tablas del modelo relacional, incluyendo claves primarias, claves foráneas, y atributos.

**Transformación de Entidades**

Cada entidad fuerte del MER fue transformada en una relación (tabla) con sus respectivos atributos y clave primaria:

- **Cliente(id_cliente, nombre)**
  Representa a los usuarios consumidores. id_cliente es la clave primaria.

- **Plan(id_plan, nombre, precio)**
  Contiene los distintos planes disponibles. id_plan es clave primaria.

- **Artista(id_artista, nombre, es_banda)**
  Representa a los artistas (solistas o bandas). id_artista es clave primaria.

- **Álbum(nombre_album, id_artista, año)**
  Tabla dependiente del artista. La clave primaria compuesta asegura que el nombre del álbum no se repita dentro del mismo artista.

- **Canción(nombre_cancion, id_artista, nombre_album, duración, género, reproducciones)**
  Cada canción está identificada de forma única dentro del contexto del artista. Además, incluye claves foráneas a Artista y Álbum.

**Transformación de Relaciones**

- **Suscribe(id_plan, id_cliente, forma_pago, es_mensual, precio_final)**
  Relación binaria con atributos. Se convierte en una tabla con clave primaria compuesta por (id_cliente, id_plan), e incluye los atributos propios de la suscripción. Las claves foráneas garantizan la integridad con Cliente y Plan.

- **Fav_Cancion(id_cliente, nombre_cancion, id_artista)**
  Relación ternaria reducida a una tabla con clave compuesta. Vincula a clientes con canciones favoritas.

- **Fav_Artista(id_cliente, id_artista)**
  Representa a los artistas favoritos de cada cliente. Se usa clave primaria compuesta y se incorporan las claves foráneas correspondientes.

- **Fav_Album(id_cliente, nombre_album, id_artista)**
  Similar a las anteriores, pero para álbumes favoritos.

El modelo relacional cumple con los puntos indicados:

- **Roles de usuario**: Si bien el modelo relacional no explicita una tabla de roles, se infiere por el uso de las entidades Cliente y Artista como representaciones de roles diferenciados en la aplicación (clientes y publicadores).
- **Suscripciones con planes pagos y gratuitos**: La tabla Suscribe permite gestionar planes de pago con información sobre forma de pago, tipo de plan (mensual/anual), y precio final.
- **Consultas de favoritos**: Las tablas Fav_Cancion, Fav_Artista y Fav_Album permiten consultar y contar la cantidad de veces que una entidad fue marcada como favorita, facilitando estadísticas requeridas.
- **Vinculación entre canciones, álbumes y artistas**: Las claves foráneas aseguran la consistencia entre estas entidades.
- **Restricción de género**: Se prevé la validación por dominio al momento de implementación lógica (en la base de datos).
- **Formas de pago**: Incluidas como atributo de Suscribe para distinguir entre los distintos métodos aceptados.

**Claves Primarias y Foráneas**

Cada tabla tiene definida una clave primaria adecuada que permite identificar unívocamente cada fila. Se utilizaron claves compuestas cuando la entidad o relación requería múltiples atributos para garantizar unicidad (como en Canción, Álbum, o Suscribe). Las claves foráneas garantizan la integridad referencial entre las tablas.

En la siguiente tabla se representan las claves candidatas de cada entidad:

| Tabla | Clave Candidata(s) |
|---|---|
| Cliente | id_cliente |
| Plan | id_plan |
| Suscribe | (id_cliente, id_plan) |
| Artista | id_artista |
| Album | (nombre_album, id_artista) |
| Cancion | (nombre_cancion, id_artista) |
| Fav_Artista | (id_cliente, id_artista) |
| Fav_Album | (id_cliente, nombre_album, id_artista) |
| Fav_Cancion | (id_cliente, nombre_cancion, id_artista) |

# Archivo SQL con creación de tablas

```sql
-- Archivo: modelo_musica_create.sql
-- Contiene todas las definiciones de tablas y triggers
-- ========================

-- Borrado de tablas
-- DROP TABLE IF EXISTS fav_album CASCADE;
-- DROP TABLE IF EXISTS fav_artista CASCADE;
-- DROP TABLE IF EXISTS fav_cancion CASCADE;
-- DROP TABLE IF EXISTS cancion CASCADE;
-- DROP TABLE IF EXISTS album CASCADE;
-- DROP TABLE IF EXISTS artista CASCADE;
-- DROP TABLE IF EXISTS suscribe CASCADE;
-- DROP TABLE IF EXISTS plan CASCADE;
-- DROP TABLE IF EXISTS cliente CASCADE;

-- Extensión para triggers en PostgreSQL
CREATE EXTENSION IF NOT EXISTS plpgsql;

-- Tablas principales
CREATE TABLE Cliente (
    id_cliente INT PRIMARY KEY,
    nombre_cliente VARCHAR(100) NOT NULL
);

CREATE TABLE Plan (
    id_plan INT PRIMARY KEY,
    nombre_plan VARCHAR(100) CHECK (nombre_plan IN ('Gratuito',
'Pago')),
    precio DECIMAL(10,2) NOT NULL
);

CREATE TABLE Suscribe (
    id_cliente INT,
    id_plan INT,
    forma_pago VARCHAR(50) CHECK (forma_pago IN ('Ninguna', 'Debito',
'Transferencia', 'Credito')),
    es_mensual BOOLEAN,
    precio_final DECIMAL(10,2),
    PRIMARY KEY (id_cliente, id_plan),
```

```
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE
CASCADE,
    FOREIGN KEY (id_plan) REFERENCES Plan(id_plan)
);

CREATE TABLE Artista (
    id_artista INT PRIMARY KEY,
    nombre_artista VARCHAR(100) NOT NULL,
    es_banda BOOLEAN
);

CREATE TABLE Album (
    nombre_album VARCHAR(100),
    año INT,
    id_artista INT,
    PRIMARY KEY (nombre_album, id_artista),
    FOREIGN KEY (id_artista) REFERENCES Artista(id_artista)
);

CREATE TABLE Cancion (
    nombre_cancion VARCHAR(100),
    duracion INT,
    genero VARCHAR(50) CHECK (genero IN ('Rock', 'Jazz', 'Clasica',
'Latina', 'Pop', 'Electronica', 'Indie', 'Hip Hop', 'Reggae', 'Folk',
'Blues', 'Country', 'R&B', 'Gospel', 'Punk', 'Salsa', 'Techno',
'Opera', 'K-Pop', 'Funk')),
    id_artista INT,
    nombre_album VARCHAR(100),
    PRIMARY KEY (nombre_cancion, id_artista),
    FOREIGN KEY (id_artista) REFERENCES Artista(id_artista),
    FOREIGN KEY (nombre_album, id_artista) REFERENCES
Album(nombre_album, id_artista)
);

CREATE TABLE Fav_cancion (
    id_cliente INT,
    nombre_cancion VARCHAR(100),
    id_artista INT,
    PRIMARY KEY (id_cliente, nombre_cancion, id_artista),
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente),
    FOREIGN KEY (nombre_cancion, id_artista) REFERENCES
Cancion(nombre_cancion, id_artista)
);
```

```
CREATE TABLE Fav_artista (
    id_cliente INT,
    id_artista INT,
    PRIMARY KEY (id_cliente, id_artista),
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE
CASCADE,
    FOREIGN KEY (id_artista) REFERENCES Artista(id_artista)
);

CREATE TABLE Fav_album (
    id_cliente INT,
    nombre_album VARCHAR(100),
    id_artista INT,
    PRIMARY KEY (id_cliente, nombre_album, id_artista),
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE
CASCADE,
    FOREIGN KEY (nombre_album, id_artista) REFERENCES
Album(nombre_album, id_artista)
);

-- ================
-- TRIGGERS
-- ================

-- Trigger para calcular el precio_final automáticamente al insertar en
Suscribe
CREATE OR REPLACE FUNCTION before_insert_suscribe_func()
RETURNS TRIGGER AS $$
DECLARE
    plan_precio DECIMAL(10,2);
BEGIN
    SELECT precio INTO plan_precio FROM Plan WHERE id_plan =
NEW.id_plan;

    IF NOT NEW.es_mensual THEN
        NEW.precio_final := plan_precio * 12 * 0.9; -- 10% de descuento
por pago anual
    ELSE
        NEW.precio_final := plan_precio;
    END IF;

    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS before_insert_suscribe ON Suscribe;
CREATE TRIGGER before_insert_suscribe
BEFORE INSERT ON Suscribe
FOR EACH ROW
EXECUTE FUNCTION before_insert_suscribe_func();

-- Trigger para agregar la suscripción gratuita automáticamente al
insertar el cliente
CREATE OR REPLACE FUNCTION after_insert_cliente_func()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Suscribe (id_cliente, id_plan, forma_pago, es_mensual)
    VALUES (NEW.id_cliente, 0, 'Ninguna', TRUE);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS after_insert_cliente ON Cliente;
CREATE TRIGGER after_insert_cliente
AFTER INSERT ON Cliente
FOR EACH ROW
EXECUTE FUNCTION after_insert_cliente_func();

-- Trigger para eliminar la suscripción gratuita si el cliente se
suscribe a un plan pago
CREATE OR REPLACE FUNCTION after_insert_suscribe_func()
RETURNS TRIGGER AS $$
BEGIN
    -- Solo elimina el plan gratuito si se está agregando un plan
distinto al 0 (Gratuito)
    IF NEW.id_plan <> 0 THEN
        DELETE FROM Suscribe
        WHERE id_cliente = NEW.id_cliente AND id_plan = 0;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
DROP TRIGGER IF EXISTS after_insert_suscribe ON Suscribe;
CREATE TRIGGER after_insert_suscribe
AFTER INSERT ON Suscribe
FOR EACH ROW
EXECUTE FUNCTION after_insert_suscribe_func();
```

Se implementaron las siguientes funcionalidades principales:

- Se crearon las tablas base: Cliente, Plan, Suscribe, Artista, Álbum, Canción, y las relaciones de favoritos (Fav_cancion, Fav_artista, Fav_album), respetando claves primarias y foráneas.
- Se definieron restricciones para asegurar la integridad de los datos, como CHECK para valores válidos (por ejemplo, género musical o forma de pago) y relaciones referenciales entre entidades.
- Se implementaron triggers automáticos con funciones PL/pgSQL para:
    - Calcular el precio final de una suscripción dependiendo si es mensual o anual, aplicando un descuento del 10% si es anual.
    - Agregar automáticamente una suscripción gratuita al crear un nuevo cliente.
    - Eliminar automáticamente la suscripción gratuita si el cliente se suscribe a un plan pago.

Este diseño busca mantener la coherencia del modelo lógico y facilitar una gestión automática y eficiente del sistema de suscripciones y preferencias de los usuarios.

# Archivo SQL con permisos para roles

```
-- ================
-- ROLES Y PERMISOS
-- ================

-- DROP OWNED BY rol;
-- DROP ROLE rol;

-- Crear roles
CREATE ROLE rol_cliente NOINHERIT;
CREATE ROLE rol_publicador NOINHERIT;
CREATE ROLE rol_administrador SUPERUSER;

-- Crear usuarios y asignar roles
CREATE USER cliente1 WITH PASSWORD 'password';
GRANT rol_cliente TO cliente1;

CREATE USER publicador1 WITH PASSWORD 'password';
GRANT rol_publicador TO publicador1;

CREATE USER admin1 WITH PASSWORD 'password';
GRANT rol_administrador TO admin1;

-- Permisos para clientes
GRANT SELECT ON Cancion, Album, Artista, Plan, fav_cancion, fav_album,
fav_artista TO rol_cliente;
GRANT INSERT ON Fav_cancion, Fav_album, Fav_artista, Suscribe TO
rol_cliente;

-- Permisos para publicadores
GRANT SELECT, INSERT, UPDATE, DELETE ON Cancion, Album TO
rol_publicador;

-- El admin tiene todos los permisos
```

Adjuntamos el archivo SQL correspondiente en el correo. En ese archivo se definieron los **roles y permisos de acceso** a la base de datos del sistema musical previamente modelado, con el objetivo de **controlar la seguridad y el acceso según el tipo de usuario**.

Se implementaron los siguientes aspectos:

- Creación de tres roles diferenciados:
    - rol_cliente: destinado a usuarios consumidores del sistema.
    - rol_publicador: orientado a usuarios encargados de cargar y gestionar contenido musical (álbumes y canciones).
    - rol_administrador: con privilegios totales sobre la base de datos.
- Creación de usuarios de prueba (cliente1, publicador1, admin1) y asignación de los roles correspondientes.
- Asignación de permisos específicos por rol:
    - Los **clientes** pueden consultar información general (SELECT) y registrar sus preferencias (INSERT en favoritos y suscripciones).
    - Los **publicadores** tienen permisos totales sobre las tablas Canción y Álbum, incluyendo lectura, escritura, modificación y eliminación.
    - Los **administradores** heredan todos los permisos gracias al atributo SUPERUSER.

Este esquema permite separar responsabilidades, proteger la integridad de los datos y reflejar un modelo de uso realista en una aplicación multiusuario.

# Archivo SQL con ejemplos de inserción

```sql
-- Archivo: modelo_musica_inserts.sql
-- Contiene todos los inserts de datos de prueba
-- ========================

-- Inserción de planes (ID 0 = Gratuito, ID 1 = Pagos)
INSERT INTO Plan (id_plan, nombre_plan, precio) VALUES
(0, 'Gratuito', 0.00),
(1, 'Pago', 500.00);

-- Inserción de clientes (el trigger insertará automáticamente la
suscripción gratuita en Suscribe)
INSERT INTO Cliente (id_cliente, nombre_cliente) VALUES
(1, 'Juan Perez'),
(2, 'Ana Gomez'),
(3, 'Carlos Ruiz'),
(4, 'Lucia Torres'),
(5, 'Miguel Angel'),
(6, 'Sofia Lopez'),
(7, 'Pedro Martinez'),
(8, 'Laura Fernandez'),
(9, 'Javier Gomez'),
(10, 'Elena Ramirez'),
(11, 'Roberto Diaz'),
(12, 'Claudia Morales'),
(13, 'Andres Jimenez'),
(14, 'Patricia Sanchez'),
(15, 'Fernando Torres'),
(16, 'Isabel Garcia'),
(17, 'Diego Perez'),
(18, 'Mariana Lopez'),
(19, 'Victor Hugo'),
(20, 'Gabriela Martinez'),
(21, 'Raul Sanchez'),
(22, 'Carmen Diaz'),
```

```
(23, 'Alberto Jimenez'),
(24, 'Sandra Torres'),
(25, 'Ricardo Fernandez');

-- Inserción de artistas
INSERT INTO Artista (id_artista, nombre_artista, es_banda) VALUES
(1, 'Los Solistas', FALSE),
(2, 'The Band', TRUE),
(3, 'Jazz Masters', TRUE),
(4, 'Pop Star', FALSE),
(5, 'Rock Legends', TRUE),
(6, 'Electronica Beats', FALSE),
(7, 'Latina Vibes', TRUE),
(8, 'Indie Sounds', FALSE),
(9, 'Classical Harmony', TRUE),
(10, 'Hip Hop Kings', FALSE),
(11, 'Reggae Roots', TRUE),
(12, 'Folk Tales', FALSE),
(13, 'Metal Warriors', TRUE),
(14, 'Blues Brothers', FALSE),
(15, 'Country Roads', TRUE),
(16, 'R&B Soul', FALSE),
(17, 'Gospel Voices', TRUE),
(18, 'Punk Rockers', FALSE),
(19, 'Salsa Kings', TRUE),
(20, 'Techno Masters', FALSE),
(21, 'Opera Stars', TRUE),
(22, 'K-Pop Queens', FALSE),
(23, 'Latin Jazz', TRUE),
(24, 'Ambient Waves', FALSE),
(25, 'Funkadelic', TRUE);

-- Inserción de álbumes
INSERT INTO Album (nombre_album, año, id_artista) VALUES
('Solista Hits', 2022, 1),
('Band Debut', 2021, 2),
```

```
('Jazz Nights', 2020, 3),
('Pop Life', 2023, 4),
('Electronica Dreams', 2022, 6),
('Latina Fiesta', 2021, 7),
('Indie Vibes', 2020, 8),
('Classical Moments', 2019, 9),
('Hip Hop Beats', 2023, 10),
('Reggae Sunshine', 2022, 11),
('Folk Stories', 2021, 12),
('Metal Fury', 2020, 13),
('Blues Night', 2019, 14),
('Country Classics', 2023, 15),
('R&B Grooves', 2022, 16),
('Gospel Joy', 2021, 17),
('Punk Revolution', 2020, 18),
('Salsa Sensation', 2019, 19),
('Techno Vibes', 2023, 20),
('Opera Magic', 2022, 21),
('K-Pop Fever', 2021, 22),
('Latin Jazz Fusion', 2020, 23),
('Ambient Chill', 2019, 24),
('Funk Groove', 2023, 25);

-- Inserción de canciones
INSERT INTO Cancion (nombre_cancion, duracion, genero, id_artista,
nombre_album) VALUES
('Caminos', 210, 'Rock', 1, 'Solista Hits'),
('Luz del Alba', 195, 'Pop', 1, 'Solista Hits'),
('Despertar', 180, 'Rock', 2, 'Band Debut'),
('Nocturno', 200, 'Jazz', 3, 'Jazz Nights'),
('Electro Party', 220, 'Electronica', 4, 'Pop Life'),
('Sentimiento', 215, 'Jazz', 3, 'Jazz Nights'),
('Bailando', 205, 'Latina', 4, 'Pop Life'),
('Fiesta', 230, 'Latina', 7, 'Latina Fiesta'),
('Indie Love', 240, 'Indie', 8, 'Indie Vibes'),
('Classical Dream', 250, 'Clasica', 9, 'Classical Moments'),
```

```
('Hip Hop Flow', 260, 'Hip Hop', 10, 'Hip Hop Beats'),
('Reggae Vibes', 270, 'Reggae', 11, 'Reggae Sunshine'),
('Folk Heart', 280, 'Hip Hop', 12, 'Folk Stories'),
('Metal Storm', 290, 'Rock', 13, 'Metal Fury'),
('Blues Soul', 300, 'Blues', 14, 'Blues Night'),
('Country Roads', 310, 'Country', 15, 'Country Classics'),
('R&B Love', 320, 'R&B', 16, 'R&B Grooves'),
('Gospel Light', 330, 'Gospel', 17, 'Gospel Joy'),
('Punk Anthem', 340, 'Punk', 18, 'Punk Revolution'),
('Salsa Beat', 350, 'Salsa', 19, 'Salsa Sensation'),
('Techno Pulse', 360, 'Electronica', 20, 'Techno Vibes'),
('Opera Aria', 370, 'Opera', 21, 'Opera Magic'),
('K-Pop Dance', 380, 'K-Pop', 22, 'K-Pop Fever'),
('Latin Jazz Groove', 390, 'Jazz', 23, 'Latin Jazz Fusion'),
('Ambient Soundscape', 400, 'Clasica', 24, 'Ambient Chill'),
('Funk Jam', 410, 'Funk', 25, 'Funk Groove'),
('Electronic Anthem', 420, 'Rock', 6, 'Electronica Dreams'),
('Jazz Fusion', 430, 'Jazz', 3, 'Jazz Nights'),
('Pop Anthem', 440, 'Pop', 4, 'Pop Life'),
('Electronica Beat', 450, 'Electronica', 6, 'Electronica Dreams'),
('Latina Groove', 460, 'Latina', 7, 'Latina Fiesta'),
('Indie Anthem', 470, 'Indie', 8, 'Indie Vibes'),
('Classical Symphony', 480, 'Clasica', 9, 'Classical Moments'),
('Hip Hop Anthem', 490, 'Hip Hop', 10, 'Hip Hop Beats'),
('Reggae Anthem', 500, 'Reggae', 11, 'Reggae Sunshine'),
('Folk Anthem', 510, 'Folk', 12, 'Folk Stories'),
('Metal Anthem', 520, 'Rock', 13, 'Metal Fury'),
('Blues Anthem', 530, 'Blues', 14, 'Blues Night'),
('Country Anthem', 540, 'Country', 15, 'Country Classics'),
('R&B Anthem', 550, 'R&B', 16, 'R&B Grooves'),
('Gospel Anthem', 560, 'Gospel', 17, 'Gospel Joy'),
('Punk Anthem II', 570, 'Punk', 18, 'Punk Revolution'),
('Salsa Anthem II', 580, 'Salsa', 19, 'Salsa Sensation'),
('Techno Anthem II', 590, 'Electronica', 20, 'Techno Vibes'),
('Opera Anthem II', 600, 'Opera', 21, 'Opera Magic'),
('K-Pop Anthem II', 610, 'K-Pop', 22, 'K-Pop Fever'),
```

```
('Latin Jazz Anthem II', 620, 'Jazz', 23, 'Latin Jazz Fusion'),
('Ambient Anthem II', 630, 'Clasica', 24, 'Ambient Chill'),
('Funkadelic Anthem II', 640, 'Electronica', 25, 'Funk Groove'),
('Caminos Remix', 215, 'Rock', 1, 'Solista Hits'),
('Luz del Alba Acoustic', 190, 'Pop', 1, 'Solista Hits'),
('Despertar Acoustic', 185, 'Rock', 2, 'Band Debut'),
('Nocturno Jazz', 205, 'Jazz', 3, 'Jazz Nights'),
('Electro Party Remix', 225, 'Electronica', 4, 'Pop Life'),
('Sentimiento Live', 220, 'Jazz', 3, 'Jazz Nights'),
('Bailando Remix', 210, 'Latina', 4, 'Pop Life'),
('Fiesta Remix', 235, 'Latina', 7, 'Latina Fiesta'),
('Indie Love Acoustic', 245, 'Indie', 8, 'Indie Vibes'),
('Classical Dream Live', 255, 'Clasica', 9, 'Classical Moments'),
('Hip Hop Flow Remix', 265, 'Hip Hop', 10, 'Hip Hop Beats'),
('Reggae Vibes Live', 275, 'Reggae', 11, 'Reggae Sunshine'),
('Folk Heart Acoustic', 285, 'Folk', 12, 'Folk Stories'),
('Metal Storm Live', 295, 'Rock', 13, 'Metal Fury'),
('Blues Soul Acoustic', 305, 'Blues', 14, 'Blues Night'),
('Country Roads Remix', 315, 'Country', 15, 'Country Classics'),
('R&B Love Live', 325, 'R&B', 16, 'R&B Grooves'),
('Gospel Light Remix', 335, 'Gospel', 17, 'Gospel Joy'),
('Punk Anthem Live', 345, 'Punk', 18, 'Punk Revolution'),
('Salsa Beat Remix', 355, 'Salsa', 19, 'Salsa Sensation'),
('Techno Pulse Remix', 365, 'Electronica', 20, 'Techno Vibes'),
('Opera Aria Live', 375, 'Opera', 21, 'Opera Magic'),
('K-Pop Dance Remix', 385, 'K-Pop', 22, 'K-Pop Fever'),
('Latin Jazz Groove Live', 395, 'Jazz', 23, 'Latin Jazz Fusion'),
('Ambient Soundscape Live', 405, 'Clasica', 24, 'Ambient Chill');

-- Inserción de suscripciones pagas (esto eliminará la suscripción
gratuita por trigger)
INSERT INTO Suscribe (id_cliente, id_plan, forma_pago, es_mensual)
VALUES
(1, 1, 'Debito', TRUE),
(2, 1, 'Credito', FALSE),
(3, 1, 'Credito', TRUE),
```

```
(4, 1, 'Debito', FALSE),
(5, 1, 'Transferencia', TRUE),
(6, 1, 'Debito', FALSE),
(7, 1, 'Credito', TRUE),
(8, 1, 'Debito', FALSE),
(9, 1, 'Transferencia', TRUE),
(10, 1, 'Transferencia', FALSE);

-- Inserción de favoritos de canciones
INSERT INTO Fav_cancion (id_cliente, nombre_cancion, id_artista) VALUES
(1, 'Caminos', 1),
(2, 'Luz del Alba', 1),
(3, 'Despertar', 2),
(4, 'Nocturno', 3),
(5, 'Electro Party', 4),
(6, 'Sentimiento', 3),
(7, 'Bailando', 4),
(8, 'Fiesta', 7),
(9, 'Indie Love', 8),
(10, 'Classical Dream', 9),
(11, 'Hip Hop Flow', 10),
(12, 'Reggae Vibes', 11),
(13, 'Folk Heart', 12),
(14, 'Metal Storm', 13),
(15, 'Blues Soul', 14),
(16, 'Country Roads', 15),
(17, 'R&B Love', 16),
(18, 'Gospel Light', 17),
(19, 'Punk Anthem', 18),
(20, 'Salsa Beat', 19),
(21, 'Techno Pulse', 20),
(22, 'Opera Aria', 21),
(23, 'K-Pop Dance', 22),
(2, 'Luz del Alba Acoustic', 1),
(3, 'Despertar Acoustic', 2),
(4, 'Nocturno Jazz', 3),
```

```
(5, 'Electro Party Remix', 4),
(8, 'Fiesta Remix', 7),
(9, 'Indie Love Acoustic', 8),
(14, 'Metal Storm Live', 13),
(15, 'Blues Soul Acoustic', 14),
(16, 'Country Roads Remix', 15),
(19, 'Punk Anthem Live', 18),
(21, 'Techno Pulse Remix', 20),
(22, 'Opera Aria Live', 21),
(23, 'K-Pop Dance Remix', 22),
(1, 'Latin Jazz Groove', 23),
(2, 'Ambient Soundscape', 24),
(3, 'Funk Jam', 25),
(1, 'Nocturno', 3),
(2, 'Bailando', 4),
(3, 'Sentimiento', 3),
(4, 'Electro Party', 4),
(4, 'Luz del Alba', 1),
(5, 'Fiesta', 7),
(6, 'Indie Love', 8),
(7, 'Classical Dream', 9),
(8, 'Hip Hop Flow', 10),
(9, 'Reggae Vibes', 11),
(10, 'Folk Heart', 12),
(11, 'Metal Storm', 13),
(12, 'Blues Soul', 14),
(13, 'Country Roads', 15),
(14, 'R&B Love', 16),
(15, 'Gospel Light', 17),
(16, 'Punk Anthem', 18),
(17, 'Salsa Beat', 19),
(18, 'Techno Pulse', 20),
(19, 'Opera Aria', 21),
(20, 'K-Pop Dance', 22),
(1, 'Caminos Remix', 1),
(6, 'Sentimiento Live', 3),
```

```
(7, 'Bailando Remix', 4),
(10, 'Classical Dream Live', 9),
(11, 'Hip Hop Flow Remix', 10),
(12, 'Reggae Vibes Live', 11),
(13, 'Folk Heart Acoustic', 12),
(17, 'R&B Love Live', 16),
(18, 'Gospel Light Remix', 17),
(20, 'Salsa Beat Remix', 19);

-- Inserción de favoritos de artistas
INSERT INTO Fav_artista (id_cliente, id_artista) VALUES
(1, 3),
(2, 4),
(3, 2),
(4, 1),
(5, 1),
(6, 2);

-- Inserción de favoritos de álbumes
INSERT INTO Fav_album (id_cliente, nombre_album, id_artista) VALUES
(1, 'Jazz Nights', 3),
(2, 'Pop Life', 4),
(3, 'Band Debut', 2),
(4, 'Solista Hits', 1),
(5, 'Electronica Dreams', 6),
(6, 'Latina Fiesta', 7),
(7, 'Indie Vibes', 8),
(8, 'Classical Moments', 9),
(9, 'Hip Hop Beats', 10),
(10, 'Reggae Sunshine', 11),
(11, 'Folk Stories', 12),
(12, 'Metal Fury', 13),
(13, 'Blues Night', 14),
(14, 'Country Classics', 15),
(15, 'R&B Grooves', 16),
(16, 'Gospel Joy', 17),
```

```
(17, 'Punk Revolution', 18),
(18, 'Salsa Sensation', 19),
(19, 'Techno Vibes', 20),
(20, 'Opera Magic', 21),
(1, 'K-Pop Fever', 22),
(2, 'Latin Jazz Fusion', 23),
(3, 'Ambient Chill', 24),
(4, 'Funk Groove', 25),
(5, 'Jazz Nights', 3),
(6, 'Pop Life', 4),
(7, 'Band Debut', 2),
(8, 'Solista Hits', 1),
(9, 'Electronica Dreams', 6),
(10, 'Latina Fiesta', 7);
```

En este archivo se insertan datos de prueba para poblar la base de datos y verificar el correcto funcionamiento del sistema. Se incluyen:

- **Planes:** Se crean los planes gratuito y pago para simular distintos tipos de suscripciones.
- **Clientes:** Se insertan múltiples clientes, activando el trigger que los suscribe automáticamente al plan gratuito.
- **Artistas, álbumes y canciones:** Se agregan registros variados para representar un catálogo musical amplio y coherente.
- **Suscripciones pagas:** Se insertan manualmente para activar el trigger que elimina la suscripción gratuita cuando corresponde.
- **Favoritos:** Se cargan favoritos de canciones, artistas y álbumes para probar relaciones N:M entre clientes y contenido musical.

Estas inserciones permiten testear la integridad referencial, los triggers y el comportamiento esperado del sistema ante distintos escenarios.

# Archivo SQL con consultas

```sql
-- Listar todas las canciones de la aplicacion
SELECT * FROM Cancion;

-- Listar todos los artistas de la plataforma
SELECT * FROM Artista;

-- Listar todos los albumes, agrupados por artistas
SELECT a.nombre_artista, ARRAY_AGG(al.nombre_album) AS albums
FROM Album al
JOIN Artista a ON al.id_artista = a.id_artista
GROUP BY a.nombre_artista
ORDER BY a.nombre_artista;

-- Publicar canciones
INSERT INTO Cancion (nombre_cancion, duracion, genero, id_artista,
nombre_album)
VALUES ('Nueva Melodia', 210, 'Rock', 1, 'Solista Hits');

-- Publicar artistas
INSERT INTO Artista (id_artista, nombre_artista, es_banda)
VALUES (26, 'Los Melodicos', TRUE);

-- Publicar un album
INSERT INTO Album (nombre_album, año, id_artista)
VALUES ('Album Debut', 2025, 2);

-- Actualizar una cancion
UPDATE Cancion
SET duracion = 215
WHERE nombre_cancion = 'Nueva Melodia' AND id_artista = 1;
```

```sql
-- Actualizar un artista
UPDATE Artista
SET nombre_artista = 'Los Melodicos Internacionales'
WHERE id_artista = 26;

-- Actualizar un album
UPDATE Album
SET año = 2026
WHERE nombre_album = 'Album Debut' AND id_artista = 2;

-- Registrar usuarios:
    -- Registrar un cliente (admin)
    INSERT INTO Cliente (id_cliente, nombre_cliente)
    VALUES (100, 'Santiago Pesa');

    -- Registrar un publicador/artista (admin)
    INSERT INTO Artista (id_artista, nombre_artista, es_banda)
    VALUES (100, 'Solista Nuevo', FALSE);

-- Realizar el pago de un plan de suscripcion (usuario con rol_cliente)
INSERT INTO Suscribe (id_cliente, id_plan, forma_pago, es_mensual)
VALUES (100, 1, 'Credito', TRUE);

-- Desregistrar a ususarios:
    -- Desregistrar a un cliente de la aplicacion (admin)
    DELETE FROM Cliente WHERE id_cliente = 100;

    -- Desregistrar a un artista de la aplicacion (admin)
    DELETE FROM Artista WHERE id_artista =100;

-- Mostrar las canciones mas populares ordenadas por cantidad de
favoritos
SELECT fc.nombre_cancion, COUNT(fc.id_cliente) AS cantidad_favoritos
FROM fav_cancion fc
GROUP BY fc.nombre_cancion
ORDER BY cantidad_favoritos DESC;
```

```sql
-- Mostrar los artistas mas populares basandose en la cantidad de
canciones favoritas
SELECT a.nombre_artista, COUNT(fc.id_cliente) AS cantidad_favoritos
FROM Fav_cancion fc
JOIN Artista a ON fc.id_artista = a.id_artista
GROUP BY a.nombre_artista
ORDER BY cantidad_favoritos DESC;
```

En este archivo se desarrollan las consultas solicitadas, entre las cuales se utilizan los métodos vistos en la materia, tales como: SELECT, WHERE, GROUP BY, UPDATE, DELETE, INSERT, etc; de manera tal que se muestren los resultados correctos o se realicen las modificaciones necesarias.