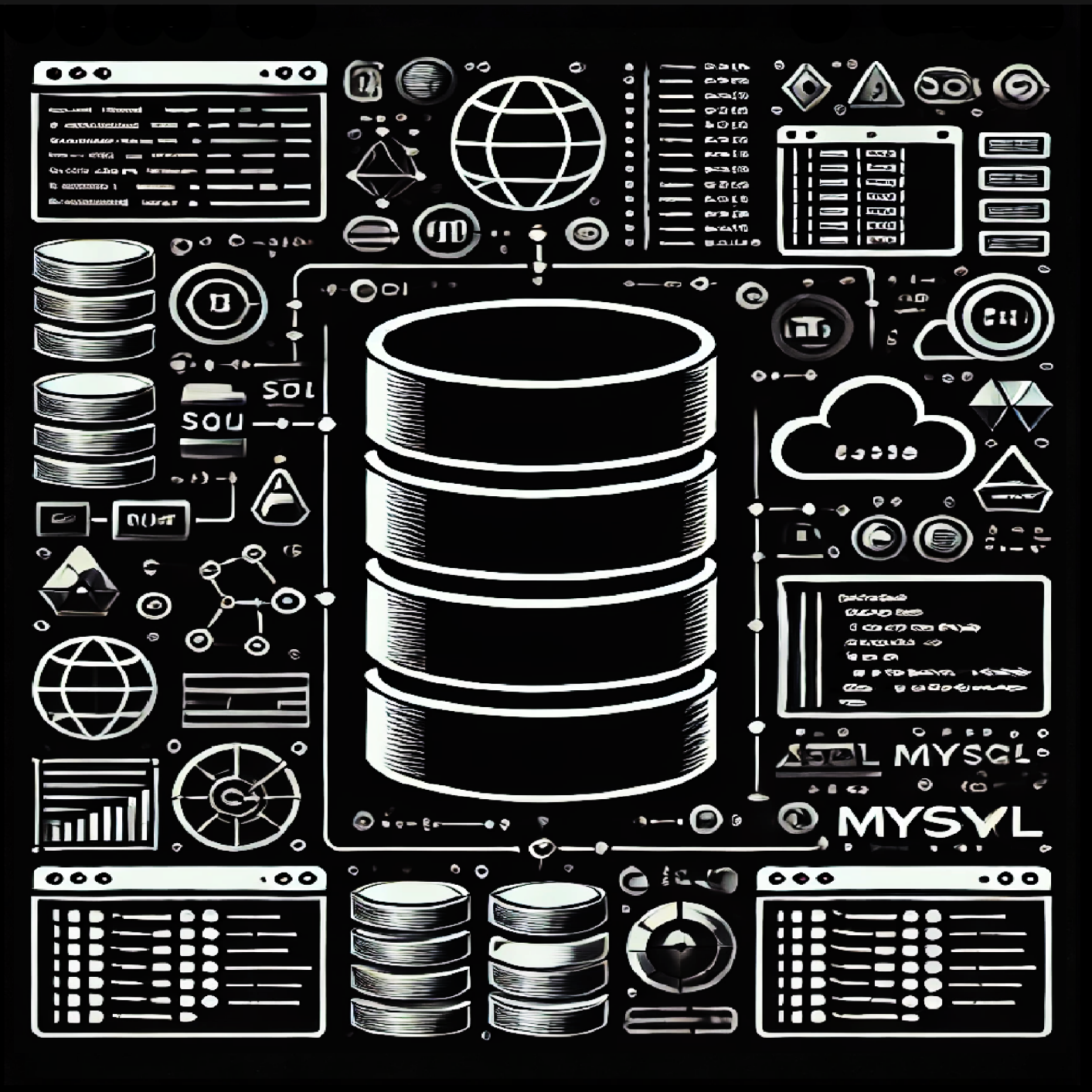
Creación de una Base de Datos

Práctica de el módulo de Bases de Datos SQL

Laura Rodríguez Ropero



Máster en Big Data, Data Science e IA

Creación de una Base de Datos

Práctica de el módulo de Bases de Datos SQL

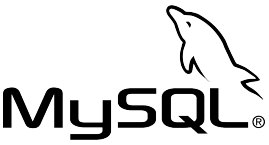
por

Laura Rodríguez Ropero

10/10/2024

Índice

Profesora: Isabel Riomoros
Facultad: Facultad de Estudios Estadísticos
Universidad: Universidad Complutense de Madrid
Máster: Big Data, Data Science e Inteligencia Artificial



1

Introduction

El proyecto tiene como objetivo principal el análisis, diseño e implementación de una base de datos relacional para la empresa “ArteVida Cultural”, dedicada a la organización de eventos culturales. La base de datos permitirá gestionar de manera eficiente todos los aspectos relacionados con los eventos, como la variedad de actividades, los artistas participantes, las ubicaciones, la venta de entradas, la asistencia del público y las valoraciones de los asistentes.

2

Diseño Conceptual

En esta sección se desarrolla el modelo conceptual de datos para la empresa “ArteVida Cultural”, a partir de los requisitos previamente definidos. Este modelo conceptual se presenta a través de un diagrama entidad-relación (E-R), recogido en la figura 2.1, que permite representar las entidades clave involucradas en la organización de eventos culturales y las relaciones entre ellas.

En primer lugar vamos a definir nuestras entidades:

- **Evento:** Es la entidad principal que define la razón de ser de la empresa, ya que la organización de eventos es su propósito. Cada evento tiene un nombre y una fecha y hora para diferenciarlo, así como una descripción breve.
  Claves: *id_evento* como clave primaria porque cada evento debe tener un identificador único.
- **Actividad:** Representa lo que ocurre dentro de cada evento, y se clasifican según su tipo (música, teatro, etc.). Las actividades tienen un costo variable según los artistas que participan en ellas, pues lo calcularemos sumando el caché de los artistas respectivos (atributo derivado).
  Claves: *id_actividad* como clave primaria.
- **Artista:** Los artistas son cruciales para definir el contenido de las actividades. Sus biografías añaden valor a la descripción del evento. El caché del artista no es un atributo fijo porque varía según la actividad.
  Claves: *id_artista* como clave primaria.
- **Ubicación:** Cada evento se organiza en una ubicación física, que tiene su capacidad o aforo y características específicas.
  Claves: *id_ubicacion* como clave primaria.
- **Asistente:** Los asistentes representan al público que compra las entradas para los eventos, y pueden dejar valoraciones. De cada uno de ellos sabemos el nombre completo (atributo compuesto), el email y el teléfono (atributo multivalorado).
  Claves: *id_asistente* como clave primaria.
- **Categoria:** La entidad Categoría es necesaria para gestionar la venta y el control de las entradas para cada evento. Así se puede manejar el precio de cada entrada, permitiendo múltiples precios por evento, de cada categoría (por ejemplo, dependiendo de si se trata de una entrada VIP, general, reducida, etc).
  Claves: *id_categoria* como clave primaria.
- **Entrada:** La entidad Entrada va a relacionar las entidades Evento, Asistente y Categoría permitiendo hacer un registro de todas las entradas vendidas.
  Claves: *id_entrada* como clave primaria.
  Restricciones: Se debe controlar que la cantidad de entradas vendidas para cada evento no supere el aforo delimitado por la ubicación.

Ahora vamos a proceder a describir las relaciones entre ellas:

- **Evento-Actividad:** Un evento está asociado a una única actividad (como un concierto, exposición, obra de teatro, etc.), pero una actividad puede realizarse en varios eventos diferentes.
  Cardinalidad: N:1.
- **Actividad-Artista (participe):** Una actividad puede incluir a varios artistas, y un artista puede participar en varias actividades.
  Cardinalidad: N:N.
  Aspecto relevante: Esta relación permite registrar el caché específico de cada artista para cada actividad como atributo.
- **Evento-Ubicación:** Cada evento se realiza en una única ubicación, pero una ubicación puede albergar varios eventos.
  Cardinalidad: N:1.
- **Evento-Asistente (valore):** Un asistente puede valorar varios eventos a los que asistió, y cada evento puede recibir valoraciones de varios asistentes.
  Cardinalidad: N:N.
  Aspecto relevante: La valoración se registra como un atributo de esta relación, con una puntuación del 0 al 5.
- **Evento-Categoria (valga):** Un evento puede vender entradas de diferentes categorías, y una categoría de entrada puede estar disponible para más de un evento.
  Cardinalidad: N:N.
  Aspecto relevante: En esta tabla se van a almacenar los precios de entradas de todas las combinaciones de categorias y eventos.

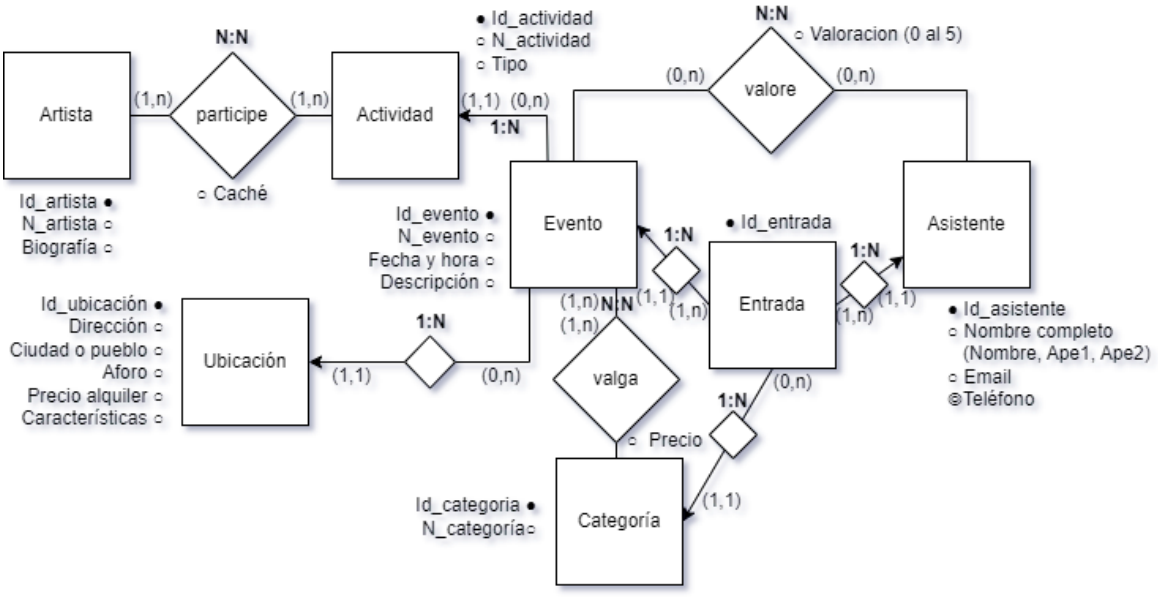


Figure 2.1: Modelo Entidad-Relación

3

Diseño Lógico

En esta tercera fase del proyecto, nos centramos en aplicar las técnicas vistas en el curso para llevar a cabo el proceso de transformación del modelo conceptual, representado a través del diagrama entidad-relación (ER), al modelo relacional. Este proceso, conocido como el ”paso a tablas”, nos permite convertir las entidades, atributos y relaciones del modelo ER en un conjunto de tablas normalizadas, que constituirán la base de datos relacional. A continuación muestro un esquema de ello:

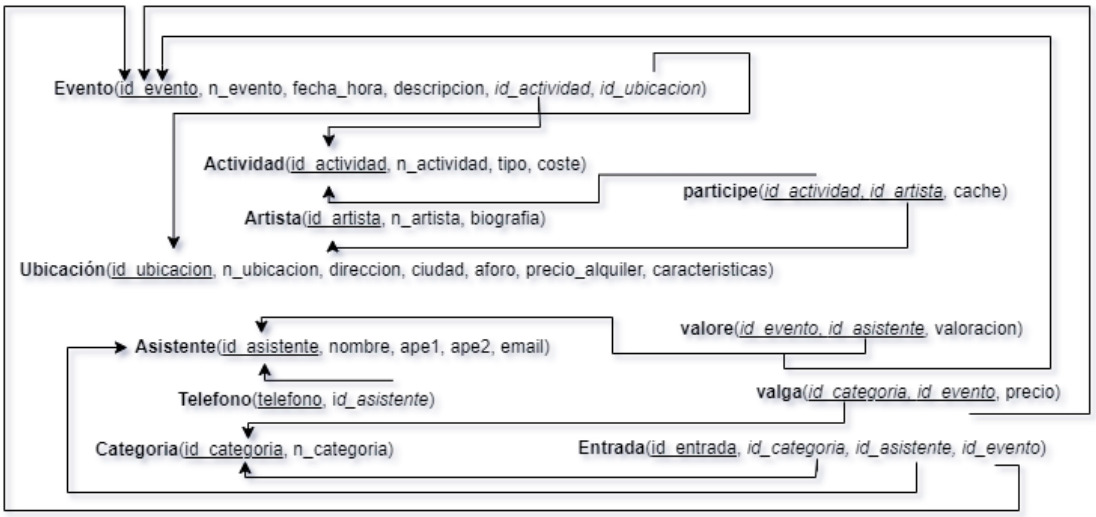


Figure 3.1: Modelo Relacional

Procedo a detallar las principales tablas y sus atributos.

- **Ubicacion**
  - *Atributos*:
    - $id\_ubicacion$ (PK): Identificador único de la ubicación.
    - $n\_ubicacion$: Nombre de la ubicación.
    - $direccion$: Dirección completa de la ubicación.
    - $ciudad$: Ciudad o pueblo de la ubicación.
    - $aforo$: Aforo máximo de la ubicación.
    - $precio\_alquiler$: Precio de alquilar el recinto de la ubicación.
    - $caracteristicas$: Breve mención de las características principales de la ubicación.

- **Actividad**
  - *Atributos*:
    - $id\_actividad$ (PK): Identificador único de la actividad.
    - $n\_actividad$: Nombre de la actividad.
    - $tipo$: Tipo de la actividad.
    - $coste$: Coste de la actividad (suma de los cachés de los artistas que participan).
- **Artista**
  - *Atributos*:
    - $id\_artista$ (PK): Identificador único del artista.
    - $n\_artista$: Nombre del artista.
    - $biografa$: Breve descripción de la vida y logros del artista.
- **Evento**
  - *Atributos*:
    - id_evento (PK): Identificador único del evento.
    - n_evento: Nombre del evento.
    - $fecha\_hora$: Fecha y hora del evento.
    - $descripcion$: Breve descripción del evento.
    - $id\_ubicacion$ (FK): Identificador de la ubicación donde se realiza el evento.
    - $id\_actividad$ (FK): Identificador de la actividad que se realiza el evento.
- **Asistente**
  - *Atributos*:
    - $id\_asistente$ (PK): Identificador único del asistente.
    - $nombre$: Nombre del asistente.
    - $ape1, ape2$: Primer y segundo apellido del asistente.
    - $email$: Correo electrónico del asistente.
- **Teléfono**
  - *Atributos*:
    - $telefono$ (PK): Dígitos que componen el nº de teléfono.
    - $id\_asistente$ (FK): Identificador del asistente al que pertenece el teléfono.
- **Categoría**
  - *Atributos*:
    - $id\_categoria$ (PK): Identificador único de la categoría de la entrada.
    - $n\_categoria$: Nombre de la categoría de la entrada.
- **Entrada**
  - *Atributos*:
    - $id\_entrada$ (PK): Identificador único de la entrada.
    - id_evento (FK): Identificador del evento para el cual vale la entrada.
    - $id\_asistente$ (FK): Identificador del asistente que ha comprado la entrada.
    - $id\_categoria$ (FK): Identificador de la categoría de la entrada.

Ahora vamos a detallar las tablas fruto de las relaciones entre las entidades:

- **participe**
  - Descripción: Esta tabla intermedia gestiona la relación entre **Actividad** y **Artista**, permitiendo que un artista participe en múltiples actividades y una actividad tenga múltiples artistas.
  - Atributos:
    - $id\_actividad$ (FK): Identificador de la actividad.
    - $id\_artista$ (FK): Identificador del artista.
    - $cache$: Honorarios o cache que se pagará al artista por su participación en la actividad.
  - Claves:
    - *PRIMARY KEY* ($id\_actividad$, $id\_artista$): Combinación única de actividad y artista.
    - *UNIQUE* ($id\_actividad$, $id\_artista$): Garantiza que no haya duplicados en la relación.
- **valore**
  - Descripción: Esta tabla intermedia gestiona la relación entre **Evento** y **Asistente**, permitiendo a los asistentes valorar los eventos a los que asisten.
  - Atributos:
    - id_evento (FK): Identificador del evento.
    - $id\_asistente$ (FK): Identificador del asistente.
    - $valoracion$: Valoración del evento, que debe estar entre 0 y 5.
  - Claves:
    - *PRIMARY KEY* (id_evento, $id\_asistente$): Combinación única de evento y asistente.
    - *UNIQUE* (id_evento, $id\_asistente$): Garantiza que no haya duplicados en la relación.
  - Restricciones:
    - *CHECK* (valoracion $\geq$ 0 **AND** valoracion $\leq$ 5): Asegura que la valoración esté en el rango permitido.
- **valga**
  - Descripción: Esta tabla intermedia gestiona la relación entre **Evento** y **Categoria**, permitiendo que un evento esté asociado a múltiples categorías con un precio específico para cada asociación.
  - Atributos:
    - id_evento (FK): Identificador del evento.
    - $id\_categoria$ (FK): Identificador de la categoría.
    - $precio$: Precio asociado a la categoría del evento.
  - Claves:
    - *PRIMARY KEY* (id_evento, $id\_categoria$): Combinación única de evento y categoría.
    - *UNIQUE* (id_evento, $id\_categoria$): Garantiza que no haya duplicados en la relación.

También hay que tener en consideración las posibles restricciones en nuestra base de datos.

- **Aforo**:
  - Para asegurar que el número total de entradas vendidas para un evento no exceda el aforo permitido de su ubicación. Esto se gestiona a través de un trigger en la base de datos que verifica que el total de entradas +1 no sea mayor que aforo antes de permitir una nueva inserción en la tabla de entradas.

4

Implementación

En esta sección se abordará la implementación de la base de datos utilizando MySQL, un sistema de gestión de bases de datos relacional fundamental para la organización y manejo eficiente de datos. El script desarrollado es el siguiente:

```sql
/* ------------------------------------------------------------------------------------------
Creación de la Base de Datos ArteVidaCultural
------------------------------------------------------------------------------------------*/
DROP DATABASE IF EXISTS ArteVidaCultural;
CREATE DATABASE IF NOT EXISTS ArteVidaCultural;
USE ArteVidaCultural;

/* ------------------------------------------------------------------------------------------
Definición de la estructura de la Base de Datos
Creación de las tablas
------------------------------------------------------------------------------------------*/

-- Tabla Ubicacion
CREATE TABLE Ubicacion (
    id_ubicacion INT AUTO_INCREMENT PRIMARY KEY,
    n_ubicacion VARCHAR(100) NOT NULL,
    direccion VARCHAR(255),
    ciudad VARCHAR(100),
    aforo INT NOT NULL,
    precio_alquiler DECIMAL(10, 2),
    caracteristicas VARCHAR(255)
);

-- Tabla Actividad
CREATE TABLE Actividad (
    id_actividad INT AUTO_INCREMENT PRIMARY KEY,
    n_actividad VARCHAR(100) NOT NULL,
    tipo VARCHAR(50),
    coste DECIMAL(10, 2)
);

-- Tabla Evento
CREATE TABLE Evento (
    id_evento INT AUTO_INCREMENT PRIMARY KEY,
    n_evento VARCHAR(100) NOT NULL,
    fecha_hora DATETIME NOT NULL,
    descripcion TEXT,
    id_actividad INT,
    id_ubicacion INT,
    UNIQUE (id_actividad, id_ubicacion),
    FOREIGN KEY (id_actividad) REFERENCES Actividad(id_actividad),
    FOREIGN KEY (id_ubicacion) REFERENCES Ubicacion(id_ubicacion)
);
```

```sql
-- Tabla Artista
CREATE TABLE Artista (
    id_artista INT AUTO_INCREMENT PRIMARY KEY,
    n_artista VARCHAR(150) NOT NULL,
    biografia TEXT
);

-- Tabla Asistente
CREATE TABLE Asistente (
    id_asistente INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    ape1 VARCHAR(50) NOT NULL,
    ape2 VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);

-- Tabla Telefono
CREATE TABLE Telefono (
    telefono NUMERIC(9,0) NOT NULL PRIMARY KEY,
    id_asistente INT NOT NULL,
    FOREIGN KEY (id_asistente) REFERENCES Asistente(id_asistente)
);

-- Tabla Categoria
CREATE TABLE Categoria (
    id_categoria INT AUTO_INCREMENT PRIMARY KEY,
    n_categoria VARCHAR(50) NOT NULL
);

-- Tabla intermedia para la relación Actividad-Artista (participe) (N:N)
CREATE TABLE Participe (
    id_actividad INT,
    id_artista INT,
    cache DECIMAL(10, 2),
    PRIMARY KEY (id_actividad, id_artista),
    UNIQUE (id_actividad, id_artista),
    FOREIGN KEY (id_actividad) REFERENCES Actividad(id_actividad),
    FOREIGN KEY (id_artista) REFERENCES Artista(id_artista)
);

-- Tabla intermedia para la relación Evento-Asistente (valore) (N:N) con valoración
CREATE TABLE Valore (
    id_evento INT,
    id_asistente INT,
    valoracion SMALLINT CHECK (valoracion >= 0 AND valoracion <= 5),
    PRIMARY KEY (id_evento, id_asistente),
    UNIQUE (id_evento, id_asistente),
    FOREIGN KEY (id_evento) REFERENCES Evento(id_evento),
    FOREIGN KEY (id_asistente) REFERENCES Asistente(id_asistente)
);

-- Tabla intermedia para la relación Evento-Categoria (valga) (N:N) con precio
CREATE TABLE Valga (
    id_evento INT,
    id_categoria INT,
    precio DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (id_evento, id_categoria),
    UNIQUE (id_evento, id_categoria),
    FOREIGN KEY (id_evento) REFERENCES Evento(id_evento),
    FOREIGN KEY (id_categoria) REFERENCES Categoria(id_categoria)
);

-- Tabla para la entidad Entrada (Evento-Asistente-Categoria)
CREATE TABLE Entrada (
    id_entrada INT AUTO_INCREMENT PRIMARY KEY,
    id_evento INT NOT NULL,
    id_asistente INT NOT NULL,
    id_categoria INT NOT NULL,
    FOREIGN KEY (id_evento) REFERENCES Evento(id_evento),
    FOREIGN KEY (id_asistente) REFERENCES Asistente(id_asistente),
    FOREIGN KEY (id_categoria) REFERENCES Categoria(id_categoria)
```

```
);



/* ----------------------------------------------------------------------------------------
Creación de 3 triggers que se encarguen de insertar, actualizar y eliminar
los datos de el atributo de actividad, coste, cuando se inserten, actualizen
o eliminen datos en la tabla participe.
----------------------------------------------------------------------------------------*/

-- Trigger para inserciones en Participe
DELIMITER //
CREATE TRIGGER actualiza_coste_actividad_insert
AFTER INSERT ON Participe
FOR EACH ROW
BEGIN
    UPDATE Actividad
    SET coste = coste + NEW.cache
    WHERE id_actividad = NEW.id_actividad;
END//
DELIMITER ;

-- Trigger para actualizaciones en Participe
DELIMITER //
CREATE TRIGGER actualiza_coste_actividad_update
AFTER UPDATE ON Participe
FOR EACH ROW
BEGIN
    UPDATE Actividad
    SET coste = coste - OLD.cache + NEW.cache
    WHERE id_actividad = NEW.id_actividad;
END//
DELIMITER ;

-- Trigger para eliminaciones en Participe
DELIMITER //
CREATE TRIGGER actualiza_coste_actividad_delete
AFTER DELETE ON Participe
FOR EACH ROW
BEGIN
    UPDATE Actividad
    SET coste = coste - OLD.cache
    WHERE id_actividad = OLD.id_actividad;
END//
DELIMITER ;



/* ----------------------------------------------------------------------------------------
Inserción de Datos
----------------------------------------------------------------------------------------*/
SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Ubicacion.csv'
INTO TABLE Ubicacion
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ';'          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(n_ubicacion,direccion,ciudad,aforo,precio_alquiler,caracteristicas);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Actividad.csv'
INTO TABLE Actividad
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(n_actividad,tipo);
```

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Evento.csv'
INTO TABLE Evento
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(n_evento,fecha_hora,descripcion,id_actividad,id_ubicacion);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Artista.csv'
INTO TABLE Artista
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(n_artista,biografia);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Asistente.csv'
INTO TABLE Asistente
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(nombre,ape1,ape2,email);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Telefono.csv'
INTO TABLE Telefono
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(telefono,id_asistente);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Categoria.csv'
INTO TABLE Categoria
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(n_categoria);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\participe.csv'
INTO TABLE participe
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(id_actividad,id_artista,cache);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\valore.csv'
INTO TABLE valore
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(id_evento,id_asistente,valoracion);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\valga.csv'
INTO TABLE valga
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(id_evento,id_categoria,precio);
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\Entrada.csv'
INTO TABLE Entrada
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','          -- delimitador de campos
LINES TERMINATED BY '\r\n'          -- terminador de línea
IGNORE 1 LINES                    -- ignora la primera línea (cabecera)
(id_evento,id_asistente,id_categoria);

```

```
257
258
259 /* ------------------------------------------------------------------------------------------
260 Creación de un trigger que se encargue de controlar que el nº de
261 entradas vendidas para cada evento no supere el aforo respectivo.
262 ------------------------------------------------------------------------------------------*/
263 DELIMITER //
264 CREATE TRIGGER before_insert_entrada
265 BEFORE INSERT ON Entrada
266 FOR EACH ROW
267 BEGIN
268     DECLARE entradas_vendidas INT;
269     DECLARE aforo_max INT;
270
271     -- Obtener el aforo máximo de la ubicación del evento
272     SELECT u.aforo INTO aforo_max
273     FROM Evento e
274     JOIN Ubicacion u ON e.id_ubicacion = u.id_ubicacion
275     WHERE e.id_evento = NEW.id_evento;
276
277     -- Contar cuántas entradas han sido vendidas para este evento
278     SELECT COUNT(*) INTO entradas_vendidas
279     FROM Entrada
280     WHERE id_evento = NEW.id_evento;
281
282     -- Comprobar si el aforo se superará
283     IF (entradas_vendidas + 1) > aforo_max THEN
284         SIGNAL SQLSTATE '45000'
285         SET MESSAGE_TEXT = 'Error: No se puede vender más entradas. Se ha alcanzado el aforo
                máximo del evento.';
286     END IF;
287 END//
288 DELIMITER ;
```

Consultas

1. Consulta de eventos por el tipo de actividad

```
1         -- ¿Qué eventos se han desarollado cuya actividad sea de tipo 'Música'?
2         SELECT e.id_evento, e.n_evento, a.tipo, a.n_actividad
3         FROM evento e
4         JOIN actividad a ON e.id_actividad = a.id_actividad
5         WHERE a.tipo = 'Música';
```

id_evento	n_evento	tipo	n_actividad
1	Noche de Rock	Música	Concierto de Rock
4	Jazz en el Parque	Música	Festival de Jazz
10	Música Electrónica en Vivo	Música	Festival de Música Electrónica

```
1         -- ¿Qué eventos se han desarollado cuya actividad sea de tipo 'Teatro'?
2         SELECT e.id_evento, e.n_evento, a.tipo, a.n_actividad
3         FROM evento e
4         JOIN actividad a ON e.id_actividad = a.id_actividad
5         WHERE a.tipo = 'Teatro';
```

id_evento	n_evento	tipo	n_actividad
2	Teatro de la Vida	Teatro	Obra de Teatro
11	Festival de Artes Escénicas	Teatro	Obra de Teatro

2. Número de eventos de cada tipo de actividad

```sql
SELECT a.tipo AS tipo_actividad, COUNT(*) AS numero_eventos
FROM evento e
JOIN actividad a ON e.id_actividad = a.id_actividad
GROUP BY a.tipo;
```

tipo_actividad	numero_eventos
Música	3
Teatro	2
Arte	4
Conferencia	2
Danza	1
Cine	1

3. ¿En qué fecha se han realizado más eventos?

```sql
SELECT DATE(fecha_hora) AS fecha, COUNT(*) AS numero_eventos
FROM evento
GROUP BY DATE(fecha_hora)
HAVING COUNT(*) = (
    SELECT MAX(eventos_por_dia)
    FROM (
        SELECT DATE(fecha_hora) AS fecha, COUNT(*) AS eventos_por_dia
        FROM evento
        GROUP BY DATE(fecha_hora)
    ) AS subconsulta
)
ORDER BY fecha;
```

fecha	numero_eventos
2024-10-25	2

4. ¿En qué ciudad se han realizado más eventos?

```sql
SELECT u.ciudad, COUNT(*) AS numero_eventos
FROM evento e
JOIN ubicacion u ON e.id_ubicacion = u.id_ubicacion
GROUP BY u.ciudad
HAVING COUNT(*) = (
    SELECT MAX(eventos_por_ciudad)
    FROM (
        SELECT u.ciudad, COUNT(*) AS eventos_por_ciudad
        FROM evento e
        JOIN ubicacion u ON e.id_ubicacion = u.id_ubicacion
        GROUP BY u.ciudad
    ) AS subconsulta
)
ORDER BY u.ciudad;
```

ciudad	numero_eventos
Barcelona	5

5. Consulta sobre actividades en las que participan un número determinado de artistas

```sql
-- ¿En qué actividades participa sólo un artista?
SELECT a.n_actividad, COUNT(p.id_artista) AS numero_artistas
FROM actividad a
JOIN participe p ON a.id_actividad = p.id_actividad
GROUP BY a.id_actividad
HAVING COUNT(p.id_artista) = 1;
```

	n_actividad	numero_artistas
▶	Charla Motivacional	1
	Ciclo de Cine	1
	Conferencia sobre Tecnología	1
	Espectáculo de Danza	1
	Festival de Jazz	1
	Festival de Música Electrónica	1
	Taller de Pintura	1

```sql
-- ¿En qué actividades participan dos artistas?
SELECT a.n_actividad, COUNT(p.id_artista) AS numero_artistas
FROM actividad a
JOIN participe p ON a.id_actividad = p.id_actividad
GROUP BY a.id_actividad
HAVING COUNT(p.id_artista) = 2;
```

	n_actividad	numero_artistas
▶	Exposición de Arte	2
	Obra de Teatro	2

6. ¿En qué ciudad se han realizado sólo eventos de teatro?

```sql
SELECT u.ciudad
FROM evento e
JOIN actividad a ON e.id_actividad = a.id_actividad
JOIN ubicacion u ON e.id_ubicacion = u.id_ubicacion
GROUP BY u.ciudad
HAVING COUNT(DISTINCT a.tipo) = 1
   AND MAX(a.tipo) = 'Teatro';
```

	ciudad
▶	Valencia

7. ¿Qué evento ha sido valorado más veces con 1 (mínimo)?

```sql
SELECT e.n_evento, COUNT(v.valoracion) AS cantidad_ceros
FROM evento e
LEFT JOIN valore v ON e.id_evento = v.id_evento
WHERE v.valoracion = 1
GROUP BY e.id_evento, e.n_evento
HAVING COUNT(v.valoracion) = (
    SELECT MAX(cantidad_ceros)
    FROM (
        SELECT COUNT(v.valoracion) AS cantidad_ceros
        FROM evento e
        LEFT JOIN valore v ON e.id_evento = v.id_evento
        WHERE v.valoracion = 1
        GROUP BY e.id_evento
    ) AS subconsulta
)
ORDER BY e.n_evento;
```

	n_evento	cantidad_ceros
▶	Mercado de Arte Local	1

8. ¿Cuántas entradas se han vendido para el evento 'Noche de Rock'?

```
        SELECT e.n_evento, COUNT(en.id_entrada) AS entradas_vendidas
        FROM Entrada en
        JOIN Evento e ON en.id_evento = e.id_evento
        WHERE e.n_evento = 'Noche de Rock';
```

n_evento	entradas_vendidas
Noche de Rock	3

9. ¿Qué artistas han participado en el evento 'Noche de Rock'?

```
        SELECT e.n_evento, ar.n_artista, p.cache
        FROM Participe p
        JOIN Evento e ON p.id_actividad = e.id_actividad
        JOIN Artista ar ON p.id_artista = ar.id_artista
        WHERE e.n_evento = 'Noche de Rock';
```

n_evento	n_artista	cache
Noche de Rock	Los Rockeros	500.00
Noche de Rock	Banda de Jazz	450.00
Noche de Rock	Grupo de Danza	300.00

10. Consulta sobre el número de eventos y el número de entradas vendidas de cada uno de ellos

```
        SELECT e.n_evento, COUNT(en.id_entrada) AS entradas_vendidas
        FROM Entrada en
        JOIN Evento e ON en.id_evento = e.id_evento
        GROUP BY e.n_evento
        ORDER BY entradas_vendidas DESC;
```

n_evento	entradas_vendidas
Exposición de Fotografía	4
Noche de Rock	3
Teatro de la Vida	2
Arte en la Calle	2
Jazz en el Parque	2
Inspírate	2
Pintura al Aire Libre	2
Danza en Escena	2
Futuro Digital	2
Cine Bajo las Estrellas	2
Música Electrónica en Vivo	2
Festival de Artes Escéni...	2
Mercado de Arte Local	2

11. ¿Cuál es el aforo disponible para el evento 'Cine Bajo las Estrellas'?

```
        SELECT e.n_evento, u.aforo - COUNT(en.id_entrada) AS aforo_restante
        FROM Evento e
        JOIN Ubicacion u ON e.id_ubicacion = u.id_ubicacion
        LEFT JOIN Entrada en ON e.id_evento = en.id_evento
        WHERE e.n_evento = 'Cine Bajo las Estrellas'
        GROUP BY e.n_evento, u.aforo;
```

n_evento	aforo_restante
Cine Bajo las Estrellas	248

12. ¿Qué eventos están programados para el día '2024-11-10'?

```
        SELECT e.n_evento, e.fecha_hora, u.n_ubicacion
        FROM Evento e
        JOIN Ubicacion u ON e.id_ubicacion = u.id_ubicacion
        WHERE DATE(e.fecha_hora) = '2024-11-10';
```

	n_evento	fecha_hora	n_ubicacion
▶	Inspírate	2024-11-10 16:00:00	Teatro Principal

13. ¿Cuál es la fecha con más eventos?

```
        SELECT DATE(fecha_hora) AS fecha_evento, COUNT(*) AS total_eventos
        FROM Evento
        GROUP BY DATE(fecha_hora)
        HAVING COUNT(*) = (
            SELECT MAX(eventos_por_dia)
            FROM (
                SELECT DATE(fecha_hora) AS fecha_evento, COUNT(*) AS eventos_por_dia
                FROM Evento
                GROUP BY DATE(fecha_hora)
            ) AS subconsulta
        )
        ORDER BY fecha_evento;
```

	fecha_evento	total_eventos
▶	2024-10-25	2

14. ¿Qué eventos están programados para esta fecha?

```
        SELECT e.n_evento, e.fecha_hora, u.n_ubicacion
        FROM Evento e
        JOIN Ubicacion u ON e.id_ubicacion = u.id_ubicacion
        WHERE DATE(e.fecha_hora) = '2024-10-25';
```

	n_evento	fecha_hora	n_ubicacion
▶	Teatro de la Vida	2024-10-25 19:30:00	Teatro Gran Vía
	Arte en la Calle	2024-10-25 11:00:00	Centro Cultural

15. ¿Cuáles son las actividades en las que ha participado el artista 'Banda de Jazz'?

```
        SELECT ar.n_artista, a.n_actividad, p.cache
        FROM Participe p
        JOIN Actividad a ON p.id_actividad = a.id_actividad
        JOIN Artista ar ON p.id_artista = ar.id_artista
        WHERE ar.n_artista = 'Banda␣de␣Jazz';
```

	n_artista	n_actividad	cache
▶	Banda de Jazz	Concierto de Rock	450.00
	Banda de Jazz	Festival de Jazz	400.00

16. ¿Qué asistentes no han valorado algún evento de los que han ido, y cuál?

```
        SELECT a.nombre, a.ape1, e.n_evento
        FROM Entrada en
        JOIN Asistente a ON en.id_asistente = a.id_asistente
        JOIN Evento e ON en.id_evento = e.id_evento
        LEFT JOIN Valore v ON en.id_evento = v.id_evento AND en.id_asistente = v.
            id_asistente
        WHERE v.valoracion IS NULL;
```

	nombre	ape1	n_evento
▶	Patricia	Ruiz	Exposición de Fotografía

17. ¿Cuál es la valoración media de cada evento?

```
        SELECT e.n_evento, AVG(v.valoracion) AS valoracion_media
        FROM Evento e
        JOIN Valore v ON e.id_evento = v.id_evento
        GROUP BY e.n_evento
        ORDER BY valoracion_media DESC;
```

	n_evento	valoracion_media
▶	Noche de Rock	4.6667
	Jazz en el Parque	4.5000
	Inspírate	4.5000
	Futuro Digital	4.5000
	Pintura al Aire Libre	4.0000
	Música Electrónica en Vivo	4.0000
	Exposición de Fotografía	4.0000
	Arte en la Calle	3.5000
	Danza en Escena	3.5000
	Teatro de la Vida	3.0000
	Festival de Artes Escénicas	3.0000
	Mercado de Arte Local	3.0000
	Cine Bajo las Estrellas	2.5000

18. ¿Cuánto cuestan las entradas VIP de cada evento?

```
        SELECT e.n_evento, c.n_categoria, v.precio
        FROM Evento e
        JOIN Valga v ON e.id_evento = v.id_evento
        JOIN Categoria c ON v.id_categoria = c.id_categoria
        WHERE c.n_categoria = 'VIP';
```

	n_evento	n_categoria	precio
▶	Noche de Rock	VIP	20.00
	Teatro de la Vida	VIP	15.00
	Arte en la Calle	VIP	10.00
	Jazz en el Parque	VIP	20.00
	Inspírate	VIP	18.00
	Pintura al Aire Libre	VIP	12.00
	Danza en Escena	VIP	20.00
	Futuro Digital	VIP	15.00
	Cine Bajo las Estrellas	VIP	10.00
	Música Electrónica en Vivo	VIP	20.00

19. Esta vista proporciona un resumen de los datos relacionados con las entradas emitidas, mostrando quién compró cada entrada (nombre y apellido del asistente), para qué evento (nombre del evento) y en qué categoría se clasifica la entrada (nombre de la categoría). Combina y organiza los datos de varias tablas para ofrecer una vista consolidada y fácil de consultar de las entradas, asistentes, eventos y categorías asociadas.

```
CREATE VIEW Vista_Entrada_Asistente_Categoria AS
SELECT
    en.id_entrada,
    a.id_asistente,
    a.nombre AS nombre_asistente,
    a.ape1 AS apellido_asistente,
    e.id_evento,
    e.n_evento AS nombre_evento,
    c.id_categoria,
    c.n_categoria AS nombre_categoria
FROM Entrada en
JOIN Asistente a ON en.id_asistente = a.id_asistente
JOIN Evento e ON en.id_evento = e.id_evento
JOIN Categoria c ON en.id_categoria = c.id_categoria;
```

id_entrada	id_asistente	nombre_asistente	apellido_asistente	id_evento	nombre_evento	id_categoria	nombre_categoria
1	1	Carlos	Gómez	1	Noche de Rock	1	VIP
6	1	Carlos	Gómez	5	Inspírate	4	Estudiante
11	1	Carlos	Gómez	10	Música Electrónica en Vivo	4	Estudiante
17	1	Carlos	Gómez	3	Arte en la Calle	2	General
26	1	Carlos	Gómez	13	Exposición de Fotografía	4	Estudiante
2	2	María	Fernández	1	Noche de Rock	2	General
7	2	María	Fernández	6	Pintura al Aire Libre	3	Reducida
18	2	María	Fernández	4	Jazz en el Parque	3	Reducida
27	2	María	Fernández	13	Exposición de Fotografía	2	General
3	3	Javier	López	2	Teatro de la Vida	1	VIP
8	3	Javier	López	7	Danza en Escena	5	Grupo
19	3	Javier	López	5	Inspírate	1	VIP
28	3	Javier	López	13	Exposición de Fotografía	5	Grupo
4	4	Ana	Jiménez	3	Arte en la Calle	3	Reducida
9	4	Ana	Jiménez	8	Futuro Digital	2	General
20	4	Ana	Jiménez	6	Pintura al Aire Libre	2	General
5	5	Lucía	Ramírez	4	Jazz en el Parque	2	General

20. Consulta con ayuda de la vista anterior que muestre el nombre del asistente, apellido, categoría del evento, y la cantidad de eventos dentro de esa categoría a los que ha asistido, pero solo si ha asistido a más de un evento en la misma categoría.

```
SELECT
    veac.nombre_asistente,
    veac.apellido_asistente,
    veac.nombre_categoria,
    COUNT(DISTINCT veac.id_evento) AS total_eventos
FROM Vista_Entrada_Asistente_Categoria veac
GROUP BY veac.nombre_asistente, veac.apellido_asistente, veac.nombre_categoria
HAVING COUNT(DISTINCT veac.id_evento) > 1;
```

nombre_asistente	apellido_asistente	nombre_categoria	total_eventos
Ana	Jiménez	General	2
Carlos	Gómez	Estudiante	3
Javier	López	Grupo	2
Javier	López	VIP	2
María	Fernández	General	2
María	Fernández	Reducida	2

En conclusión, la creación de esta base de datos para ”ArteVida Cultural” no solo permite gestionar de manera eficiente la organización de eventos y todas sus actividades relacionadas, sino que también ofrece una herramienta escalable y flexible que se adapta a las necesidades cambiantes de la empresa. La implementación de restricciones y triggers asegura la integridad de los datos y optimiza la precisión de la información almacenada. Además, las consultas desarrolladas proporcionan información clave para la toma de decisiones estratégicas, mejorando la capacidad de planificación y análisis. En conjunto, este sistema proporciona una solución robusta para gestionar la complejidad de los eventos culturales, garantizando un control completo y una visión integral de la operación.