Creación de una base de datos

Alumno: Pablo Pérez Calvo

Índice

1. Diseño Conceptual

1.1. Identificación de entidades

Se define como **entidad** toda cosa u objeto en el mundo real que es distinguible de todos los demás objetos. Es un concepto que representa algo significativo dentro de un dominio concreto. Un **atributo** describe las características de una entidad o relación.

En nuestro caso queremos crear la base de datos de una empresa que organiza eventos culturales únicos. Veamos cuales son las entidades y sus correspondientes atributos.

1.1.1. Entidad Actividad

En nuestra empresa ofrecemos una serie de **actividades** *que tienen un nombre, un tipo: concierto de distintos tipos de música (clásica, pop, blues, soul, rock and roll, jazz, reggaeton, góspel, country, . . .), exposiciones, obras de teatro y conferencias, aunque en un futuro estamos dispuestos a organizar otras actividades. Además, en cada actividad participa uno o varios artistas y un coste (suma del caché de los artistas).*

- IdActividad: Se trata de la clave primaria de la entidad, identifica de manera única a cada una de las actividades. Atributo simple de tipo entero, único para cada actividad.
- NomActividad: Atributo simple que almacena el título de una actividad. Se trata de una cadena de caracteres.
- Tipo: Categoría a la que pertenece la actividad (concierto, conferencia, etc.). Atributo simple de tipo cadena de caracteres.
- Coste: Atributo derivado que corresponde al costo asociado a la actividad. Atributo de tipo numérico ya que se trata de un valor monetario que depende del caché de los artistas que participen en la actividad, por defecto es 0.

Figura 1: Entidad Actividad

1.1.2. Entidad Artista

El **artista** *tiene un nombre, un caché que depende de la actividad en la que participe y una breve biografía.*

- IdArtista: Se trata de la clave primaria de la entidad, identifica de manera única a cada uno de los artistas. Atributo simple de tipo entero, único para cada artista.
- NomArtista: Atributo simple que almacena el nombre de un artista. Se trata de una cadena de caracteres.
- Biografía: Breve descripción de la trayectoria del artista. Atributo simple de tipo cadena de caracteres, 500 caracteres como máximo.

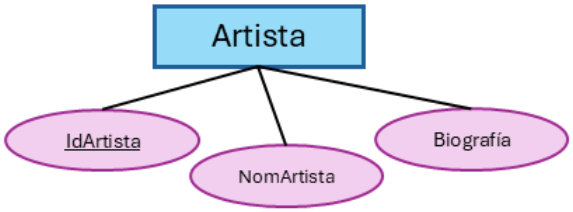

Figura 2: Entidad Artista

1.1.3. Entidad Ubicación

La **ubicación** *tendrá un nombre (Teatro Maria Guerrero, Estadio Santiago Bernabeu, . . .), dirección, ciudad o pueblo, aforo, precio del alquiler y características*

- IdUbicacion: Se trata de la clave primaria de la entidad, identifica de manera única a cada una de las ubicaciones. Atributo simple de tipo entero, único para cada ubicación.
- NomUbicacion: Atributo simple que almacena el nombre de la ubicación. Se trata de una cadena de caracteres.
- Dirección: Atributo simple de tipo cadena de caracteres, indica la dirección del lugar donde se realizará el evento.
- Localidad: Ciudad o pueblo al que pertenece la ubicación. Atributo simple de tipo cadena de caracteres.
- Aforo: Capacidad máxima de personas que pueden asistir al evento. Atributo simple de tipo numérico.
- PrecioAlquiler: Se trata del precio que cuesta alquilar el recinto para realizar el evento. Atributo simple de tipo numérico porque corresponde a un valor monetario.
- Caracteristicas: Breve descripción de las características del lugar. Atributo simple de tipo cadena de caracteres, máximo 200 caracteres .

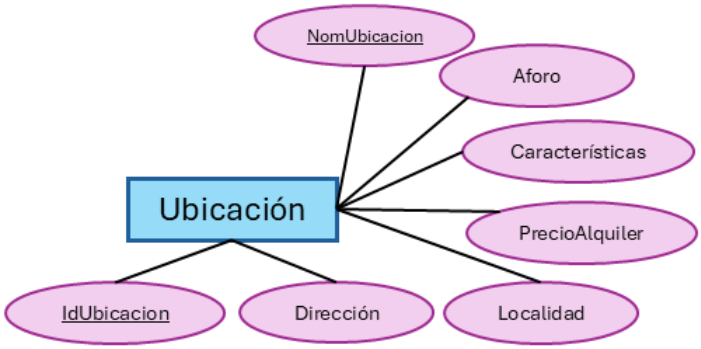

Figura 3: Entidad Ubicacion

1.1.4. Entidad Evento

De cada **evento** *tenemos que saber el nombre del evento (p.e. "VI festival de música clásica de Alcobendas"), la actividad, la ubicación, el precio de la entrada, la fecha y la hora, así como una breve descripción del mismo. En un evento sólo se realiza una actividad*

- IdEvento: Se trata de la clave primaria de la entidad, identifica de manera única a cada uno de los eventos. Atributo simple de tipo entero, único para cada evento.
- NomEvento: Atributo simple que almacena el nombre del evento que se quiere realizar. Se trata de una cadena de caracteres.
- PrecioEvento: Atributo simple de tipo numérico ya que corresponde a un valor monetario. Indica el precio del evento.
- Fecha: Atributo simple de tipo fecha que indica cuándo se realizará el evento. El formato de la fecha será YYYY-MM-DD.
- Hora: Atributo simple de tipo hora que corresponde a la hora en la que se realizará el evento. El formato de la hora será HH:MM:SS.
- Descripcion: Breve descripción de las características del evento. Atributo simple de tipo cadena de caracteres, máximo 500 caracteres .

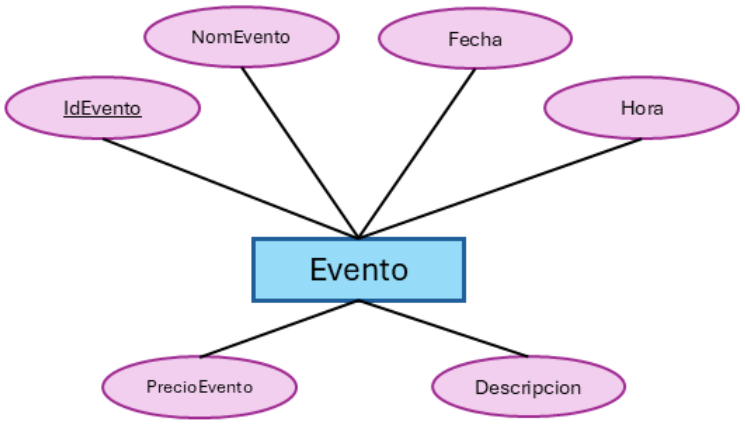

Figura 4: Entidad Evento

1.1.5. Entidad Asistente

También tendremos en cuenta los **asistentes** *a los eventos, de los que sabemos su nombre completo, sus teléfonos de contacto y su email. Una persona puede asistir a más de un evento y a un evento pueden asistir varias personas.*

- IdAsistente: Se trata de la clave primaria de la entidad, identifica de manera única a cada uno de los asistentes. Atributo simple de tipo entero, único para cada asistente.
- NomAsisente: Atributo simple que almacena el nombre de la persona que asiste a un evento. Se trata de una cadena de caracteres.
- Email: Atributo simple de tipo cadena de caracteres, el correo electrónico del asistente.
- Teléfono: Atributo simple de tipo numérico con 9 cifras que indica el número de teléfono del asistente.

Figura 5: Entidad Asistentes

1.2. Identificación de relaciones

En los eventos **se realizan** actividades. A cada evento se le asigna una actividad y una misma actividad puede ser realizada en distintos eventos.

Cardinalidad de Actividad: en un evento se realiza como mínimo y máximo una actividad $(1, 1)$.

Cardinalidad de Evento: una actividad debe realizarse como mínimo una vez en algún evento (si fuera 0 no se añadiría a la base de datos) y como máximo n veces $(1, n)$.

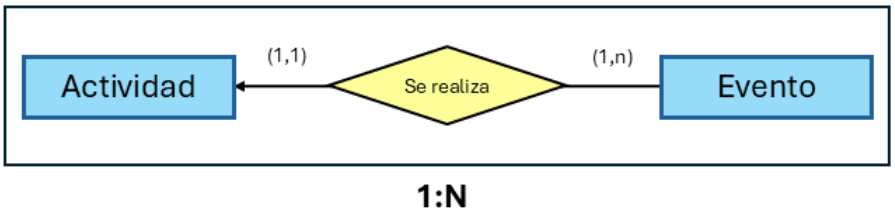

Figura 6: Relación "se realiza" (Actividad-Evento)

Los eventos **tienen lugar** en solo una ubicación, esta ubicación puede ser utilizada para varios eventos.

Cardinalidad de Evento: una ubicación puede ser utilizada como mínimo una vez y como máximo n veces $(1, n)$.

Cardinalidad de Ubicación: un evento tiene lugar como mínimo y máximo en una sola ubicación $(1, 1)$.

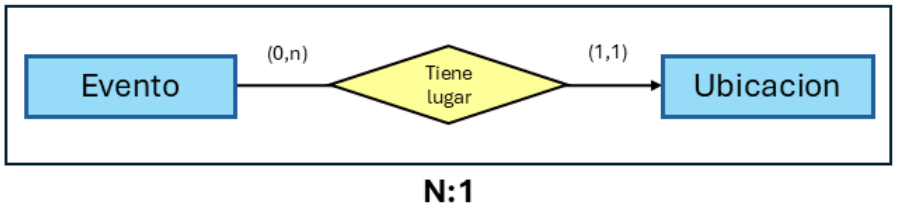

Figura 7: Relación "tiene lugar" (Evento-Ubicación

Los asistentes **asisten** a los eventos. En la relación *asiste* hay un atributo porque cada vez que una persona asiste a un evento, tenemos que saber su reseña con una calificación del 1 al 10.

Cardinalidad de Evento: una persona puede no estar asignada a ningún evento y puede asistir a varios eventos $(0, n)$.

Cardinalidad de Asistente: Un evento puede no tener asistentes aún y puede tener muchos asistentes $(0, n)$.

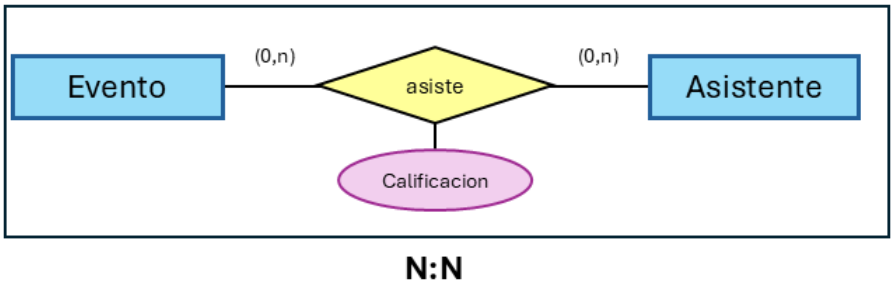

Figura 8: Relación "asiste" (Evento-Asistente)

Los artistas **participan** en las actividades que se realizan en los eventos. La relación *participa* tiene un atributo, el caché del artista que depende de la actividad que vaya a realizar.

Cardinalidad de Actividad: un artista debe participar al menos en una actividad y puede participar en distintas actividades $(1, n)$.

Cardinalidad de Artista: Una actividad debe tener asignado al menos un artista y pueden participar varios artistas en la misma actividad $(1, n)$.

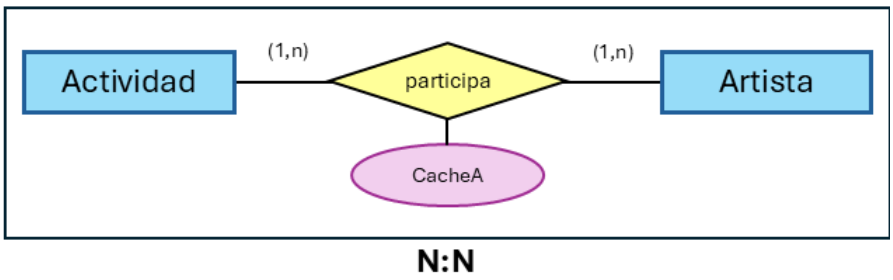

Figura 9: Relación "participa" (Actividad-Artista)

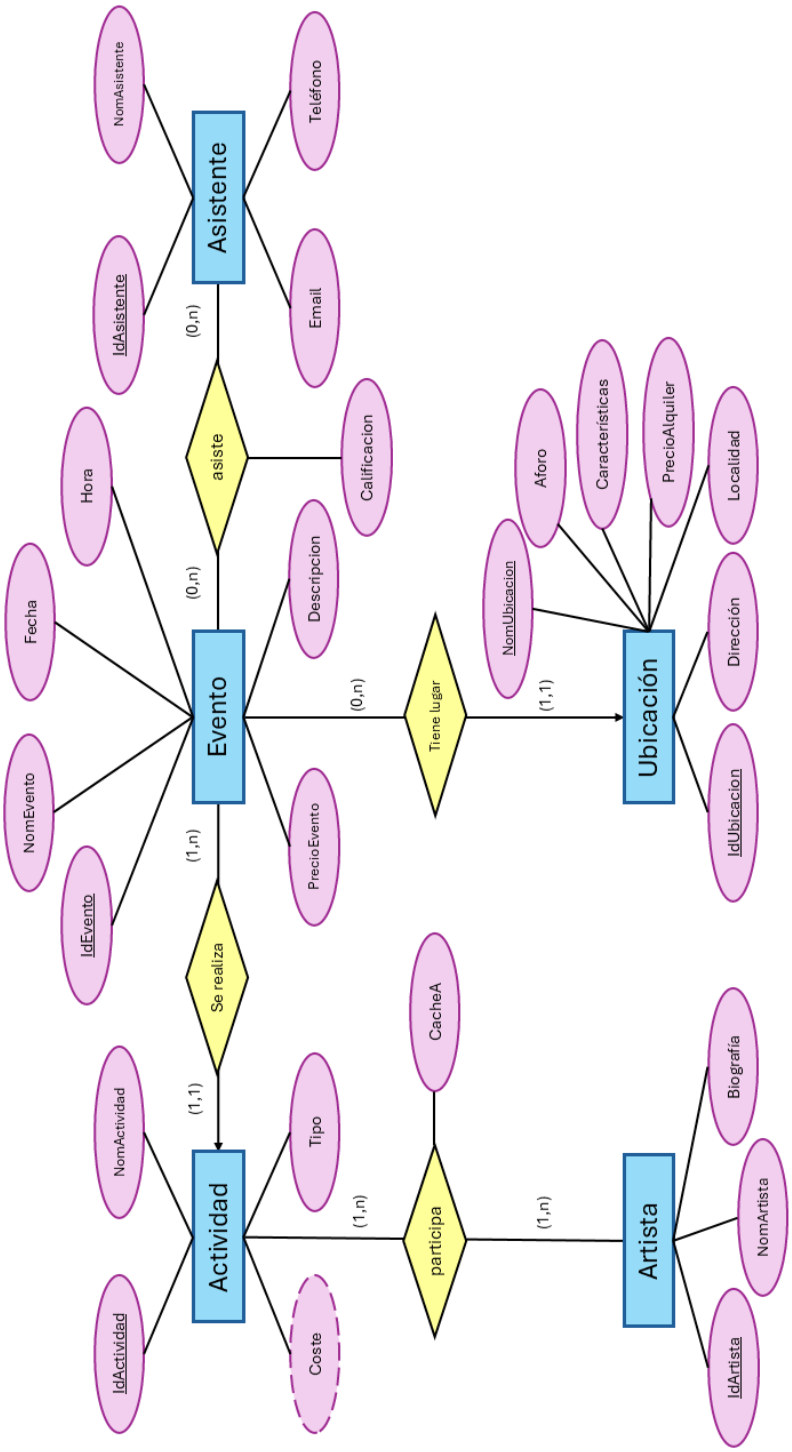

Figura 10: Diagrama Entidad-Relación

2. Modelo relacional

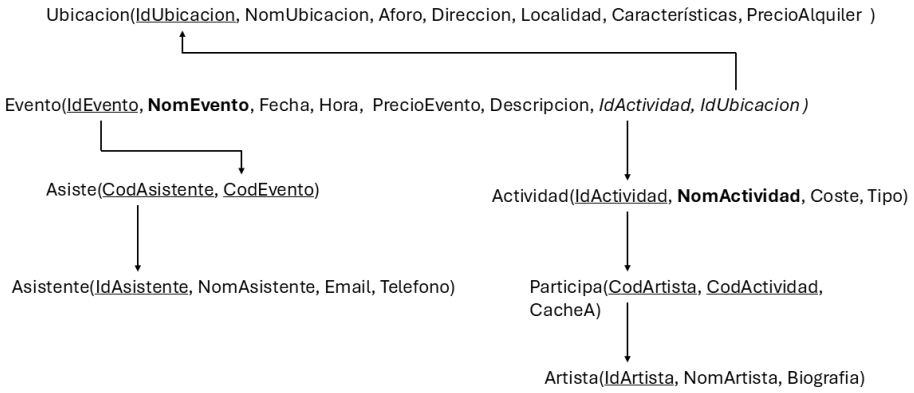

Figura 11: Modelo relacional

3. Implementación en SQL

```sql
-- PABLO PEREZ CALVO
DROP DATABASE IF EXISTS Organizaeventos;
CREATE DATABASE Organizaeventos;
USE Organizaeventos;

DROP TABLE IF EXISTS Evento;
DROP TABLE IF EXISTS Ubicacion;
DROP TABLE IF EXISTS Actividad;
DROP TABLE IF EXISTS Asistente;
DROP TABLE IF EXISTS Artista;
DROP TABLE IF EXISTS Participa;
DROP TABLE IF EXISTS Asiste;
```

```sql
-- CREACION DE TABLAS

CREATE TABLE Actividad (
    IdActividad SMALLINT PRIMARY KEY,
    NomActividad VARCHAR(100),
    Coste NUMERIC(10,2) DEFAULT 0,
    Tipo VARCHAR(50)
);


```

```sql
CREATE TABLE Ubicacion (
    IdUbicacion SMALLINT PRIMARY KEY,
    NomUbicacion VARCHAR(50),
    PrecioAlquiler NUMERIC(10,2) NOT NULL,
    Direccion VARCHAR(100) NOT NULL,
    Aforo NUMERIC(10,0),
    Localidad VARCHAR(20),
    Caracteristicas VARCHAR(200)
);

CREATE TABLE Evento (
    IdEvento SMALLINT PRIMARY KEY,
    NomEvento VARCHAR(100) NOT NULL,
    Fecha DATE NOT NULL,
    Hora TIME,
    PrecioEvento NUMERIC(6,2) NOT NULL,
    Descripcion VARCHAR(500),
    CodActividad SMALLINT,
    CodUbicacion SMALLINT,
    FOREIGN KEY (CodActividad) REFERENCES
        Actividad(IdActividad)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (CodUbicacion) REFERENCES
        Ubicacion(IdUbicacion)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE Asistente(
    IdAsistente SMALLINT PRIMARY KEY,
    NomAsistente VARCHAR(50) NOT NULL,
    Email VARCHAR(50),
    Telefono Numeric(9,0)
);

CREATE TABLE Artista(
    IdArtista SMALLINT PRIMARY KEY,
    NomArtista VARCHAR(50) NOT NULL,
    Biografia VARCHAR(500)
);

CREATE TABLE Participa (
    CodArtista SMALLINT,
    CodActividad SMALLINT,
    CacheA NUMERIC(10,2),
    PRIMARY KEY (CodArtista, CodActividad),
    FOREIGN KEY (CodArtista) REFERENCES Artista(IdArtista)
        ON DELETE CASCADE,
    FOREIGN KEY (CodActividad) REFERENCES
```

```
        Actividad(IdActividad) ON DELETE CASCADE
);



CREATE TABLE Asiste (
    CodAsistente SMALLINT,
    CodEvento SMALLINT,
    PRIMARY KEY (CodAsistente, CodEvento),
    FOREIGN KEY (CodAsistente) REFERENCES
        Asistente(IdAsistente) ON DELETE CASCADE,
    FOREIGN KEY (CodEvento) REFERENCES Evento(IdEvento) ON
        DELETE CASCADE,
    Calificacion NUMERIC(2,0)
);
```

```
-- TRIGGERS

DELIMITER //
-- 1. TRIGGER para actualizar el coste de una actividad
    despues de anadir un artista con su respectivo cache
CREATE TRIGGER AnadeCosteActividad
AFTER INSERT ON Participa
FOR EACH ROW
BEGIN
    UPDATE Actividad
    SET Coste = Coste + NEW.CacheA
    WHERE IdActividad = NEW.CodActividad;
END;
//

-- 2. TRIGGER para actualizar el coste de una actividad
    tras eliminar un artista de dicha actividad
CREATE TRIGGER RestaCosteActividad
AFTER DELETE ON Participa
FOR EACH ROW
BEGIN
    UPDATE Actividad
    SET Coste = Coste - OLD.CacheA
    WHERE IdActividad = OLD.CodActividad;
END;
//
```

```sql
-- 3. TRIGGER para verificar que el aforo maximo de cada ubicacion no se supera al anadir asistentes
CREATE TRIGGER VerificarAforo
BEFORE INSERT ON Asiste
FOR EACH ROW
BEGIN
    DECLARE aforomax INT;
    DECLARE numasistentes INT;
    SELECT ubi.Aforo INTO aforomax
    FROM Evento as ev
    JOIN Ubicacion as ubi ON ev.CodUbicacion = ubi.IdUbicacion
    WHERE ev.IdEvento = NEW.CodEvento;

    SELECT COUNT(*) INTO numasistentes
    FROM Asiste
    WHERE CodEvento = NEW.CodEvento;

    IF numasistentes >= aforomax THEN
        SIGNAL SQLSTATE '45523' SET MESSAGE_TEXT = 'El aforo esta completo, no quedan mas entradas disponibles para este evento';
    END IF;
END;
DELIMITER ;
```

```sql
-- IMPORTACION DE TODOS LOS DATOS

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Actividades.csv'
INTO TABLE actividad
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 rows
(IdActividad, NomActividad, Tipo );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Ubicacion.csv'
INTO TABLE ubicacion
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 rows
(IdUbicacion, NomUbicacion, PrecioAlquiler, Direccion, Aforo, Localidad, Caracteristicas);


```

```
21 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
       8.0/Uploads/Evento.csv'
22 INTO TABLE evento
23 CHARACTER SET latin1
24 FIELDS TERMINATED BY ';'
25 LINES TERMINATED BY '\n'
26 IGNORE 1 rows
27 (IdEvento, NomEvento, Fecha, Hora, PrecioEvento,
       Descripcion, CodActividad, CodUbicacion);
28
29
30 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
       8.0/Uploads/Asistentes.csv'
31 INTO TABLE asistente
32 CHARACTER SET latin1
33 FIELDS TERMINATED BY ';'
34 LINES TERMINATED BY '\n'
35 IGNORE 1 rows
36 (IdAsistente, NomAsistente, Email, Telefono);
37
38 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
       8.0/Uploads/Asiste.csv'
39 INTO TABLE Asiste
40 CHARACTER SET latin1
41 FIELDS TERMINATED BY ';'
42 LINES TERMINATED BY '\n'
43 IGNORE 1 rows
44 (CodAsistente, CodEvento, Calificacion);
45
46 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
       8.0/Uploads/Artistas.csv'
47 INTO TABLE Artista
48 CHARACTER SET latin1
49 FIELDS TERMINATED BY ';'
50 LINES TERMINATED BY '\n'
51 IGNORE 1 rows
52 (IdArtista, NomArtista, Biografia);
53
54 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
       8.0/Uploads/Participa.csv'
55 INTO TABLE participa
56 CHARACTER SET latin1
57 FIELDS TERMINATED BY ';'
58 LINES TERMINATED BY '\n'
59 IGNORE 1 rows
60 (CodArtista, CodActividad, CacheA);
```

```sql
-- CONSULTAS

-- 4. Obtener el nombre de los asistentes que han ido a la Final del Concurso de Agrupaciones Carnavalescas
SELECT NomAsistente
FROM asistente, asiste
WHERE asistente.IdAsistente = asiste.CodAsistente AND asiste.CodEvento = 1;

-- 5. Obtener todos los nombres de Eventos que se realizan en la Provincia de Cadiz (ya sea un pueblo o Cadiz capital)
SELECT ev.NomEvento, ubi.NomUbicacion, ubi.Localidad
FROM evento as ev, ubicacion as ubi
WHERE ubi.Localidad LIKE '%Cadiz%' and ubi.IdUbicacion = ev.CodUbicacion
order by ev.NomEvento ;

-- 6. Obtener un ranking ordenado de los eventos por su calificacion media dada por los asistentes
-- y el nombre de la persona que mejor nota haya calificado al evento.
SELECT ev.NomEvento, AVG(asis.Calificacion) AS Calificacionmedia,
        (SELECT a.NomAsistente FROM Asistente as a
     JOIN Asiste AS asis ON a.IdAsistente = asis.CodAsistente
     WHERE asis.CodEvento = ev.IdEvento
     ORDER BY asis.Calificacion DESC, a.NomAsistente ASC
      LIMIT 1) AS MejorAsistente
FROM Evento as ev
JOIN Asiste as asis ON ev.IdEvento = asis.CodEvento
GROUP BY ev.IdEvento, ev.NomEvento
ORDER BY Calificacionmedia DESC;

-- 7. Seleccionar las actividades con los artistas mas destacados (con un cache mayor a 5000 euros)
SELECT ac.NomActividad, ar.NomArtista, par.CacheA
FROM Participa as par
JOIN Actividad as ac ON par.CodActividad = ac.IdActividad
JOIN Artista as ar ON par.CodArtista = ar.IdArtista
WHERE par.CacheA > 5000;

-- 8. Queremos conocer todo el detalle de los eventos que se realizaran desde la fecha actual hasta dentro de 30 d as
SELECT ev.NomEvento, ev.Fecha, ev.Hora, ac.NomActividad , ubi.NomUbicacion, ubi.Localidad,
     GROUP_CONCAT(DISTINCT ar.NomArtista SEPARATOR ', ') AS Artistas,
```

```
    COUNT(DISTINCT asis.CodAsistente) AS TotalAsistentes,
        ev.PrecioEvento
FROM Evento as ev
LEFT JOIN Actividad as ac ON ev.CodActividad =
    ac.IdActividad
LEFT JOIN Ubicacion as ubi ON ev.CodUbicacion =
    ubi.IdUbicacion
LEFT JOIN Participa as par ON ac.IdActividad =
    par.CodActividad
LEFT JOIN Artista  as ar ON par.CodArtista = ar.IdArtista
LEFT JOIN Asiste as asis ON ev.IdEvento = asis.CodEvento
WHERE ev.Fecha BETWEEN CURDATE() AND DATE_ADD(CURDATE(),
    INTERVAL 30 DAY)
GROUP BY ev.IdEvento
ORDER BY ev.Fecha ASC;
```

```
-- VISTAS Y CONSULTAS UTILIZANDO LAS VISTAS

-- Crear una vista que calcula los ingresos generados por
    cada evento
CREATE VIEW IngresosEvento AS
SELECT ev.IdEvento, ev.NomEvento, COUNT(asis.CodAsistente)
    * ev.PrecioEvento AS IngresosTotales
FROM Evento as ev
JOIN Asiste as asis ON ev.IdEvento = asis.CodEvento
GROUP BY ev.IdEvento;

-- 9. Seleccionar aquellos eventos cuyos ingresos totales
    hayan sido mas de 100 euros
SELECT * FROM IngresosEvento WHERE IngresosTotales > 100;

-- 10. Determinar los eventos que generan perdidas
    (Ingresos actuales < Inversion en el evento)
SELECT inev.NomEvento, ubi.PrecioAlquiler ,
    inev.IngresosTotales,
    (inev.IngresosTotales - ubi.PrecioAlquiler - ac.Coste)
        AS Beneficio
FROM IngresosEvento as inev
JOIN  Evento as ev ON inev.IdEvento = ev.IdEvento
JOIN Actividad as ac ON ac.IdActividad = ev.CodActividad
JOIN  Ubicacion as ubi ON ev.CodUbicacion = ubi.IdUbicacion
WHERE (inev.IngresosTotales - ubi.PrecioAlquiler -
    ac.Coste) < 0
ORDER BY Beneficio ASC;
```

```sql
-- Crear una vista que permita saber cuantos asistentes va a cada evento
CREATE VIEW AsistentesPorEvento AS
SELECT ev.IdEvento, ev.NomEvento, COUNT(DISTINCT asis.CodAsistente) AS NumeroAsistentes
FROM Evento as ev
JOIN Asiste as asis ON ev.IdEvento = asis.CodEvento
GROUP BY ev.IdEvento;

-- 11. Obtener el Nombre de los eventos con baja ocupacion, es decir, no superen un 60% del porcentaje de llenado
SELECT ev.NomEvento, ubi.NomUbicacion, ubi.Aforo, asiev.Numeroasistentes,
    CONCAT(ROUND( asiev.Numeroasistentes / ubi.Aforo * 100, 2), '%') AS PorcentajeAsistencia
FROM AsistentesporEvento as asiev
JOIN Evento as ev ON asiev.IdEvento = ev.IdEvento
JOIN Ubicacion as ubi ON ev.CodUbicacion = ubi.IdUbicacion
JOIN Asiste as asis ON ev.IdEvento = asis.CodEvento
GROUP BY ev.IdEvento
HAVING PorcentajeAsistencia < 60
ORDER BY PorcentajeAsistencia ASC;

-- 12. Determinar el nombre de los eventos a los que asistan mas personas que al evento "Big Data Conference Europe 2024"
SELECT ev.NomEvento, asiev.NumeroAsistentes
FROM AsistentesporEvento as asiev
JOIN Evento as ev ON asiev.IdEvento = ev.IdEvento
WHERE asiev.NumeroAsistentes > (
    SELECT asiev.NumeroAsistentes
    FROM AsistentesporEvento as asiev
    JOIN Evento as ev ON asiev.IdEvento = ev.IdEvento
    WHERE ev.NomEvento = 'Big Data Conference Europe 2024'
);
```