Creación de una base de datos

Bases de Datos SQL

Bartolomé Mestre Fons

Lunes, 28 de Octubre de 2024

Bases de Datos SQL
2024-2025
UCM - Universidad Complutense de Madrid

Índice

1. Diagrama E-R

A continuación se muestra el diagrama E-R realizado con la herramienta online *draw.io* siguienndo los requisitos enunciados en el problema:

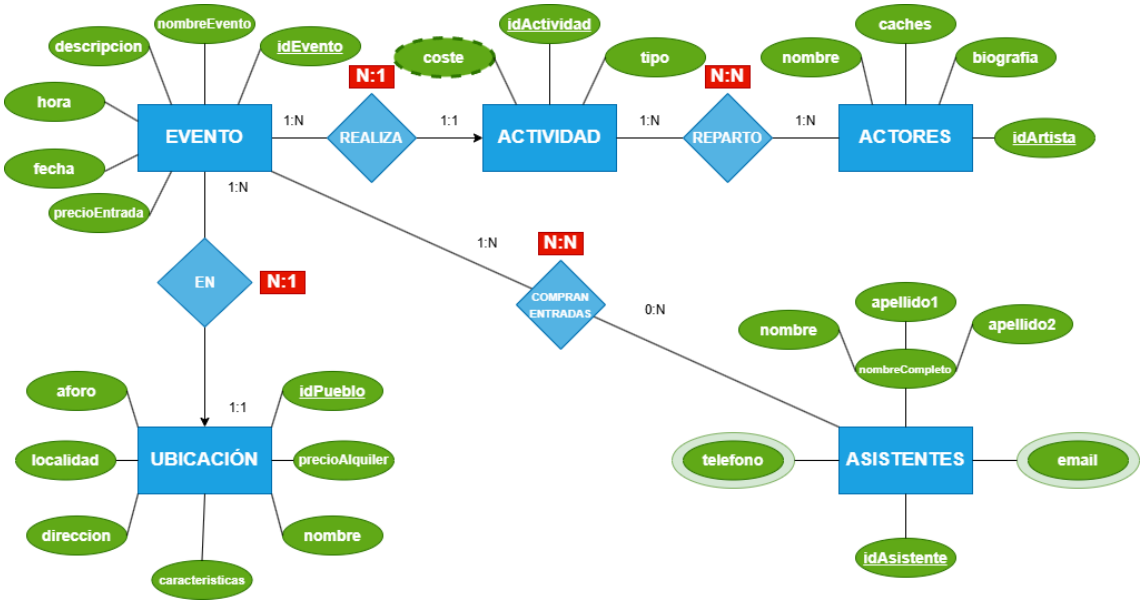

Figura 1: *Diagrama E-R.*

Vayamos a describir primero las entidades principales del diagrama E-R:

1. ***Evento***: es la entidad principal del diagrama E-R, y sobre ella se construirán el resto de entidades. Posee los siguientes atributos:

   - **idEvento**: identificamos cada evento con una ID única, que funcionará como su *Primary Key*. Este atributo ha sido creado para ofrecer un atributo inmutable que cumpla los requisitos de una llave primaria. Uno podría haber elegido como llave primaria el atributo nombreEvento, pero se halla sujeto a modificaciones futuras por parte de la empresa, por ello no es un buen candidato para ser una llave primaria. Se espera que sea una cadena de carácteres numéricos tipo SMALLINT.

   - **nombreEvento**: atributo que representa el nombre del evento. Se espera que sea una cadena de carácteres alfanuméricos tipo VARCHAR.

   - **descripcion**: atributo que ofrece una descripción del evento. Se espera que sea una cadena de carácteres alfanuméricos tipo VARCHAR.

   - **hora**: atributo que representa la hora a la que se realiza el evento. Se espera que su formato sea TIME.

   - **fecha**: atributo que representa la fecha en la que se realizará el evento. Alternativamente, se podrían haber puesto los atributos hora y fecha bajo un mismo atributo compuesto, pero se ha optado por

separarlos. Se espera que su formato sea DATE.

- **precioEntradas**: aributo que representa el precio por entrada del evento. Se espera que su formato sea SMALLINT.

2. ***Ubicación***: indica la ubicación donde se lleva a cabo el evento. Posee los siguientes atributos:
   - **idPueblo**: dicho atributo ha sido creado para poder identificar el pueblo de manera única e inmutable, dado que el nombre del pueblo puede estar sujeto a futuras modificaciones. Se espera un formato de SMALLINT.
   - **precioAlquiler**: es el precio del alquiler de las infraestructuras de la ubicación donde se lleva a cabo el evento. Se espera un formato SMALLINT.
   - **nombre**: es el nombre del pueblo. Se espera un formato VARCHAR.
   - **caracteristicas**: representa un conjunto de características de las instalaciones de la ubicación. Dado que no se especifica su estructura, se opta por no representarlo como un atributo compuesto. Se espera un formato VARCHAR.
   - **direccion**: es la dirección de la ubicación donde se lleva a cabo el evento. Se espera un formato VARCHAR.
   - **localidad**: es la localidad donde se realiza el evento. Uno podría haber interpretado el atributo localidad como un atributo derivado de direccion si en este último se incluyese la localidad de la ubicación. Se ha optado por separarlos y escribirlos independientemente. Se espera un formato VARCHAR.
   - **aforo**: es el número de plazas disponibles de las instalaciones donde se lleva a cabo el evento. Se espera un formato SMALLINT.
3. ***Actividad***: es la actividad realizada en cada evento. Posee los siguientes atributos:
   - **idActividad**: se identifica el tipo de actividad mediante una ID única en vez de por su nombre. Actuará como llave primaria de la entidad. Se espera un formato SMALLINT.
   - **tipo**: indica el tipo de actividad llevada a cabo en el evento. Se espera un formato VARCHAR.
   - **coste**: representa la suma de todos los caches de los actores que participan en la actividad. Es un atributo derivado, de ahí que se rodee con línea discontinua. Se espera un formato SMALLINT.
4. ***Actores***: esta entidad representa a los actores que llevarán a cabo una determinada actividad en el evento. Sus atributos son los siguientes:
   - **idArtista**: se identifiaca el artista con una ID creada al no poder utilizar el nombre del artista como llave primaria fiable. Se espera un formato SMALLINT.

- **nombreCompleto**: dicho atributo representa el nombre del artista. No se especifica en el enunciado que se de el nombre completo del artista. Aún así, se entiende que debe representar el nombre completo del artista, por ello se tratará como un atributo compuesto por dos atributos: el nombre del artista (nombre) y su primer apellido (apellido1). Se espera un formato VARCHAR para cada uno de los dos atributos.

- **caches**: es el pago realizado al artista por sus servicios, dependiendo de la actividad que realice. Se espera un formato SMALLINT.

- **biografia**: un pequeño resumen del artista. Se espera un formato VARCHAR.

5. ***Asistentes***: esta entidad representa a los asistentes de los eventos. Posee los siguientes atributos:

   - **idAsistente**: se identifica a cada asistente con una ID única inmutable. Se espera un formato SMALLINT.

   - **telefono**: este atributo representa los posibles números de teléfono de cada asistente. Puesto que cada asistente puede tener múltiples números de teléfono, se tratará como un atributo multivaluado, por lo que atendiendo a la primera forma normal (1FN) se tendrá que convertir en una tabla. Se espera un formato SMALLINT.

   - **email**: representa los emails de contacto de los asistentes. Puesto que cada asistente puede tener un email o varios de contacto, se tratará como un atributo multivaluado, y por ello, debe convertirse a una tabla debido a la primera forma normal (1FN). Se espera un formato VARCHAR.

   - **nombre**: es el nombre del asistente. Forma parte del atributo compuesto nombreCompleto. Se espera un formato VARCHAR.

   - **appelido1**: es el primer apellido del asistente. Forma parte del atributo compuesto nombreCompleto. Se espera un formato VARCHAR.

   - **appelido2**: es el segundo apellido del asistente. Forma parte del atributo compuesto nombreCompleto. Se espera un formato VARCHAR.

Todas las entidades mencionadas arriba se representarán mediante tablas a la hora de crear la base de datos. Vayamos ahora a tratar las relaciones, cardinalidades y rangos de las relaciones entre las diferentes entidades:

1. **En**: indica la ubicación donde se realizará el evento. Relaciona las entidades ***Evento*** y ***Ubicación***. Para este caso, el rango de la entidad *Evento* es de 1 a N (uno o varios eventos pueden tener lugar en una misma ubicación), mientas que el rango de *Ubicación* es de 1 a 1 (un evento tiene una única ubicación). Entonces, la cardinalidad de la relación es N:1, apuntando la flecha hacia *Ubicación*.

2. **Realiza**: indica la actividad que se realizará en el evento. Relaciona las entidades ***Evento*** y ***Actividad***. Para este caso, el rango de la entidad *Evento* es 1:N (uno o varios eventos pueden tener una determinada actividad), mientras que el rango de *Actividad* será 1:1 (en un evento se realiza una única actividad).

Entonces, la cardinalidad es N:1, con la flecha apuntando a *Actividad*.

3. **Reparto**: indica los actores que actúan en cierta actividad. Relaciona las entidades ***Actividad*** y ***Actores***. El rango de *Actividad* será 1:N (un actor puede poder realizar una o varias actividades), mientras que el rango de *Actores* es de 1:N (una actividad puede ser realizada por uno o varios actores), por lo que conduce a una cardinalidad de la relación de N:N. Esto permite que dicha relación, a la hora de crear la base de datos, se traduzca en una tabla por si sola.

4. **CompranEntradas**: relación que vincula las entidades ***Evento*** y ***Asistentes***. El rango de *Eventos* aquí es de 1:N (los asistentes pueden comprar entradas de uno o varios eventos), mientras que el rango de *Asistentes* es de 0:N (al evento pueden ir ninguno o varios asistentes), por lo que la cardinalidad de la relación es de N:N. Esto se traduce en poder representar la relación mediante una tabla.

2. Modelo relacional

En esta sección se convertirán las entidades, relaciones y atributos anteriormente descritos en tablas según como convenga. Para las entidades, se obtendrá lo siguiente:

1. ***Evento***: sus atributos son idEvento que actuará como llave primaria, nombreEvento, descripcion, hora, fecha y precioEntrada. Por otra parte, debido a la relación *En* con la entidad *Ubicación* de cardinalidad N:1 la llave primaria de esta última (idPueblo) se introducirá en *Eventos* como llave externa o secundaria, pasando a ser un atributo más. Por otra parte, debido a la relación *Realiza* de cardinalidad N:1, idActividad (su llave primaria) pasa a ser atributo de la primera como llave externa. Como consecuencia, el paso de la entidad *Evento* a tabla se traduce en lo siguiente:

EVENTO(idEvento, nombreEvento, descripcion, hora, fecha, precioEntrada, *idActividad*, *idPueblo*)

2. ***Ubicación***: posee los atributos idPueblo como llave primaria, precioAlquiler, nombre, caracteristicas, direccion, localidad y aforo. La única relación que posee con otra entidad es de cardinalidad N:1 apuntando a *Ubicación*, entonces no se debe añadir ningún atributo de las otras entidades. Se traduce en lo siguiente:

UBICACION(idPueblo, precioAlquiler, nombre, caracteristicas, direccion, localidad, aforo)

3. ***Actividad***: sus atributos son idActividad como llave primaria, coste como atributo derivado y tipo. Posee dos relaciones: una con *Evento* y otra con *Actores*. La primera es de cardinalidad N:1 hacia *Actividad*, entonces dicha entidad no debe poseer ninguna llave secundaria de *Evento*. Mientras, la segunda es de cardinalidad N:N, entonces formará de por si una tabla, sin necesidad de incluir una llave secundaria en *Actividad* procedente de *Actores*. Se obtiene lo siguiente:

ACTIVIDAD(idActividad, tipo, coste)

4. ***Actores***: posee los atributos idArtista como llave primaria, biografia, caches y nombreCompleto, atributo compuesto por los atributos de nombre y apellido1. Por los mismos argumentos explicados en la entidad anterior, no se debe incluir ninguna llave secundaria de *Actividad* mediante la relación *Reparto*. Se obtiene lo siguiente:

ACTORES(idArtista, nombre, apellido1, biografia, caches)

5. ***Asistentes***: posee como atributos idAsistente como llave primaria, nombreCompleto (atributo compuesto por los atributos nombre, apellido1 y apellido2) y los atributos multivaluados telefono y email. Estos dos últimos, respetando la primera forma normal, se convertirán en tablas de por si más abajo, con el fin de evitar redundancias. La entidad solo posee una única relación llamada *CompranEntradas* con *Evento*, pero es de cardinalidad N:N, entonces no es necesario incluir ninguna clave externa procedente de *Evento*. El resultado es el siguiente:

ASISTENTES(idAsistente, nombre, apellido1, apellido2)

En cuanto a las relaciones, transformaremos a tablas aquellas con cardinalidad N:N, es decir, las relaciones **Reparto** y **CompranEntradas**:

1. **Reparto**: no posee ningún atributo, pero tomará como llaves primarias las llaves primarias de *Actividad* y *Actores* y las utilizará como llaves externas también. Recibirán el nombre, respectivamente, de codActividad y codArtista. Se obtiene lo siguiente:

REPARTO(*codActividad*, *codArtista*)

2. **CompranEntradas**: no posee ningún atributo, pero tomará como claves primarias las claves primarias de *Evento* y *Asistentes* y las utilizará como claves externas también. Recibirán el nombre, respectivamente, de codEvento y codAsistente. Se obtiene lo siguiente:

COMPRANENTRADAS(*codEvento*, *codAsistente*)

Finalmente, traduciremos a tablas aquellos atributos multivaluados:

1. **telefono**: es uno de los atributos multivaluados de la entidad *Asistentes*. Tomará la llave primaria de *Asistentes* como llave primaria propia llamándose codAsistente, usándose como llave externa a la vez. Tendrá además un atributo adicional que indique el número de teléfono, llamado numTelefono. Se obtiene lo siguiente:

TELEFONO(*codAsistente*, numTelefono)

2. **email**: es el otro atributo multivaluado de la entidad *Asistentes*. Tomará la llave primaria de *Asistentes* como llave primaria propia llamándose codAsistente, usándose como llave externa a la vez. Tendrá además un atributo adicional que muestre la dirección email del asistente, llamado dirEmail. Se obtiene lo siguiente:

EMAIL(*codAsistente*, dirEmail)

En resumen, todas las tablas se representan en la siguiente imagen con sus respectivas conexiones:

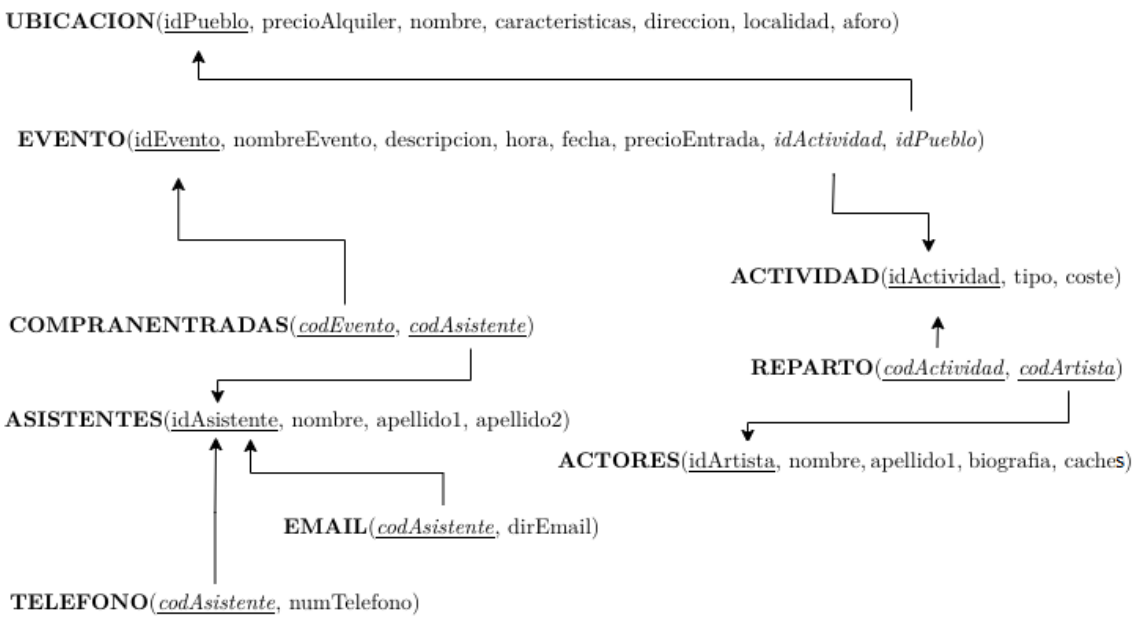

Figura 2: *Diagrama relacional.*

Fíjese en que tan solo se ha mencionado la primera forma normal (1FN). Uno puede comprobar que las demás formas normales (segunda y tercera) también se cumplen en las tablas creadas sobre el hecho de que cada atributo está vinculado a las claves primarias de la tabla y no de forma transitiva a través de otro campo.

3. Implementación de la Base de Datos con MySQL

A continuación, se pretende implementar la base de datos anteriormente diseñada en SQL mediante el software de MySQL. A continuación, se procede a mostrar las *queries* de creación y manipulación de los datos de la base de datos:

```
/* ----------------------------------------------------------------------------------------------------
Nombre del autor: Bartolomé Metre Fons
Nombre de la base de datos: ArteVidaCultural
----------------------------------------------------------------------------------------------------*/
/* ----------------------------------------------------------------------------------------------------
Definición de la estructura de la base de datos
----------------------------------------------------------------------------------------------------*/
-- Se crea la database.
DROP DATABASE IF EXISTS ArteVidaCultural;
CREATE DATABASE ArteVidaCultural;
USE ArteVidaCultural;

-- Se crea la tabla ACTIVIDAD.
DROP TABLE IF EXISTS ACTIVIDAD;
CREATE TABLE ACTIVIDAD(
```

```
idActividad SMALLINT PRIMARY KEY,
tipo VARCHAR(30),
coste DECIMAL(10,2)
); -- Se introducen decimales por si el coste incluye céntimos.

-- Se crea la tabla UBICACIÓN.
DROP TABLE IF EXISTS UBICACION;
CREATE TABLE UBICACION(
idPueblo SMALLINT PRIMARY KEY,
precioAlquiler SMALLINT,
nombre VARCHAR(30) NOT NULL,
caracteristicas VARCHAR(100) DEFAULT 'Sin características notables',
direccion VARCHAR(30),
localidad VARCHAR(30),
aforo SMALLINT
);

-- Se crea la tabla EVENTO.
DROP TABLE IF EXISTS EVENTO;
CREATE TABLE EVENTO(
idEvento SMALLINT PRIMARY KEY,
nombreEvento VARCHAR(30),
descripcion VARCHAR(100) DEFAULT 'Sin descripción', -- Por si no se ofrece una descripción.
hora TIME,
fecha DATE,
precioEntrada DECIMAL(10,2),
idActividad SMALLINT,
idPueblo SMALLINT,
FOREIGN KEY (idActividad) REFERENCES ACTIVIDAD(idActividad)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (idPueblo) REFERENCES UBICACION(idPueblo)
ON UPDATE CASCADE ON DELETE CASCADE
);

-- Se crea la tabla ACTORES.
DROP TABLE IF EXISTS ACTORES;
CREATE TABLE ACTORES(
idArtista SMALLINT PRIMARY KEY,
nombre VARCHAR(15) NOT NULL,
apellido1 VARCHAR(15),
biografia VARCHAR(100) DEFAULT 'Sin biografía conocida', -- Por si no se ofrece una biografía.
caches DECIMAL(10,2)
); -- Por si el cache del artista incluye céntimos.

-- Se crea la tabla REPARTO.
DROP TABLE IF EXISTS REPARTO;
CREATE TABLE REPARTO(
codActividad SMALLINT,
codArtista SMALLINT,
```

```
PRIMARY KEY(codActividad, codArtista),
FOREIGN KEY (codActividad) REFERENCES ACTIVIDAD(idActividad)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (codArtista) REFERENCES ACTORES(idArtista)
ON UPDATE CASCADE ON DELETE CASCADE
);

-- Se crea la tabla ASISTENTES.
DROP TABLE IF EXISTS ASISTENTES;
CREATE TABLE ASISTENTES(
idAsistente SMALLINT PRIMARY KEY,
nombre VARCHAR(10) NOT NULL, -- No puede existir una id sin un nombre asociado.
apellido1 VARCHAR(10),
apellido2 VARCHAR(10)
);

-- Se crea la tabla COMPRANENTRADAS.
DROP TABLE IF EXISTS COMPRANENTRADAS;
CREATE TABLE COMPRANENTRADAS(
codEvento SMALLINT,
codAsistente SMALLINT,
PRIMARY KEY(codEvento, codAsistente),
FOREIGN KEY (codEvento) REFERENCES EVENTO(idEvento)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (codAsistente) REFERENCES ASISTENTES(idAsistente)
ON UPDATE CASCADE ON DELETE CASCADE
);

-- Se crea la tabla EMAIL.
DROP TABLE IF EXISTS EMAIL;
CREATE TABLE EMAIL(
codAsistente SMALLINT PRIMARY KEY,
dirEmail VARCHAR(30),
FOREIGN KEY (codAsistente) REFERENCES ASISTENTES(idAsistente)
ON UPDATE CASCADE ON DELETE CASCADE
);

-- Se crea la tabla TELEFONO
DROP TABLE IF EXISTS TELEFONO;
CREATE TABLE TELEFONO(
codAsistente SMALLINT PRIMARY KEY,
numTelefono INT,
FOREIGN KEY (codAsistente) REFERENCES ASISTENTES(idAsistente)
ON UPDATE CASCADE ON DELETE CASCADE
);

/* Se procede a continuación a aplicar diversas restricciones
sobre los atrinbutos de las tablas.*/
```

```
-- El coste de la actividad tiene que ser positivo.
ALTER TABLE ACTIVIDAD ADD CONSTRAINT CHECK (coste > 0);
-- El precio del alquiler tiene que ser positivo.
ALTER TABLE UBICACION ADD CONSTRAINT CHECK (precioAlquiler > 0);
-- El precio de la entrada debe ser positivo.
ALTER TABLE EVENTO ADD CONSTRAINT CHECK (precioEntrada > 0);
-- El cache del actor tiene que ser positivo.
ALTER TABLE ACTORES ADD CONSTRAINT CHECK (caches > 0);

/* Ahora, se procede a aplicar restricciones a los rangos
de las entidades.*/
/* La primera, hay que asegurar que por cada evento hay una y solo
una ubicación asociada.*/

DROP TRIGGER IF EXISTS unEventoUnaUbicacion;

DELIMITER //
CREATE TRIGGER unEventoUnaUbicacion
BEFORE INSERT
ON EVENTO
FOR EACH ROW
BEGIN
DECLARE numLoc INT; -- En esta variable se guarda el número de veces que sale una ubicación.
SELECT COUNT(*) INTO numLoc
    FROM UBICACION ubi
    JOIN EVENTO e ON e.idPueblo = ubi.idPueblo -- JOIN y INNER JOIN cumplen la misma función.
    WHERE e.idPueblo = NEW.idPueblo;
    IF  numloc > 1 THEN -- Si el número de ubicaciones por evento es superior a 1,
-- se devuelve un error con mensaje.
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Más de una ubicación por evento';
    END IF;
END; //

DELIMITER ;

/*Ahora, otro trigger para asegurarse de que en un evento solo se realiza una única actividad.*/

DROP TRIGGER IF EXISTS unEventoUnaActividad

DELIMITER //
CREATE TRIGGER unEventoUnaActividad
BEFORE INSERT
ON EVENTO
FOR EACH ROW
BEGIN
DECLARE num_actividades INT;
    SELECT COUNT(*) INTO num_actividades
    FROM ACTIVIDAD act
```

```
    JOIN EVENTO e ON e.idActividad = act.idActividad
    WHERE e.idActividad = NEW.idActividad;
    IF num_actividades > 1 THEN -- Si el número de actividades por evento es superior a 1,
-- se devuelve un error.
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'No puede haber más de una actividad por evento';
    END IF;
END;//

DELIMITER;

/*Por otra parte, se desea que el correo electrónico
proporcionado por el asistente tenga un formato correcto,
tal que incluya una arroba. Mediante un trigger
se asegura que dicha condición se cumpla */


DROP TRIGGER IF EXISTS formatEmail;

DELIMITER //
CREATE TRIGGER formatEmail
BEFORE INSERT
ON EMAIL
FOR EACH ROW
BEGIN
IF NEW.dirEmail NOT LIKE '%@%' THEN -- Se devuelve un error si el email no contiene @.
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Formato de email incorrecto';
    END IF ;
END;//

DELIMITER ;


/*-----------------------------------------------------------------------------
Trigger
Inserción de datos
-----------------------------------------------------------------------------*/


/* Se escribe a continuación el trigger que actualizará automáticamente el coste de la actividad si el
cache de los artistas se ve modificado.*/

DROP TRIGGER IF EXISTS actualizaCostes

DELIMITER \\
CREATE TRIGGER actualizaCostes
AFTER UPDATE
ON ACTORES
```

```
FOR EACH ROW
BEGIN
    UPDATE ACTIVIDAD a -- Se actualiza la tabla ACTIVIDAD si se modifica el cache de su artista.
    SET a.coste = (
        SELECT SUM(act.caches)
        FROM ACTORES act
JOIN REPARTO re ON re.codArtista = act.idArtista
        WHERE a.idActividad = re.codActividad
        );
END \\

DELIMITER ;

/*Ahora, se procede a poblar las tablas de datos. Uno puede proceder aquí
mediante la creación de un archivo .csv y cargarlo directamente, o bien
insertarlos manualmente. Puesto que dichos datos se utilizarán con el fin
de comprobar la funcionalidad de la base de datos, se prefiere insertarlos
manualmente para un reducido número de entradas de datos.*/

INSERT INTO UBICACION VALUES
(101, 2000, 'Pueblo Playa', 'Espacio abierto, escenario al aire libre',
 'Calle Mar, 5', 'Playa Azul', 500),
(102, 1500, 'Cine Independiente', 'Auditorio con pantallas de alta definición',
 'Avenida Central, 10', 'Ciudad del Cine', 300),
(103, 500, 'Centro Cultural', 'Salón para conferencias y competiciones',
 'Calle Ajedrez, 7', 'Villa del Rey', 100),
(104, 800, 'Galería Central', 'Galería con exhibiciones de arte contemporáneo',
 'Calle del Arte, 15', 'Capital de la Cultura', 150),
(105, 1200, 'Teatro Principal', 'Teatro con escenario y capacidad para obras grandes',
 'Plaza Central, 1', 'Ciudad Teatro', 400),
(106, 300, 'Parque Urbano', 'Área abierta para eventos deportivos y recreativos',
 'Avenida Verde, 2', 'Ciudad Verde', 1000);

INSERT INTO ACTIVIDAD VALUES
(1, 'Música en Vivo', 5000),
(2, 'Proyección de Películas', 3000),
(3, 'Torneo de Ajedrez', 2000),
(4, 'Exposición de Arte', 1500),
(5, 'Obra de Teatro', 4000),
(6, 'Competencia Deportiva', 1000);

INSERT INTO EVENTO VALUES
(1, 'Concierto de Verano', 'Un concierto de música en vivo al aire libre', '20:00:00',
 '2024-06-15', 50, 1, 101),
(2, 'Festival de Cine', 'Proyecciones de cine independiente', '18:30:00',
 '2024-09-10', 20, 2, 102),
(3, 'Torneo de Ajedrez', 'Competencia de ajedrez abierta al público', '10:00:00',
 '2024-08-05', 10, 3, 103),
(4, 'Exposición de Arte', 'Exhibición de arte contemporáneo local', '12:00:00',
```

```
 '2024-07-22', 15, 4, 104),
(5, 'Obra de Teatro', 'Interpretación de la obra clásica "Hamlet"', '19:00:00',
 '2024-11-15', 30, 5, 105),
(6, 'Carrera de 5K', 'Competencia de atletismo en el centro de la ciudad', '08:00:00',
 '2024-09-01', 5, 6, 106);

INSERT INTO ACTORES VALUES
(1, 'Carlos', 'Vives', 'Cantante colombiano de vallenato y pop.', 5000),
(2, 'Ana', 'Sofía', 'Actriz galardonada en el cine de arte.', 3000),
(3, 'Luis', 'Pardo', 'Gran maestro de ajedrez nacional.', 2000),
(4, 'Marta', 'Cano', 'Artista de renombre en pintura contemporánea.', 1500),
(5, 'Teatro Ensamble','', 'Compañía de teatro experimental.', 4000),
(6, 'Juan', 'Méndez', 'Atleta profesional en carreras de fondo.', 1000);

INSERT INTO ASISTENTES VALUES
(1, 'Pedro', 'Martínez', 'García'),
(2, 'Laura', 'Sánchez', 'Rodríguez'),
(3, 'Miguel', 'López', 'Pérez'),
(4, 'Lucía', 'Gómez', 'Fernández'),
(5, 'Javier', 'Martín', 'Hernández'),
(6, 'María', 'Ruiz', 'González'),
(7, 'Fernando', 'Díaz', 'López'),
(8, 'Sara', 'Morales', 'Vega'),
(9, 'Raúl', 'Torres', 'Serrano'),
(10, 'Ana', 'Ramos', 'Molina'),
(11, 'Juan', 'Mendez', 'Gomila');

INSERT INTO REPARTO VALUES
(1, 1),  -- Carlos Vives participa en Música en Vivo.
(2, 2),  -- Ana Sofía actúa en Proyección de Películas.
(3, 3),  -- Luis Pardo en el Torneo de Ajedrez.
(4, 4),  -- Marta Cano en la Exposición de Arte.
(5, 5),  -- Teatro Ensamble presenta la Obra de Teatro.
(6, 6);  -- Juan Méndez participa en la Carrera de 5K.

INSERT INTO COMPRANENTRADAS VALUES
(1, 1),  -- Pedro compra entrada para Concierto de Verano.
(1, 2),  -- Laura compra entrada para Concierto de Verano.
(2, 3),  -- Miguel compra entrada para Festival de Cine.
(3, 4),  -- Lucía compra entrada para Torneo de Ajedrez.
(4, 5),  -- Javier compra entrada para Exposición de Arte.
(5, 6),  -- María compra entrada para Obra de Teatro.
(6, 7),  -- Fernando compra entrada para Carrera de 5K.
(2, 8),  -- Sara compra entrada para Festival de Cine.
(3, 9),  -- Raúl compra entrada para Torneo de Ajedrez.
(2, 9),  -- Miguel compra entrada para Torneo de Ajedrez.
(4, 10), -- Ana compra entrada para Exposición de Arte.
(4, 11); -- Juan compra entrada para Exposición de Arte.
```

```
INSERT INTO TELEFONO VALUES
(1, 123456789),  -- Teléfono de Pedro.
(2, 987654321),  -- Teléfono de Laura.
(3, 456123789),  -- Teléfono de Miguel.
(4, 321654987),  -- Teléfono de Lucía.
(5, 741852963),  -- Teléfono de Javier.
(6, 852963741),  -- Teléfono de María.
(7, 963852741),  -- Teléfono de Fernando.
(8, 147258369),  -- Teléfono de Sara.
(9, 369258147),  -- Teléfono de Raúl.
(10, 258147369), -- Teléfono de Ana.
(11, 257147369); -- Teléfono de Juan.

INSERT INTO EMAIL VALUES
(1, 'pedromartinez@example.com'),  -- Email de Pedro.
(2, 'laurasanchez@example.com'),   -- Email de Laura.
(3, 'miguellopez@example.com'),    -- Email de Miguel.
(4, 'luciagomez@example.com'),     -- Email de Lucía.
(5, 'javiermartin@example.com'),   -- Email de Javier.
(6, 'mariaruiz@example.com'),      -- Email de María.
(7, 'fernandodiaz@example.com'),   -- Email de Fernando.
(8, 'saramorales@example.com'),    -- Email de Sara.
(9, 'raultorres@example.com'),     -- Email de Raúl.
(10, 'anaramos@example.com'),      -- Email de Ana.
(11, 'juanmendez@example.com');    -- Email de Juan.

/*-----------------------------------------------------------------------------
Consultas, modificaciones, borrados y vistas con enunciado
-----------------------------------------------------------------------------*/

/* 1. Muestra cuál ha sido el evento con más recaudación de dinero.*/
SET @idmost = (
SELECT codEvento FROM COMPRANENTRADAS GROUP BY codEvento
ORDER BY COUNT(*) DESC
    LIMIT 1
    ); -- Se almacena en una variable el id del evento con más frecuencia.

SET @num_max =  (
SELECT COUNT(*) AS contaje FROM COMPRANENTRADAS GROUP BY codEvento
ORDER BY contaje DESC
    LIMIT 1
    ); -- En otra variable las veces que ha aparecido.

SELECT idEvento AS ID_Evento, nombreEvento AS Nombre_evento, precioEntrada * @num_max AS Recaudacion
FROM EVENTO WHERE idEvento = @idmost; -- Este ejercicio se podría haber implementado de otra forma,
  -- si bien se ha puesto aquí en práctica la creación de variables.

/* 2. Busca la actividad que más dinero ha generado. Para ello se crea una view.*/
SELECT a.tipo AS Actividad, SUM(e.precioEntrada) AS Recaudacion_total
```

```
FROM EVENTO e
JOIN COMPRANENTRADAS ce ON e.idEvento = ce.codEvento
JOIN ACTIVIDAD a ON e.idActividad = a.idActividad
GROUP BY a.idActividad
ORDER BY Recaudacion_total DESC
LIMIT 1;

/* 3. Se actualiza el cache de un actor para comprobar que se modifica
automáticamente el precio de la actividad.*/
SET SQL_SAFE_UPDATES = 0; -- Necesario para poder actualizar las entradas en la tabla.
UPDATE ACTORES SET caches = 2000 WHERE idArtista = 1; -- Se actualiza el cache del artista con id=1.
SELECT a.coste
FROM ACTIVIDAD a
JOIN REPARTO re ON a.idActividad = re.codActividad
JOIN ACTORES ac ON ac.idArtista = re.codArtista
WHERE ac.idArtista = 1; -- Devolvemos el coste de la actividad donde participa el anterior artista.

/*4. Muestra la recaudación por mes.*/
SELECT MONTH(fecha) AS mes, COUNT(*) AS num
FROM EVENTO
GROUP BY mes
ORDER BY num DESC;

/*5. Consulta los eventos que ocurrirán en una
localidad específica (por ejemplo, "Pueblo Playa"). */
SELECT idEvento AS 'ID Evento', nombreEvento AS 'Nombre Evento'
FROM EVENTO e
JOIN UBICACION ubi ON ubi.idPueblo = e.idPueblo
WHERE ubi.nombre = 'Pueblo Playa';

/*6 Crea una vista de ingresos y asistencia por actividad
 y ubicación de 'Cine Independiente'.*/
CREATE VIEW IngresosActividadUbicacion AS
SELECT ubi.nombre AS ubicacion, act.tipo AS actividad, e.nombreEvento as evento,
COUNT(c.codAsistente) AS total_asistentes, SUM(e.precioEntrada) AS total_recaudado
FROM UBICACION ubi
JOIN EVENTO e ON e.idPueblo = ubi.idPueblo
JOIN ACTIVIDAD act ON act.idActividad = e.idActividad
JOIN COMPRANENTRADAS c ON c.codEvento = e.idEvento
GROUP BY ubi.nombre, act.tipo, e.nombreEvento
ORDER BY total_recaudado;

SELECT * FROM IngresosActividadUbicacion; -- Mostramos la View.

/*7. Encuentra el gasto por asistente en eventos.*/
SELECT a.idAsistente, a.nombre, a.apellido1, a.apellido2,
SUM(e.precioEntrada) as Gasto_total
FROM ASISTENTES a
JOIN COMPRANENTRADAS ce ON ce.codAsistente = a.idAsistente
```

```
JOIN EVENTO e ON e.idEvento = ce.codEvento
GROUP BY a.idAsistente
ORDER BY Gasto_total;

 /*8. Crea un procedimiento para registrar la asistencia
 a un evento, y si el número de asistentes supera
 el aforo de la ubicación, indicamos que ya no hay
 disponibilidad para comprar más entradas.*/

 DROP PROCEDURE IF EXISTS registrarVentaEntrada;

 DELIMITER //
 CREATE PROCEDURE registrarVentaEntrada(
IN eventoId INT,
IN asistenteId INT
 )
 BEGIN
DECLARE entradasCompradas INT; -- Se registra en una variable el número de entradas compradas.
    DECLARE aforoUbicacion INT; -- Se registra en una variable el aforo de la ubicación.

    SELECT COUNT(*)
    INTO entradasCompradas  -- En la primera variable se almacena el resultado de la query.
    FROM COMPRANENTRADAS c
    WHERE c.codEvento = eventoId;

    SELECT ubi.aforo
    INTO aforoUbicacion -- En la segunda variable se almacena el resultado de la query.
    FROM UBICACION ubi
    JOIN EVENTO e ON e.idPueblo = ubi.idPueblo
    WHERE e.idEvento = eventoId;

    IF entradasCompradas < aforoUbicacion THEN
INSERT INTO COMPRANENTRADAS(codEvento, codAsistente)
        VALUES (eventoId, asistenteId); -- Si hay sitio aún, se añade la entrada a la tabla.
        SELECT 'Venta registrada con éxito';
ELSE
SELECT 'No hay más espacio disponible para nuevas entradas'; -- Sino, se devuelve este mensaje.
END IF;
END //

DELIMITER ;

CALL RegistrarVentaEntrada(4,7); -- Se pone a prueba el procedimiento
     -- (Fernando acude a exposición de arte).

/*9. Consulta la ubicación que más ingresos ha generado. */
SELECT ubi.idPueblo AS IDPueblo, ubi.nombre AS Nombre, SUM(e.precioEntrada) AS Recaudacion
FROM UBICACION ubi
JOIN EVENTO e ON e.idPueblo = ubi.idPueblo
```

```
JOIN COMPRANENTRADAS ce ON e.idEvento = ce.codEvento
GROUP BY ubi.idPueblo
ORDER BY Recaudacion DESC -- Se ordenan los resultados y tan solo nos quedamos con el primero.
LIMIT 1;

/*10. Consulta los artistas más populares
según el número de eventos en los que participan.*/

SELECT a.idArtista AS IDArtista, a.nombre AS NombreArtista, COUNT(e.idEvento) AS Veces
FROM  ACTORES a
JOIN REPARTO re ON a.idArtista = re.codArtista
JOIN ACTIVIDAD act ON re.codActividad = act.idActividad
JOIN EVENTO e ON act.idActividad = e.idActividad
GROUP BY a.idArtista
ORDER BY Veces DESC;
```

4. Bibliografia

Se han consultado las siguientes fuentes a la hora de realizar el presente trabajo:

- Apuntes del módulo *Bases de Datos relacionales* por Dr.Isabel Riomoros.

- *Trigger Syntax and Examples*, 2024.*dev.mysql.com*. Consultado desde: https://dev.mysql.com/doc/refman/8.4/en/trigger-syntax.html