Bases de Datos: IIC2413
Grupo 13

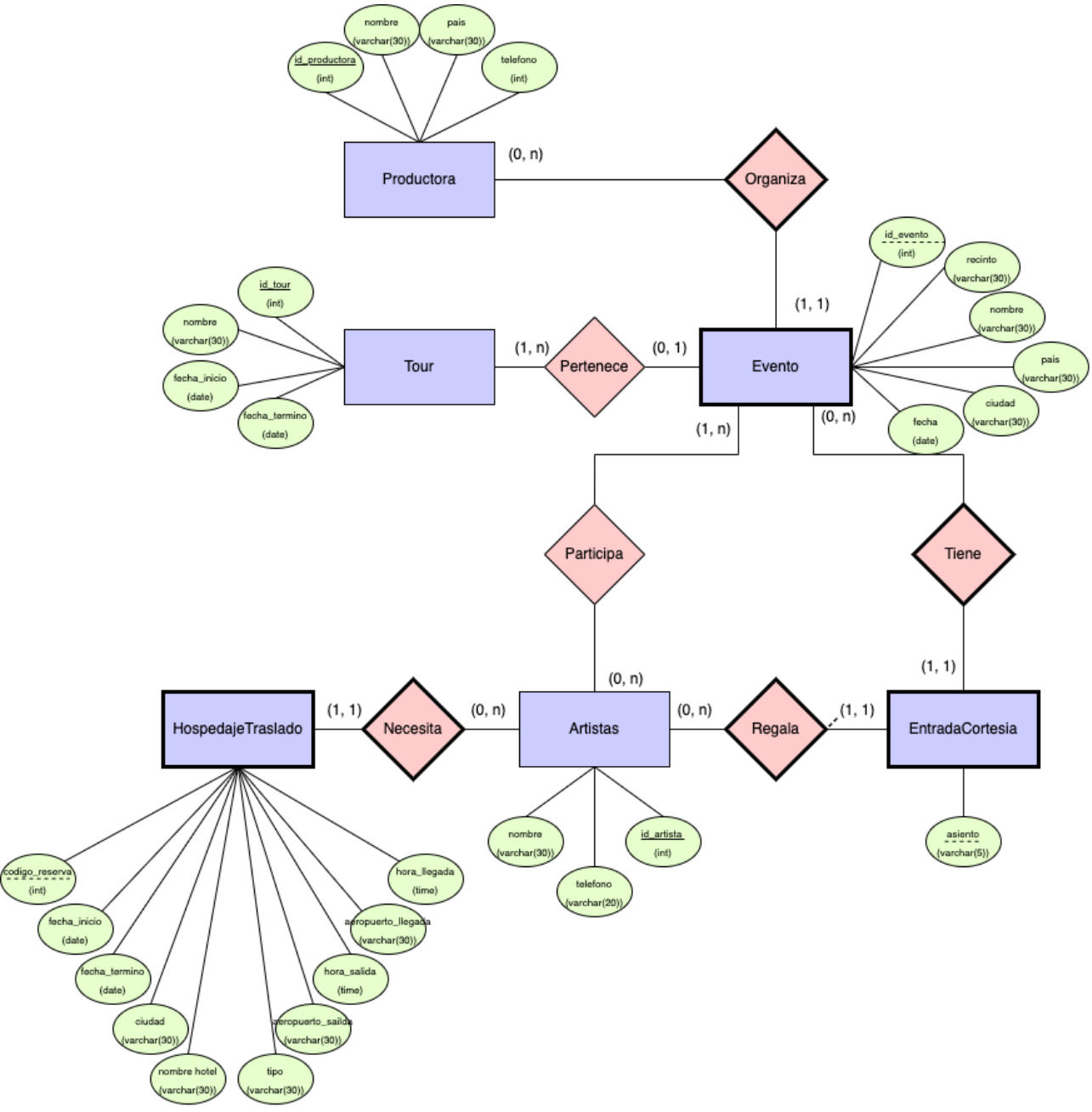
Entrega 2:
“Diseño de una Aplicación”

Integrantes
Borja Marquez de la Plata M.
Martín Caldentey L.

Indice

Diagrama ER

A continuación, se presenta el diagrama construido para modelar el caso presentado con las entidades y sus relaciones.



Esquema Relacional

Para modelar los procesos de los artistas, se crearon las siguientes relaciones a partir de los datos entregados. Algunas tablas incluyen llaves foráneas que referencian atributos de otra relación de la base de datos:

1. **Relación de los atributos de un artista:**

   Artistas(id_artista: **int PRIMARY KEY**, nombre_artistico: **varchar(30) NOT NULL**, inicio_carrera: **date NOT NULL,** numero_contacto: **varchar(15) NOT NULL**)

2. **Relación de los atributos de un evento:**

   Eventos(id_evento: **int PRIMARY KEY**, evento: **varchar(30) NOT NULL,** recinto: **varchar(200) NOT NULL**, ciudad: **varchar(30) NOT NULL,** pais: **varchar(30) NOT NULL,** fecha_inicio: **date NOT NULL**, id_productora: **INT NOT NULL, FOREIGN KEY(**id_productora**) REFERENCES** Productoras(id_productora))

3. **Relación de los atributos de un tour:**

   Tour(id_tour: **int PRIMARY KEY**, nombre: **varchar(30) NOT NULL**, fecha_inicio: **date NOT NULL**, fecha_termino: **date NOT NULL, CHECK(**fecha_termino >= fecha_inicio))

   *Aquí es necesario hacer un CHECK sobre la fecha, ya que un tour no puede terminar antes de partir.

4. **Relación de los atributos de una productora:**

   Productoras(id_productora: **int PRIMARY KEY**, nombre: **varchar(100) NOT NULL**, pais: **varchar(30) NOT NULL**, numero_contacto: **varchar(30) NOT NULL**)

5. **Los artistas pueden otorgar entradas de cortesía, de sus propios conciertos, a personas cercanas. Se presenta la relación de estas:**

   Entradas_cortesia(id_evento: **int NOT NULL**, asiento: **varchar(4),** id_artista: **int NOT NULL,** id_productora **int NOT NULL, FOREIGN KEY(**id_evento**) REFERENCES** Eventos(id_evento), **FOREIGN KEY**(id_artista) **REFERENCES** Artistas(id_artista), **FOREIGN KEY**(id_productora) **REFERENCES** productoras(id_productora))

6. **Relación de atributos del hospedaje y traslado de un artista.**
   Notar que, si un traslado es de tipo aéreo o mixto tendrá entradas para aeropuerto_salida, aeropuerto_llegada, hora_salida y hora_llegada, pero si es tipo terrestre esas entradas serán NULL**:**

   Hospedajes_y_reservas(codigo_reserva: **varchar(30) PRIMARY KEY**, id_artista: **int NOT NULL,** fecha_inicio: **date NOT NULL**, fecha_termino: **date NOT NULL**, lugar: **varchar(30) NOT NULL**, nombre_hotel: **varchar(30) NOT NULL,** tipo_traslado: **varchar(10) NOT NULL,** aeropuerto_salida: **varchar(60),** hora_salida: **time**, aeropuerto_llegada: **varchar(60)**, hora_llegada: **time, CHECK(**fecha_termino >= fecha **inicio), UNIQUE(**codigo_reserva**), FOREIGN KEY(**id_artista**) REFERENCES** Artistas(id_artista))

   *Acá se debe hacer un CHECK sobre las fechas, ya que una reserva no puede terminar antes de partir. También, los códigos de reservas deben ser UNIQUE, ya que son códigos que referencian solo una reserva.

Adicionalmente, se creó la siguiente tabla auxiliar con el objetivo de cumplir con BCNF para reducir la redundancia de eventos con múltiples artistas, lo cual se explica a continuación:

1. *Artista_en_evento*(id_productora: **int**, id_evento: **int**, id_artista: **int**)

Justificación del Modelo

Para que el modelo cumpliera con BCNF se estudiaron las dependencias funcionales de cada caso. Dentro de artistas no existían dependencias, ya que estos solo pueden tener un número de teléfono y una fecha de inicio de su carrera. Para las productoras tampoco se encontró, ya que estas están en un solo país y tienen un solo número de contacto. Este es el caso también de los tours. Con los eventos nos encontramos con el primer caso de consulta funcional. Estos pueden tener varios artistas para el mismo evento, como en este ejemplo:

id_evento	Evento	Recinto	Artista	Ciudad	Pais	Fecha de incio	id_productora
0	BZRP Tour	Pabellon Max Schmeling	Paulo Londra	BERLIN	ALEMANIA	09-05-20	227
7	BZRP Tour	Pabellon Max Schmeling	Snow	BERLIN	ALEMANIA	09-05-20	227
53	BZRP Tour	Pabellon Max Schmeling	Bizarrap	BERLIN	ALEMANIA	09-05-20	227
78	BZRP Tour	Pabellon Max Schmeling	Cazzu	BERLIN	ALEMANIA	09-05-20	227
178	BZRP Tour	Pabellon Max Schmeling	Nicki Nicole	BERLIN	ALEMANIA	09-05-20	227

Toda la información del evento se encuentra repetida aparte del artista, tenemos dependencia funcional. Para esto se creó una tabla auxiliar llamada artista en evento, la cual nos dice que artistas van a qué evento. Lo anterior nos permite eliminar la columna artistas de eventos y así no tenemos el problema presentado. En el caso de las entradas de cortesía, notamos que cada una de estas es única por evento y por artista. La última tabla de hospedajes y transportes también es única para cada reserva, y por lo tanto ya está normalizada.
Dada la modificación agregada para los eventos, se puede llegar a un modelo normalizado sin dependencias funcionales.

Carga de Datos

Antes de cargar los datos se realizó una limpieza extensa de esta utilizando la biblioteca Pandas de Python. Se eliminaron entradas duplicadas, agregaron los ids y se creó la tabla auxiliar Artista en Evento. Se arreglaron los datos incorrectos, como las fechas que no existían, pasándolas a la más cercana válida dentro del mismo mes. Para verificar que no se pierda información, se compararon los archivos finales obtenidos con los iniciales y se comprobó que no hubo pérdida de información. Un supuesto que se debió tomar fue respecto a las entradas de cortesía, ya que estas están asociadas a un evento a través del nombre de este. Sin embargo, existen instancias de eventos distintos con el mismo nombre. Entonces, a la entrada se le asignó el id del primer evento en las tablas.
Para cargar los datos en las tablas, primero se subieron los archivos al servidor usando github y luego se usó el comando COPY. Se presenta un ejemplo a continuación:

\COPY entradas_cortesia(asiento, id_artista, id_productora, id_evento) FROM '/home/grupo13/Sites/Datos Impares/Entradas Cortesia_limpio.csv' DELIMITER ',' CSV HEADER;

Consultas SQL

Para las relaciones presentadas anteriormente, se realizaron las siguientes consultas en SQL. Para las consultas que tendrán input de tipo entrada de texto se usará el comando ILIKE para permitir la búsqueda en casos de error de tipeo. En cambio, para las de tipo drop down se usará el comando LOWER para evitar tener problemas con las mayúsculas. Cual tipo de input se usará para cada consulta se explica en la sección Interfaz Web.

1. **Entregue un listado de nombre y teléfono de cada artista:**

   SELECT nombre_artistico, numero_contacto
   FROM Artistas;

2. **Dado un artista (ej. Bizarrap), entregue el número de entradas de cortesía que ha entregado:**

   SELECT artistas.nombre_artistico, **COUNT**(artistas.nombre_artistico) as entradas_de_cortesía_entregadas
   FROM entradas_cortesia **INNER JOIN** artistas **ON** entradas_cortesia.id_artista = artistas.id_artista
   WHERE LOWER(artistas.nombre_artistico) = **LOWER**('Bizarrap')
   GROUP BY artistas.nombre_artistico;

3. **Dado un artista (ej. Bizarrap), entregue los datos de su último tour (el más reciente):**

   SELECT tours.id_tour, tours.nombre, tours.fecha_inicio, tours.fecha_termino
   FROM Artistas
   INNER JOIN Artista_en_evento **ON** artistas.id_artista = artista_en_evento.id_artista
   INNER JOIN Eventos
   ON eventos.id_productora = artista_en_evento.id_productora **AND** eventos.id_evento = artista_en_evento.id_evento
   INNER JOIN Tours **ON** eventos.evento = tours.nombre
   WHERE LOWER(artistas.nombre_artistico) = **LOWER**('Bizarrap')
   ORDER BY tours.fecha_inicio **DESC**
   LIMIT 1;

4. **Dado un tour (ej. WAP), liste los países que serán visitados en dicho tour:**

   SELECT DISTINCT eventos.pais
   FROM Eventos
   WHERE LOWER(eventos.evento) = **LOWER** ('WAP');

5. **Dado un artista (ej. Bizarrap), liste todas las productoras con las que ha trabajado dicho artista:**

```
SELECT DISTINCT productoras.nombre
FROM Artistas
INNER JOIN Artista_en_evento ON artistas.id_artista = artista_en_evento.id_artista
INNER JOIN Eventos
ON eventos.id_productora = artista_en_evento.id_productora AND  eventos.id_evento = artista_en_evento.id_evento
INNER JOIN Productoras ON eventos.id_productora = productoras.id_productora
WHERE LOWER(artistas.nombre_artistico) = LOWER('Bizarrap')
```

6. **Dado un artista (ej. Bizarrap), liste todos los hoteles en los que se ha hospedado y cuantas veces se ha hospedado en cada uno (con códigos de reserva distinto, no cantidad de noches):**

```
SELECT hospedajes_y_reservas.nombre_hotel, COUNT(
hospedajes_y_reservas.nombre_hotel) as Estadias_en_hotel
FROM Artistas INNER JOIN Hospedajes_y_reservas
ON artistas.id_artista = hospedajes_y_reservas.id_artista
WHERE LOWER(artistas.nombre_artistico) = LOWER('Bizarrap')
GROUP BY hospedajes_y_reservas.nombre_hotel;
```

7. **Muestra al artista que ha entregado la mayor cantidad de entradas de cortesía:**

```
SELECT Artistas.nombre_artistico
FROM Entradas_cortesia INNER JOIN Artistas ON entradas_cortesia.id_artista = artistas.id_artista
GROUP BY Artistas.nombre_artistico
ORDER BY COUNT(artistas.nombre_artistico) DESC
LIMIT 1;
```

Interfaz Web

El landing page de la pagina creada se encuentra en el siguiente link: https://codd.ing.puc.cl/~grupo13/index.php?
Para construir la página, se tuvo que decidir cuando usar entradas de text y cuando un drop down. Basándose en como estaba redactada la consulta pedida se tomó esta decisión. Cuando se explicita que se dará el nombre de una entidad, se usó una entrada de texto, mientras que en los casos donde se dará la entidad, se usó dropdown. Por ejemplo:

1. Dado el nombre de una productora, entregue los datos del último evento que ha producido

En este caso se usó una entrada de texto para la consulta.

2. Dado un artista, entregue el número de entradas de cortesía que ha entregado

En este caso se usó una entrada de texto para la consulta.

```
SELECT artistas.nombre_artistico, COUNT(artistas.nombre_artistico) as entradas_de_cortesía_entregadas FROM entradas_cortesia INNER JOIN artistas ON entradas_cortesia.id_artista = artistas.id_artista WHERE LOWER(artistas.nombre_artistico) = LOWER() GROUP BY artistas.nombre_artistico;
```