Guía de ejercicios - Manipulación de datos y transaccionalidad en las operaciones (I)

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

Tabla de contenidos

¡Comencemos!

Actividad guiada: Base de datos para registro de viajeros

A continuación, un cliente que realiza viajes a nivel nacional nos entrega el siguiente diagrama.

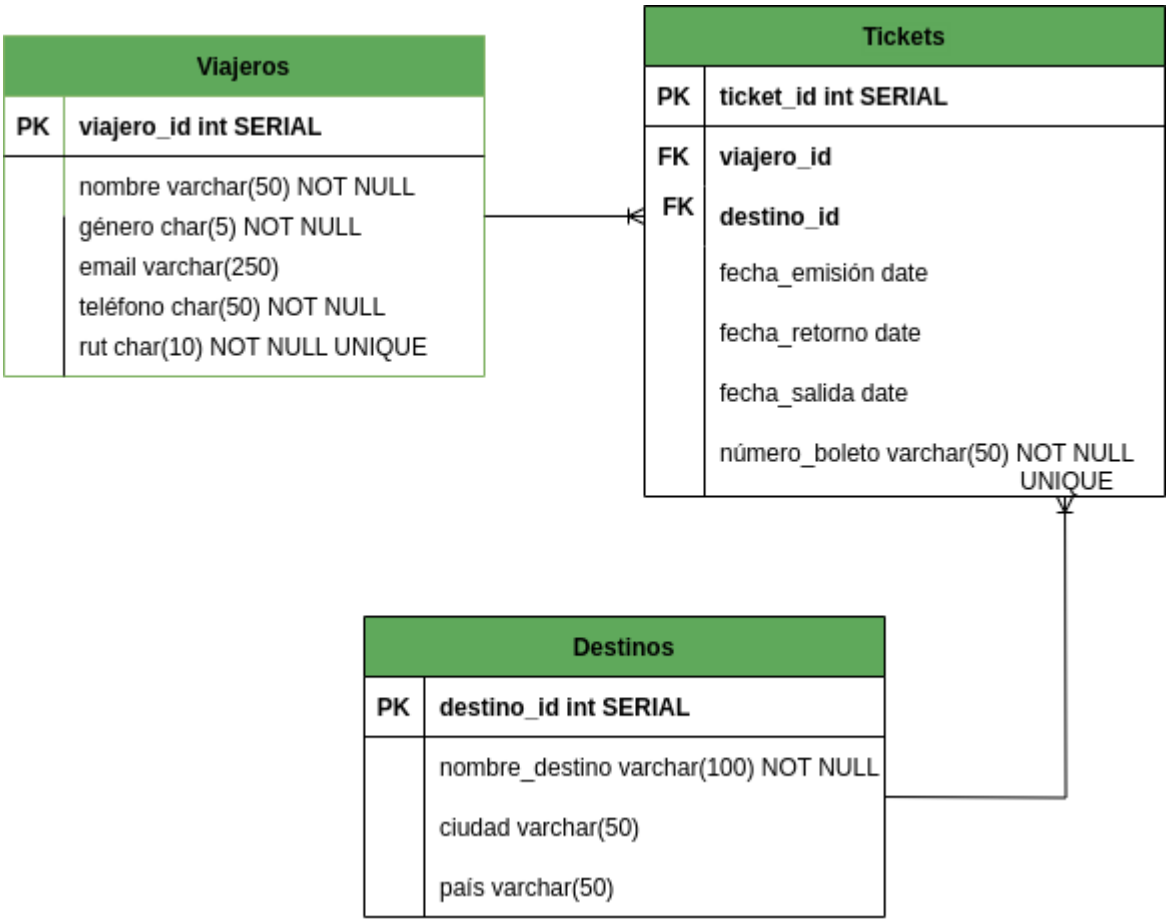

Imagen 1. Diagrama base de datos.
Fuente: Desafío Latam

Su requerimiento consiste en que podamos implementar las siguientes consultas en PostgreSQL para obtener información respecto a los viajeros y sus tickets:

1. Obtener una lista de todos los viajeros y la información de sus boletos, incluyendo aquellos viajeros que no tengan boletos emitidos. En el caso de los viajeros con boleto, mostrar su destino.
2. Mostrar la información del boleto T123456 junto con los detalles del viajero y destino correspondiente a ese boleto.
3. Listar todos los viajeros que tienen fecha de salida o de retorno el '2024-01-10'
4. Obtener el número total de boletos por cada género
5. Obtener un listado de todos los viajeros que han viajado a Playa del Carmen

Además, desea que insertemos algunos datos de prueba y comprobemos las restricciones de los campos.

La información de ambas tablas podemos encontrarla en el material complementario "03 - *Material de apoyo - Guía de ejercicios - Manipulación de datos y transaccionalidad en las operaciones (I)*".

- **Paso 1:** Generamos la base de datos con el nombre registro_viajeros

```
create database registro_viajeros;
```

- **Paso 2:** Nos conectamos a la base de datos.

```
\c registro_viajeros;
```

- **Paso 3:** Creamos la tabla viajeros con los datos presentes en el material de apoyo.

```
CREATE TABLE Viajeros (
    viajero_id SERIAL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    genero CHAR(5) NOT NULL,
    email VARCHAR(250),
    telefono CHAR(50) NOT NULL,
    rut CHAR(10) NOT NULL UNIQUE
);
```

- **Paso 4:** Creamos la tabla destinos con los datos presentes en el material de apoyo.

```
CREATE TABLE Destinos (
    destino_id SERIAL PRIMARY KEY,
    nombre_destino VARCHAR(100) NOT NULL,
    ciudad VARCHAR(50),
    pais VARCHAR(50)
);
```

- **Paso 5:** Creamos la tabla tickets con los datos presentes en el material de apoyo.

```
CREATE TABLE Tickets (
    ticket_id SERIAL PRIMARY KEY,
    destino_id INT REFERENCES DESTINOS (destino_id),
    viajero_id INT REFERENCES Viajeros(viajero_id),
    numero_boleto VARCHAR(50) NOT NULL UNIQUE,
    fecha_emision DATE,
    fecha_salida DATE,
    fecha_retorno DATE
);
```

- **Paso 6:** Insertamos los datos de las 3 tablas aplicando los inserts del material de apoyo.

Si comprobamos el ingreso del registro podremos observar lo siguiente:

```
select * from tickets;
```

ticket_id	destino_id	viajero_id	numero_boleto	fecha_emision	fecha_salida	fecha_retorno
1	1	1	T111111	2024-01-04	2024-01-10	2024-01-20
2	2	2	T222222	2024-01-05	2024-02-01	2024-02-15
3	3	2	T333333	2024-01-06	2024-03-05	2024-03-20
4	4	4	T444444	2024-01-07	2024-04-12	2024-05-01
5	5	5	T555555	2024-01-08	2024-06-02	2024-06-20
6	1	6	T666666	2024-01-09	2024-07-15	2024-08-01
7	2	4	T777777	2024-01-10	2024-09-03	2024-09-20
8	3	8	T888888	2024-01-11	2024-10-18	2024-11-01
9	4	9	T999999	2024-01-12	2024-12-05	2024-12-20
10	5	10	T101010	2024-01-13	2025-01-02	2025-01-20
11	1	15	T1111111	2024-01-14	2025-02-10	2025-02-25
12	2	12	T121212	2024-01-15	2025-03-15	2025-04-01
13	3	13	T131313	2024-01-16	2025-05-02	2025-05-20
14	4	14	T141414	2024-01-17	2025-06-12	2025-06-30
15	5	15	T151515	2024-01-18	2025-07-20	2025-08-05

(15 filas)

Imagen 2. Tabla Tickets creada en la base de datos.
Fuente: Desafío Latam

- **Paso 7:** El cliente nos solicita un pequeño reporte de pruebas donde se registren el nombre de todos los viajeros con y sin boletos, el número de boleto (para los casos que correspondan) y el nombre del destino.

```
SELECT viajeros.nombre, tickets.numero_boleto, destinos.nombre_destino
FROM viajeros
LEFT JOIN tickets ON viajeros.viajero_id = tickets.viajero_id
LEFT JOIN destinos ON tickets.destino_id = destinos.destino_id;
```

```
     nombre      | numero_boleto |    nombre_destino
-----------------+---------------+-----------------------
 Juan Perez      | T111111       | Playa del Carmen
 Maria Rodriguez | T222222       | Machu Picchu
 Maria Rodriguez | T333333       | Torres del Paine
 Luisa Martinez  | T444444       | Gran Barrera de Coral
 Pedro Hernandez | T555555       | Monte Everest
 Ana Lopez       | T666666       | Playa del Carmen
 Luisa Martinez  | T777777       | Machu Picchu
 Sofia Torres    | T888888       | Torres del Paine
 Daniel Castro   | T999999       | Gran Barrera de Coral
 Laura Garcia    | T101010       | Monte Everest
 Hector Mendoza  | T1111111      | Playa del Carmen
 Elena Vargas    | T121212       | Machu Picchu
 Gabriel Morales | T131313       | Torres del Paine
 Isabel Rios     | T141414       | Gran Barrera de Coral
 Hector Mendoza  | T151515       | Monte Everest
 Manuel Silva    |               |
 Carlos Gonzalez |               |
 Jorge Ramirez   |               |
(18 filas)
```

Imagen 3. Viajeros con y sin tickets y sus respectivos destinos.
Fuente: Desafío Latam

¡Manos a la obra! - Genera el reporte con el resto de las consultas de nuestro cliente.

En el paso 7 respondimos la primera consulta de nuestro cliente con el objetivo de guiarte en la realización de joins entre 3 tablas. Ahora te toca a ti!!! Realiza las consultas correspondientes a los 5 requerimientos restantes y genera un reporte. El reporte consistirá simplemente en pantallazos con el resultado de tus consultas.

¡Manos a la obra! - Insertando y borrando datos manteniendo integridad referencial

Ingresa tres nuevos tickets con la siguiente información:

1. Ticket 1

```
destino_id:3
viajero_id:4
numero_boleto:T171717
fecha_salida:2024-03-28
fecha_retorno:2024-04-01
```

2. Ticket 2

```
destino_id:5
viajero_id:10
numero_boleto:T888888
fecha_salida:2024-03-28
fecha_retorno:2024-04-01
```

3. Ticket 3

```
destino_id:4
numero_boleto:T202020
fecha_salida:2024-03-28
fecha_retorno:2024-04-01
```

¿Qué sucede en los tres casos? ¿A qué se debe la diferencia de comportamiento en los tres inserts?

A continuación borra los siguientes registros:

- Ticket con ID 4
- Viajero con ID 2
- Destino con ID 5

¿Por qué pudiste borrar algunos registros y otros no?
¿Qué solución podrías entregar para borrar todos los registros solicitados?

¡Manos a la obra! - Creando una nueva tabla

La tabla Destinos contiene las columnas País y Ciudad. Muchos destinos pueden corresponder al mismo país o ciudad por lo que quedarían muchos datos repetidos. ¿Y si creamos otra tabla? Crea una tabla llamada País que contenga las siguientes columnas: **país_id**, **nombre_país**, **ciudad** y **código_postal**. Recuerda indicar sus correspondientes tipos de datos y restricciones (el nombre del país y el código postal no pueden ser nulos).

Luego, realiza los inserts correspondientes a los países y ciudades que tenías originalmente registrados en la tabla destinos.

Finalmente, realiza estos tres pasos:

- Modifica la tabla Destinos para agregar el pais_id como FK
- Actualiza la tabla Destinos para que coincidan los pais_id en ambas tablas según el nombre de destino.
- Borra las columnas país y ciudad de la tabla Destinos

Solución "Genera el reporte con el resto de las consultas de nuestro cliente"

- Mostrar la información del boleto T123456 junto con los detalles del viajero y destino correspondiente a ese boleto.

```
SELECT Viajeros.*, Tickets.*, Destinos.*
FROM Tickets
JOIN Viajeros ON Tickets.viajero_id = Viajeros.viajero_id
JOIN Destinos ON Tickets.destino_id = Destinos.destino_id
WHERE Tickets.numero_boleto = 'T666666';
```

- Listar todos los viajeros que tienen fecha de salida o de retorno el '2024-01-10'

```
SELECT DISTINCT Viajeros.*
FROM Viajeros
JOIN Tickets ON Viajeros.viajero_id = Tickets.viajero_id
WHERE Tickets.fecha_salida = '2024-01-10' OR Tickets.fecha_retorno =
'2024-01-10';
```

- Obtener el número total de boletos por cada género

```
SELECT Viajeros.genero, COUNT(Tickets.ticket_id) AS total_boletos
FROM Viajeros
LEFT JOIN Tickets ON Viajeros.viajero_id = Tickets.viajero_id
GROUP BY Viajeros.genero;
```

- Obtener un listado de todos los viajeros que han viajado a Playa del Carmen

```
SELECT DISTINCT Viajeros.*
FROM Viajeros
JOIN Tickets ON Viajeros.viajero_id = Tickets.viajero_id
JOIN Destinos ON Tickets.destino_id = Destinos.destino_id
WHERE Destinos.nombre_destino = 'Playa del Carmen';
```

Solución “Creando una nueva tabla”

```
CREATE TABLE Paises (
    pais_id SERIAL PRIMARY KEY,
    nombre_pais VARCHAR(100) NOT NULL,
    ciudad VARCHAR(50),
    codigo_postal VARCHAR(20) NOT NULL
);
```

```
-- Insertar datos en la tabla Paises basados en los países existentes en
Destinos
INSERT INTO Paises (nombre_pais, ciudad, codigo_postal)
VALUES
('México', 'Playa del Carmen', '12345'),
('Perú', 'Cuzco', '54321'),
('Chile', 'Puerto Natales', '67890'),
('Australia', 'Queensland', '11111'),
('Nepal', 'Khumbu', '22222'),
('Grecia', 'Santorini', '33333'),
('Marruecos', 'Marrakech', '44444'),
('Japón', 'Kioto', '55555');
```

```
-- Agregar columna pais_id a la tabla Destinos
ALTER TABLE Destinos
ADD COLUMN pais_id INT REFERENCES Paises(pais_id);

-- Actualizar valores de pais_id basado en la información de la tabla
Paises
UPDATE Destinos
SET pais_id = (
    SELECT pais_id
    FROM Paises
    WHERE Paises.nombre_pais = Destinos.pais
    LIMIT 1
);

-- Eliminar columnas pais y ciudad de la tabla Destinos
ALTER TABLE Destinos
DROP COLUMN pais,
DROP COLUMN ciudad;
```