Facultad de Ingeniería
Base de Datos

Liseth Natalia Rivera Córdoba
2223510-3743

MoveFast

Diseño e implementación de la base de datos de un startup de alquiler de vehículos.

Resultados de las consultas realizadas con los datos insertados:

1. Obtener los vehículos disponibles en una ciudad específica.
Cali

	placa [PK] character varying (20)	marca character varying (100)	modelo_anho integer	sucursal_id integer
1	HEL-359	Chevrolet	2025	4

Bogotá

	placa [PK] character varying (20)	marca character varying (100)	modelo_anho integer	sucursal_id integer
1	ABC-123	Toyota	2020	1
2	MIZ-246	Mercedes	2023	2

2. Listar los alquileres activos con información del cliente y vehículo.

	aid integer	nombre character varying (100)	apellido character varying (100)	placa character varying (20)	marca character varying (100)	fecha_ini timestamp without time zone
1	1	Gross	Klein	ABC-123	Toyota	2025-04-30 00:11:28.784659
2	2	Nikol	Rivera	MIZ-246	Mercedes	2025-04-30 00:11:28.784659
3	4	Luis	Gómez	XYZ-789	Mazda	2025-04-30 00:11:28.784659
4	3	Ana	Pérez	HEL-359	Chevrolet	2025-04-30 00:11:28.784659

3. Calcular los ingresos totales por sucursal considerando solo vehículos con más de 3 alquileres.

	placa character varying (20)	sucursal_id integer	ing_totales numeric

4. Filtrar solo vehículos con más de 5 alquileres (usar subconsulta).

	placa [PK] character varying (20)	marca character varying (100)	modelo_anho integer	estado boolean	sucursal_id integer

5. Sumar los montos de todos los pagos asociados

	total_pagos numeric
1	3850000.00

Validación del Comportamiento (Restricciones)

Para garantizar que el esquema mantiene la integridad de los datos, realicé los siguientes casos de prueba en el Query Tool de PostgreSQL (en la parte del DML):

1. **NOT NULL:** El nombre del cliente no puede ser NULL. Al intentar ingresar un cliente con nombre NULL esta falla con violación de constraint not-null

```
ERROR:  null value in column "nombre" violates not-null constraint
Failing row contains (10, null, Pérez, null, null).
```

2. **CHECK:** El modelo_anho debe estar entre 2000 y 2025. Al intentar insertar un vehículo con año = 1999 esta falla con violación de check constraint

```
ERROR:  new row for relation "vehiculo" violates check constraint "vehiculo_modelo_anho_check"
Failing row contains (ERR-001, Test, 1999, t, 1).
```

3. **UNIQUE:** El email de Cliente debe ser único. Al intentar ingresar un cliente con un email repetido esta falla con violación de unique constraint

```
ERROR:  duplicate key value violates unique constraint "cliente_email_key"
Key (email)=(ana.perez@mail.com) already exists.
```

4. **ON DELETE CASCADE:** Primero inserte un cliente y un alquiler asociado

	aid [PK] integer	fecha_ini timestamp without time zone	fecha_fin timestamp without time zone	cid integer	placa character varying (20)
1	6	2025-04-30 02:01:15.413396	[null]	6	XYZ-789

Posteriormente, eliminé el cliente y así verifique que el alquiler se elimina automáticamente

	aid [PK] integer	fecha_ini timestamp without time zone	fecha_fin timestamp without time zone	cid integer	placa character varying (20)

5. **ON UPDATE CASCADE:** Actualizo sucursal_id en Sucursal de 2 a 22

	sucursal_id [PK] integer	snombre character varying (100)	calle character varying (20)	carrera character varying (20)	numero character varying (20)	ciudad character varying (20)
1	1	Centro	Av. 1	Cra. 5	10-20	Bogotá
2	3	Sur	[null]	Cra. 86	23-40	Yumbo
3	4	Norte	Cl. 2	[null]	82-A12	Cali
4	22	Norte	Cl. 100	Cra. 11	25-30	Bogotá

Luego, comprobé que cambió en la Relación Vehiculo verificando que el vehículo ‘MIZ-246' asociado modificó su sucursal_id a 22

	placa [PK] character varying (20)	sucursal_id integer
1	MIZ-246	22

Comportamiento ante Violaciones

1. **NOT NULL - UNIQUE - CHECK**
   PostgreSQL rechazó la operación con un mensaje de error señalando la restricción violada en cada caso y la fila problemática. Esto previene la corrupción o inconsistencia de datos.

2. **ON DELETE SET NULL**
   Al eliminar el registro padre (Vehiculo), las filas hijas (Alquiler) conservan su existencia, pero "pierden" la referencia, evitando registros huérfanos que apunten a una clave inexistente.

3. **ON DELETE CASCADE**:
   Borrar un cliente provoca automáticamente la eliminación en cascada de todos sus alquileres vinculados.

4. **ON UPDATE CASCADE**:
   Si se modifica la clave primaria de una tabla padre (por ejemplo, sucursal_id), todas las referencias en las tablas hijas (Vehiculo) se actualizan en bloque, preservando la conexión sin necesidad de actualizar manualmente cada fila.

5. **DEFAULT**
   Facilita la consistencia temporal (marcar fechas de creación/pago), reduciendo errores dy garantizando que ciertos campos siempre tengan un valor válido, por ejemplo, la fecha de pago, o la fecha de inicio de alquiler.

Ventajas y Desventajas de las Restricciones Implementadas

NOT NULL:

- **Ventaja**: ayuda a garantiza que los campos esenciales (nombres, fechas, montos) siempre estén completos
- **Desventaja**: obliga a proveer datos incluso cuando no estén disponibles

CHECK:

- **Ventaja**: impone reglas de dominio, capturando datos ilógicos
- **Desventaja**: si la lógica cambia, este debe cambiar

ON UPDATE CASCADE:

- **Ventaja**: mantiene intactas las relaciones cuando se cambian o renumeran claves primarias
- **Desventaja**: poco común renumerar llaves primarias por lo que implementarlo puede añadir complejidad innecesaria

ON DELETE SET NULL:

- **Ventaja**: preserva el registro hijo, limpiando solo la referencia, esto, aunque el padre desaparezca
- **Desventaja**: puede producir registros "incompletos" complicando la interpretación

ON DELETE CASCADE:

- **Ventaja**: elimina automáticamente todos los registros hijos cuando se borra un padre, manteniendo la integridad referencial
- **Desventaja**: dificulta la recuperación de datos borrados en cascada. Es importante saber que nunca es bueno eliminar datos, a no ser que sea estrictamente necesario.