
# INSTITUTO POLITÉCNICO NACIONAL

# ESCUELA SUPERIOR DE CÓMPUTO

## BASES DE DATOS

GRUPO: 3CV5

Profesor: Gabriel Hurtado Avilés

# “Práctica 4 Restricciones de dominio y asignación de permisos”

**Alumno**:

- González González Erick Emiliano
- De la Rosa Hernández Tania

Fecha de Entrega: 04/11/25

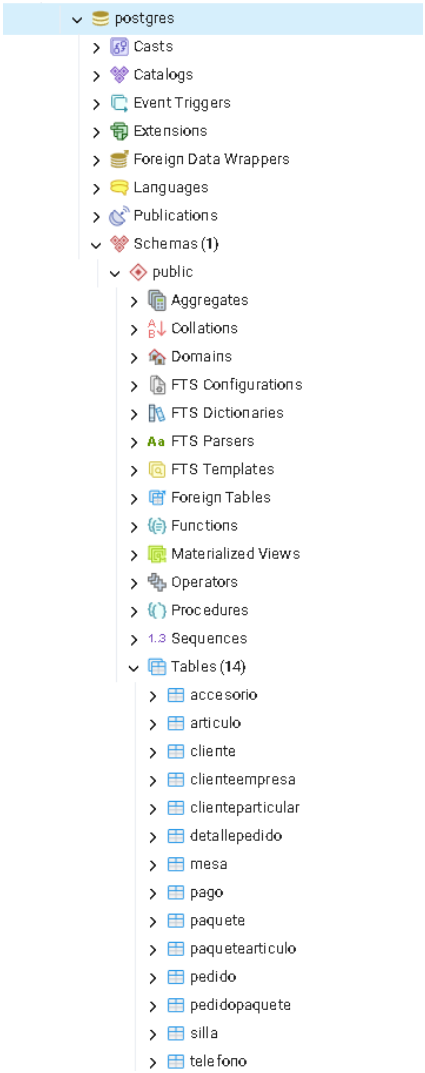
# Ejercicio 1: Preparación del Entorno de Base de Datos

1.2 Creación de la Base de Datos

Nombre: postgres
Character Set: UTF8

Justificación:

- Se usa UTF8 porque es el estándar universal de codificación, compatible con acentos y caracteres del idioma español.
- El nombre mantiene la convención en minúsculas y con guiones bajos para compatibilidad con comandos SQL.



# Ejercicio 2: Implementación con DDL - Restricciones de Dominio

## 2.1 Definición de Restricciones de Dominio

| Tabla | Atributo | Tipo de Dato | Restricción | Descripción / Justificación |
|---|---|---|---|---|
| Cliente | nombre | VARCHAR(100) | NOT NULL | El nombre del cliente es obligatorio para identificarlo. |
| Cliente | correo | VARCHAR(100) | UNIQUE | No pueden existir dos clientes con el mismo correo electrónico. |
| Articulo | estado | VARCHAR(20) | CHECK (estado IN ('Disponible', 'Rentado', 'Mantenimiento')) | Define un conjunto válido de estados posibles para cada artículo. |
| Articulo | cantidadTotal | INT | CHECK (cantidadTotal >= 0) | No se permiten cantidades negativas. |
| Articulo | costoRenta | NUMERIC(10,2) | CHECK (costoRenta > 0) | El costo de renta debe ser mayor que cero. |
| Pedido | fechaEvento | DATE | NOT NULL | Todo pedido debe tener una fecha programada para el evento. |
| Pedido | montoTotal | NUMERIC(10,2) | CHECK (montoTotal >= 0) | Evita montos negativos en los pedidos. |
| Pago | estadoPago | VARCHAR(20) | CHECK (estadoPago IN ('Pendiente', 'Completado', 'Cancelado')) | Establece los estados válidos para los pagos. |
| Telefono | numero | VARCHAR(15) | CHECK (numero ~ '^[0-9]{10}$') | Asegura que el número telefónico tenga exactamente 10 dígitos. |
| Mesa | capacidadPersonas | INT | CHECK (capacidadPersonas > 0) | La capacidad debe ser positiva. |

## 2.2 Ejemplo de implementación de restricciones en SQL

```sql
ALTER TABLE Articulo
    ADD CONSTRAINT chk_estado_articulo CHECK (estado IN ('Disponible', 'Rentado', 'Mantenimiento')),
    ADD CONSTRAINT chk_cantidad_total CHECK (cantidadTotal >= 0),
    ADD CONSTRAINT chk_costo_renta CHECK (costoRenta > 0);

ALTER TABLE Pedido
    ADD CONSTRAINT chk_monto_total CHECK (montoTotal >= 0);

ALTER TABLE Pago
    ADD CONSTRAINT chk_estado_pago CHECK (estadoPago IN ('Pendiente', 'Completado', 'Cancelado'));

ALTER TABLE Telefono
    ADD CONSTRAINT chk_numero_telefono CHECK (numero ~ '^[0-9]{10}$');

ALTER TABLE Mesa
    ADD CONSTRAINT chk_capacidad_personas CHECK (capacidadPersonas > 0);
```

```sql
211 ALTER TABLE Articulo
212     ADD CONSTRAINT chk_estado_articulo CHECK (estado IN ('Disponible', 'Rentado', 'Mantenimiento')),
213     ADD CONSTRAINT chk_cantidad_total CHECK (cantidadTotal >= 0),
214     ADD CONSTRAINT chk_costo_renta CHECK (costoRenta > 0);
215
216 ALTER TABLE Pedido
217     ADD CONSTRAINT chk_monto_total CHECK (montoTotal >= 0);
218
219 ALTER TABLE Pago
220     ADD CONSTRAINT chk_estado_pago CHECK (estadoPago IN ('Pendiente', 'Completado', 'Cancelado'));
221
222 ALTER TABLE Telefono
223     ADD CONSTRAINT chk_numero_telefono CHECK (numero ~ '^[0-9]{10}$');
224
225 ALTER TABLE Mesa
226     ADD CONSTRAINT chk_capacidad_personas CHECK (capacidadPersonas > 0);
227
228
```

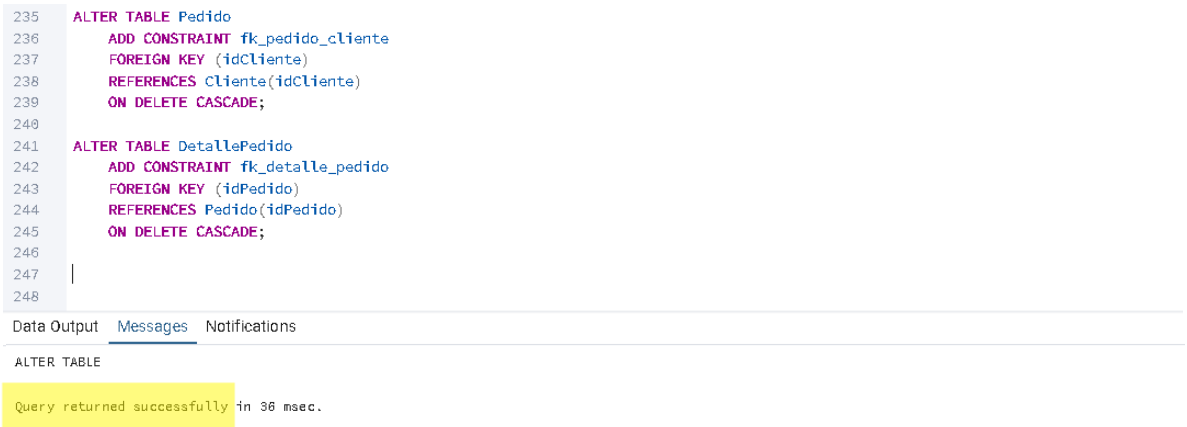
Data Output | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 36 msec.
```

# Ejercicio 3: Implementación con DDL - Integridad Referencial

## 3.1 Ejemplo de implementación de claves foráneas en PostgreSQL

```sql
ALTER TABLE Pedido
ADD CONSTRAINT fk_pedido_cliente
FOREIGN KEY (idCliente)
REFERENCES Cliente(idCliente)
ON DELETE CASCADE;

ALTER TABLE DetallePedido
ADD CONSTRAINT fk_detalle_pedido
FOREIGN KEY (idPedido)
REFERENCES Pedido(idPedido)
ON DELETE CASCADE;
```



## 3.2 Llaves Primarias y Foráneas existentes con sus acciones referenciales

| Tabla Hija | Atributo FK | Tabla Padre | Atributo PK Referenciado | Acción Referencial | Justificación |
|---|---|---|---|---|---|
| ClienteParticular | idCliente | Cliente | idCliente | ON DELETE CASCADE | Si un cliente es eliminado, también se elimina su registro en ClienteParticular. |
| ClienteEmpresa | idCliente | Cliente | idCliente | ON DELETE CASCADE | Elimina automáticamente la empresa si se borra el cliente principal. |
| Telefono | idCliente | Cliente | idCliente | ON DELETE CASCADE | Elimina los teléfonos asociados cuando el cliente deja de existir. |
| Silla | idArticulo | Articulo | idArticulo | ON DELETE CASCADE | Elimina la silla si el artículo general se borra. |
| Mesa | idArticulo | Articulo | idArticulo | ON DELETE CASCADE | Elimina la mesa cuando se borra el artículo padre. |
| Accesorio | idArticulo | Articulo | idArticulo | ON DELETE CASCADE | Elimina el accesorio si el artículo base desaparece. |
| Pedido | idCliente | Cliente | idCliente | ON DELETE CASCADE | Si un cliente se elimina, sus pedidos también. |
| DetallePedido | idPedido | Pedido | idPedido | ON DELETE CASCADE | Si el pedido se borra, también se eliminan sus detalles. |
| DetallePedido | idArticulo | Articulo | idArticulo | ON DELETE CASCADE | Evita que un artículo con pedidos activos sea eliminado sin eliminar el detalle. |
| Pago | idPedido | Pedido | idPedido | ON DELETE CASCADE | Si un pedido se borra, sus pagos se eliminan. |
| PaqueteArticulo | idPaquete | Paquete | idPaquete | ON DELETE CASCADE | Elimina la relación si el paquete ya no existe. |
| PaqueteArticulo | idArticulo | Articulo | idArticulo | ON DELETE CASCADE | Limpia las referencias a artículos eliminados. |
| PedidoPaquete | idPedido | Pedido | idPedido | ON DELETE CASCADE | Borra relaciones de paquetes al eliminar un pedido. |
| PedidoPaquete | idPaquete | Paquete | idPaquete | ON DELETE CASCADE | Evita referencias a paquetes inexistentes. |

# Ejercicio 4: Scripts DDL Completos

## 4.1 Scripts de prueba

```sql
-- Clientes
INSERT INTO Cliente (nombre, direccion, correo)
VALUES
('Carlos López', 'Av. Reforma 123', 'carlos@example.com'),
('María Pérez', 'Calle Juárez 56', 'maria@example.com'),
('Eventos Gala', 'Blvd. Hidalgo 102', 'contacto@eventosgala.mx'),
('Fiestas MX', 'Calle Palma 11', 'info@fiestasmx.com'),
('Luis Hernández', 'Priv. Jazmín 77', 'luis@example.com');

-- Cliente particulares y empresas
INSERT INTO ClienteParticular (idCliente, fechaNacimiento, CURP)
VALUES (1, '1990-05-12', 'LOLC900512HDFRZN01'),
       (2, '1987-09-25', 'PEMJ870925MDFTRN02'),
       (5, '1995-02-11', 'HEHL950211HDFCRS03');

INSERT INTO ClienteEmpresa (idCliente, razonSocial, RFC, contactoEmpresa)
VALUES (3, 'Eventos Gala S.A. de C.V.', 'EGA001122A12', 'Juan Ortega'),
       (4, 'Fiestas MX S.A.', 'FMX050707B89', 'Sofía Ramos');

-- Artículos
INSERT INTO Articulo (nombre, estado, cantidadTotal, costoRenta)
VALUES
('Silla blanca plegable', 'Disponible', 100, 15.00),
('Mesa redonda 10 personas', 'Disponible', 20, 80.00),
('Mantel blanco', 'Disponible', 50, 25.00),
('Silla Tiffany dorada', 'Disponible', 40, 35.00),
('Mesa rectangular', 'Disponible', 15, 90.00);

INSERT INTO Silla (idArticulo, tipoSilla, material)
VALUES (1, 'Plegable', 'Plástico'), (4, 'Tiffany', 'Metal');

INSERT INTO Mesa (idArticulo, forma, capacidadPersonas, tamaño)
VALUES (2, 'Redonda', 10, '1.8m'), (5, 'Rectangular', 12, '2m');

INSERT INTO Accesorio (idArticulo, descripcion, fragilidad)
VALUES (3, 'Mantel de tela blanca', 'Baja');

-- Pedidos y detalles
INSERT INTO Pedido (idCliente, fechaEvento, fechaEntrega, fechaDevolucion, montoTotal)
VALUES
(1, '2025-11-10', '2025-11-09', '2025-11-11', 500.00),
(3, '2025-12-15', '2025-12-14', '2025-12-16', 1200.00);

INSERT INTO DetallePedido (idPedido, idArticulo, cantidad)
VALUES
(1, 1, 20), (1, 2, 2), (2, 4, 15), (2, 5, 4);

-- Pagos
INSERT INTO Pago (idPedido, fechaPago, monto, estadoPago)
VALUES
(1, '2025-10-01', 250.00, 'Pendiente'),
(1, '2025-10-05', 250.00, 'Completado'),
(2, '2025-10-15', 600.00, 'Completado');

-- Paquetes
INSERT INTO Paquete (nombre, precioEspecial)
VALUES ('Paquete Fiesta', 450.00), ('Paquete Premium', 900.00);

INSERT INTO PaqueteArticulo (idPaquete, idArticulo, cantidad)
VALUES (1, 1, 20), (1, 2, 2), (2, 4, 15), (2, 5, 4);

INSERT INTO PedidoPaquete (idPedido, idPaquete, cantidad)
VALUES (1, 1, 1), (2, 2, 1);
```

Data Output | Messages | Notifications

```
INSERT 0 2

Query returned successfully in 86 msec.
```

## 4.3 Script de Consultas de Verificación

```sql
-- Verificar valores nulos indebidos
SELECT * FROM Cliente WHERE nombre IS NULL;
SELECT * FROM Articulo WHERE costoRenta IS NULL;

-- Verificar valores fuera de rango
SELECT * FROM Articulo WHERE costoRenta <= 0 OR cantidadTotal < 0;

-- Verificar claves foráneas huérfanas
SELECT d.* FROM DetallePedido d
LEFT JOIN Pedido p ON d.idPedido = p.idPedido
WHERE p.idPedido IS NULL;

-- Verificar duplicados en campos únicos
SELECT correo, COUNT(*) FROM Cliente GROUP BY correo HAVING COUNT(*) > 1;

-- Intento de insertar datos inválidos (debe fallar)
INSERT INTO Articulo (nombre, estado, cantidadTotal, costoRenta)
VALUES ('Silla rota', 'Disponible', -5, -10.00);
```

```
373  -- Intento de insertar datos inválidos (debe fallar)
374  INSERT INTO Articulo (nombre, estado, cantidadTotal, costoRenta)
375  VALUES ('Silla rota', 'Disponible', -5, -10.00);
376
377
378
379
```

Data Output | Messages | Notifications

```
ERROR:  new row for relation "articulo" violates check constraint "chk_cantidad_total"
Failing row contains (7, Silla rota, Disponible, -5, -10.00).

SQL state: 23514
Detail: Failing row contains (7, Silla rota, Disponible, -5, -10.00).
```

# 5: Implementación con DCL - Gestión de Usuarios y Seguridad

## 5.1 Roles a crear

| Rol | Descripción | Permisos principales |
|---|---|---|
| administrador | Supervisa todo el sistema, incluyendo usuarios y tablas. | ALL PRIVILEGES sobre todas las tablas y vistas. |
| empleado | Gestiona pedidos, pagos y artículos. | SELECT, INSERT, UPDATE en tablas operativas. |
| consultor | Solo puede visualizar información. | SELECT en todas las tablas. |

## 5.2 Creación de roles

```sql
-- Crear roles
CREATE ROLE administrador;
CREATE ROLE empleado;
CREATE ROLE consultor;

-- Crear usuarios y asignar roles
CREATE USER admin_user WITH PASSWORD 'Admin2025';
CREATE USER empleado_user WITH PASSWORD 'Empleado2025';
CREATE USER consultor_user WITH PASSWORD 'Consulta2025';

-- Asignar roles a los usuarios
GRANT administrador TO admin_user;
GRANT empleado TO empleado_user;
GRANT consultor TO consultor_user;
```

## 5.3 Asignación de Privilegios

```sql
-- Privilegios del Administrador
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO administrador;

-- Privilegios del Empleado
GRANT SELECT, INSERT, UPDATE ON
    Cliente, Pedido, Articulo, Pago, DetallePedido, Paquete
TO empleado;

-- Permitir uso de las secuencias de las tablas anteriores
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO empleado;

-- Privilegios del Consultor
GRANT SELECT ON ALL TABLES IN SCHEMA public TO consultor;
```

```sql
-- Privilegios del Administrador
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO administrador;

-- Privilegios del Empleado
GRANT SELECT, INSERT, UPDATE ON
    Cliente, Pedido, Articulo, Pago, DetallePedido, Paquete
TO empleado;

-- Permitir uso de las secuencias de las tablas anteriores
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO empleado;

-- Privilegios del Consultor
GRANT SELECT ON ALL TABLES IN SCHEMA public TO consultor;
```

Data Output | Messages | Notifications

```
GRANT

Query returned successfully in 43 msec.
```

## 5.4 Revocación de Privilegios

```sql
-- Ejemplo: revocar acceso total al empleado
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM empleado;
REVOKE empleado FROM empleado_user;
```

## 5.6 Matriz de Permisos

| Usuario/Rol | Tabla | SELECT | INSERT | UPDATE | DELETE | Sentencia DCL utilizada |
|---|---|---|---|---|---|---|
| administrador | Todas las tablas | ✓ | ✓ | ✓ | ✓ | GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador; |
| empleado | Cliente, Pedido, Articulo, Pago, DetallePedido, Paquete | ✓ | ✓ | ✓ | ✗ | GRANT SELECT, INSERT, UPDATE ON ... TO empleado; |
| consultor | Todas las tablas | ✓ | ✗ | ✗ | ✗ | GRANT SELECT ON ALL TABLES IN SCHEMA public TO consultor; |

# Ejercicio 6: Pruebas de Integridad y Seguridad

| # | Operación Intentada (SQL) | Resultado Esperado | Resultado Obtenido | Conclusión |
|---|---|---|---|---|
| 1 | INSERT INTO Articulo (nombre, estado, cantidadTotal, costoRenta) VALUES ('Silla plástica', 'Disponible', -10, 20.00); | Error: cantidadTotal no puede ser negativa | PostgreSQL lanza error: *violates check constraint "articulo_cantidadtotal_check"* | La restricción CHECK (cantidadTotal >= 0) funciona correctamente. |
| 2 | INSERT INTO Articulo (nombre, estado, cantidadTotal, costoRenta) VALUES ('Mesa baja', 'Disponible', 10, -5.00); | Error: costoRenta no puede ser menor o igual a 0 | Error lanzado | La restricción de rango en costoRenta está activa. |
| 3 | INSERT INTO Cliente (nombre, direccion, correo) VALUES (NULL, 'Av. Juárez 11', 'cliente@test.com'); | Error: el campo nombre es obligatorio | Error: *null value in column "nombre" violates not-null constraint* | Restricción NOT NULL en nombre activa. |
| 4 | INSERT INTO Cliente (nombre, direccion, correo) VALUES ('Juan', 'Calle 5', 'maria@example.com'); | Error: correo duplicado | Error: *duplicate key value violates unique constraint "cliente_correo_key"* | Restricción UNIQUE en correo funciona. |
| 5 | INSERT INTO Articulo (nombre, cantidadTotal, costoRenta) VALUES ('Lámpara decorativa', 5, 50.00); | Estado debe tomar valor por defecto “Disponible” | Registro insertado con estado “Disponible” | Restricción DEFAULT funcional. |
| 6 | INSERT INTO Mesa (idArticulo, forma, capacidadPersonas, tamaño) VALUES (2, 'Redonda', 0, '1m'); | Error: capacidadPersonas debe ser > 0 | Error lanzado | La restricción de rango se cumple. |
| 7 | INSERT INTO Pago (idPedido, fechaPago, monto, estadoPago) VALUES (1, '2025-10-01', -200.00, 'Pendiente'); | Error: monto negativo no permitido | Error lanzado | CHECK (monto >= 0) funciona. |
| 8 | INSERT INTO Pago (idPedido, fechaPago, monto, estadoPago) VALUES (1, '2025-10-01', 200.00, 'Inexistente'); | Error: valor no válido para estadoPago | Error lanzado | CHECK (estadoPago IN (...)) funciona. |
| 9 | INSERT INTO Telefono (idCliente, numero) VALUES (1, 'abc123'); | Error: número no cumple regex | Error lanzado | CHECK (numero ~ '^[0-9]{10}$') valida correctamente. |
| 10 | INSERT INTO Pedido (idCliente, fechaEvento, montoTotal) VALUES (1, '2025-10-30', -50.00); | Error: montoTotal negativo | Error lanzado | CHECK (montoTotal >= 0) verificado. |

# Conclusiones

Durante esta práctica se fortaleció nuestro conocimiento del lenguaje SQL en sus correspondientes comandos DDL (Data Definition Language) y DCL (Data Control Language), aplicados directamente sobre la base de datos *Sillas y Mesas Hernández* de forma local.
El desarrollo permitió no solo crear y estructurar correctamente las tablas con todas sus restricciones de integridad, sino también controlar el acceso a los datos mediante la creación de roles y permisos específicos.

Se comprobó la importancia de definir correctamente las relaciones y restricciones de dominio, ya que estas aseguran la consistencia de la información dentro de la base de datos y evitan errores lógicos.

A través de las pruebas de integridad referencial se evidenció cómo las acciones como ON DELETE CASCADE u ON DELETE RESTRICT afectan directamente la forma en que se propagan los cambios entre tablas relacionadas.

En la parte de seguridad, se logró comprender el papel del control de acceso basado en roles, estableciendo distintos niveles de privilegios (SELECT, INSERT, UPDATE, DELETE) para cada usuario según sus funciones.

Esto permitió observar cómo las sentencias GRANT y REVOKE ayudan a proteger la información sensible y a garantizar que cada usuario interactúe con la base de datos únicamente dentro de sus límites autorizados.

En general, esta práctica permitió consolidar el ciclo completo de diseño, implementación, pruebas y seguridad de una base de datos real, reforzando la comprensión de los principios de integridad, consistencia y control de acceso que son esenciales en cualquier sistema de gestión de información.

# Bibliografía

Awati, R. (2022, junio 29). *Data Definition Language (DDL)*. WhatIs; TechTarget. https://www.techtarget.com/whatis/definition/Data-Definition-Language-DDL

*Cascade in SQL*. (2023, diciembre 29). GeeksforGeeks. https://www.geeksforgeeks.org/sql/cascade-in-sql/

*DCL full form - data control language*. (2020, mayo 5). GeeksforGeeks. https://www.geeksforgeeks.org/sql/dcl-full-form/

Mounish, V. (2024, junio 24). *What are SQL DCL(Data Control Language) Commands?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2024/06/sql-dcl-data-control-language-commands/