PostgreSQL - Cheatsheet (DDL / DML / DCL / TCL + psql)

Notación

- **DDL: Data Definition Language**, define estructura (tablas, índices, esquemas).
- **DML: Data Manipulation Language**, manipula datos (insert/select/update/delete).
- **DCL: Data Control Language**, permisos y roles (grant/revoke).
- **TCL: Transaction Control Language**, control transaccional (commit/rollback/savepoint).

Esquemas (schema)

```
/* ============================================================
   ESQUEMAS (schema)
   ------------------------------------------------------------
   - Los objetos viven en: esquema.tabla  (ej: public.clientes)
   - Si escribes solo 'clientes', PostgreSQL busca en el search_path
   - El search_path por defecto suele ser: "$user", public
   - En DDL y scripts es buena práctica usar esquema.tabla
   ============================================================ */

/* Muestra el search_path actual */
SHOW search_path;

/* Cambia el search_path solo para la sesión actual */
SET search_path TO ventas, public;
```

1) DDL - DATA DEFINITION

```
/* Crear base de datos (se ejecuta fuera de una transacción en muchos clientes) */

-- Crea una base de datos nueva
CREATE DATABASE mi_db;
-- Elimina una base de datos (cuidado: destructivo)
DROP DATABASE mi_db;

/* Conectarse a una DB (esto es psql, lo dejo aquí como recordatorio) */
-- \c mi_db                                  -- Conecta a otra base de datos (psql)

/* Esquemas */
-- Crea un esquema (namespace) para organizar objetos
CREATE SCHEMA ventas;
-- Borra un esquema (falla si tiene objetos)
DROP SCHEMA ventas;
-- Borra esquema y todo lo que contiene (destructivo)
DROP SCHEMA ventas CASCADE;
-- Renombra el esquema
ALTER SCHEMA ventas RENAME TO comercial;

/* Tablas */
-- Crea una tabla si no existe
CREATE TABLE IF NOT EXISTS public.cliente (
  -- id_cliente: PK autoincremental (identity "clásico")
  id_cliente  BIGSERIAL PRIMARY KEY,
  -- nombre: Columna obligatoria
  nombre      TEXT NOT NULL,
  -- email: Debe ser único (permite NULL, pero los valores no nulos no se repiten)
  email       TEXT UNIQUE,
  -- creado_en: Fecha-hora con zona, por defecto "ahora"
  creado_en   TIMESTAMPTZ DEFAULT now()
);

-- Borra tabla (falla si hay dependencias)
DROP TABLE public.cliente;
-- Borra tabla y dependientes (vistas, FKs, etc.)
DROP TABLE public.cliente CASCADE;
-- Renombra la tabla
ALTER TABLE public.cliente RENAME TO clientes;
-- Agrega columna
ALTER TABLE public.clientes ADD COLUMN telefono TEXT;
-- Borra columna
ALTER TABLE public.clientes DROP COLUMN telefono;
-- Hace columna obligatoria
```

```sql
ALTER TABLE public.clientes ALTER COLUMN email SET NOT NULL;
-- Quita obligatoriedad
ALTER TABLE public.clientes ALTER COLUMN email DROP NOT NULL;
-- Cambia tipo (puede requerir USING)
ALTER TABLE public.clientes ALTER COLUMN nombre TYPE VARCHAR(120);

/* Identidad (recomendado en diseños modernos) */
CREATE TABLE producto (
  -- Crea tabla con identidad y check
  id_producto BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  -- id_producto: Autoincrement "identity" estándar SQL
  nombre      TEXT NOT NULL,
  -- precio: Restricción de validación
  precio      NUMERIC(10,2) NOT NULL CHECK (precio >= 0)
);

/* Constraints */
-- Evita nombres duplicados
ALTER TABLE producto ADD CONSTRAINT uq_producto_nombre UNIQUE (nombre);
-- Valida regla de negocio
ALTER TABLE producto ADD CONSTRAINT ck_precio CHECK (precio >= 0);
-- Elimina constraint por nombre
ALTER TABLE producto DROP CONSTRAINT uq_producto_nombre;

/* Claves foráneas */
CREATE TABLE pedido (
  id_pedido   BIGSERIAL PRIMARY KEY,
  id_cliente  BIGINT NOT NULL,
  creado_en   TIMESTAMPTZ DEFAULT now(),
  CONSTRAINT fk_pedido_cliente
    -- Relación a clientes
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
    -- ON DELETE RESTRICT/NO ACTION (por defecto), evita borrar cliente si tiene pedidos
);

-- Agrega FK a tabla existente
ALTER TABLE pedido ADD CONSTRAINT fk_pedido_cliente2
  FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente);

/* Índices */
-- Acelera búsquedas por email
CREATE INDEX idx_cliente_email ON clientes (email);
-- Índice único (similar a constraint UNIQUE)
CREATE UNIQUE INDEX idx_producto_nombre ON producto (nombre);
-- Borra índice
DROP INDEX idx_cliente_email;

/* Vistas */
-- Vista: "consulta guardada"
CREATE VIEW v_pedidos AS
SELECT p.id_pedido, c.nombre, p.creado_en
FROM pedido p
JOIN clientes c ON c.id_cliente = p.id_cliente;

-- Elimina la vista
DROP VIEW v_pedidos;

/* Extensiones útiles */
-- Habilita funciones crypto (gen_random_uuid, etc.)
CREATE EXTENSION IF NOT EXISTS pgcrypto;
-- UUIDs (según necesidad/entorno)
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

2) DML - DATA MANIPULATION

```sql
/* INSERT */
-- Inserta filas
INSERT INTO clientes (nombre, email)
VALUES ('Ana', 'ana@correo.com');

-- Devuelve datos generados (PK/fechas)
INSERT INTO clientes (nombre, email)
VALUES ('Beto', 'beto@correo.com')
RETURNING id_cliente, creado_en;

/* INSERT masivo */
-- Inserta varias filas de una
INSERT INTO producto (nombre, precio)
VALUES
  ('Cuerda 010', 5.99),
  ('Púa Jazz', 2.50);

/* SELECT básico */
-- Lista todas las columnas/filas (útil, pero no ideal en prod)
SELECT * FROM clientes;
```

```sql
/* SELECT con filtro, orden, límite */
SELECT id_cliente, nombre, email
FROM clientes
-- Filtra por condición
WHERE email IS NOT NULL
-- Ordena descendente
ORDER BY id_cliente DESC
-- Pagina resultados
LIMIT 10 OFFSET 0;

/* DISTINCT */
-- Quita duplicados del resultado
SELECT DISTINCT email
FROM clientes
WHERE email IS NOT NULL;

/* LIKE / ILIKE */
-- ILIKE = case-insensitive (PostgreSQL)
SELECT * FROM clientes
WHERE nombre ILIKE '%an%';

/* IN */
-- Coincide con una lista de valores
SELECT * FROM producto
WHERE nombre IN ('Púa Jazz', 'Cuerda 010');

/* BETWEEN */
-- Rango de fechas (incluye extremos)
SELECT * FROM pedido
WHERE creado_en BETWEEN now() - interval '7 days' AND now();

/* JOINs */
-- INNER JOIN: solo coincidencias
SELECT p.id_pedido, c.nombre
FROM pedido p
JOIN clientes c ON c.id_cliente = p.id_cliente;

-- LEFT JOIN: incluye clientes sin pedidos (p.* puede ser NULL)
SELECT c.nombre, p.id_pedido
FROM clientes c
LEFT JOIN pedido p ON p.id_cliente = c.id_cliente;

/* Agregaciones */
SELECT id_cliente, COUNT(*) AS total_pedidos
FROM pedido
-- Agrupa por cliente
GROUP BY id_cliente
-- Filtra grupos (no filas)
HAVING COUNT(*) >= 2;

/* Subquery */
-- Clientes que tienen al menos un pedido
SELECT *
FROM clientes
WHERE id_cliente IN (
  SELECT id_cliente FROM pedido
);

/* CTE (WITH) */
WITH ultimos_pedidos AS (
  SELECT * FROM pedido ORDER BY creado_en DESC LIMIT 5
)
SELECT u.id_pedido, c.nombre
FROM ultimos_pedidos u
-- CTE mejora legibilidad/organización
JOIN clientes c ON c.id_cliente = u.id_cliente;

/* UPDATE */
-- Modifica columnas
-- Devuelve filas afectadas
UPDATE clientes
SET email = 'ana.nuevo@correo.com'
WHERE id_cliente = 1
RETURNING *;

/* DELETE */
-- Elimina y devuelve lo borrado (si existía)
DELETE FROM clientes
WHERE id_cliente = 999
RETURNING *;

/* UPSERT (ON CONFLICT) */
INSERT INTO producto (nombre, precio)
VALUES ('Púa Jazz', 3.00)
ON CONFLICT (nombre) DO UPDATE
-- EXCLUDED = valor propuesto en el INSERT
SET precio = EXCLUDED.precio
```

```
-- Inserta o actualiza según conflicto en UNIQUE/PK
RETURNING *;
```

3) DCL - DATA CONTROL

```
/* Roles/usuarios */
-- Crea rol con login (equivalente a usuario)
CREATE ROLE app_user LOGIN PASSWORD 'cambia_esto';
-- Cambia contraseña
ALTER ROLE app_user PASSWORD 'otra_clave';
-- Deshabilita login
ALTER ROLE app_user NOLOGIN;
-- Borra rol (si no tiene dependencias)
DROP ROLE app_user;

/* Membresías de roles */
-- Rol "grupo" sin login
CREATE ROLE readonly;
-- app_user hereda permisos de readonly
GRANT readonly TO app_user;
-- Quita membresía
REVOKE readonly FROM app_user;

/* Permisos sobre base de datos */
-- Permite conectarse
GRANT CONNECT ON DATABASE mi_db TO app_user;
-- Quita conexión
REVOKE CONNECT ON DATABASE mi_db FROM app_user;

/* Permisos sobre esquemas */
-- Permite "ver" objetos del esquema
GRANT USAGE ON SCHEMA public TO app_user;
-- Permite crear objetos en el esquema
GRANT CREATE ON SCHEMA public TO app_user;

/* Permisos sobre tablas */
-- Permite leer datos
GRANT SELECT ON TABLE clientes TO app_user;
-- Permite insertar/modificar/borrar
GRANT INSERT, UPDATE, DELETE ON TABLE clientes TO app_user;
-- Quita todos los permisos sobre esa tabla
REVOKE ALL ON TABLE clientes FROM app_user;

/* Permisos sobre secuencias (si usas SERIAL) */
-- Permite usar nextval/currval
GRANT USAGE, SELECT ON SEQUENCE clientes_id_cliente_seq TO app_user;

/* Permisos por defecto para objetos futuros (muy útil) */
-- Tablas nuevas tendrán SELECT para readonly
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT ON TABLES TO readonly;

-- Secuencias nuevas accesibles para readonly
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT USAGE, SELECT ON SEQUENCES TO readonly;
```

4) TCL - TRANSACTION CONTROL

```
/* Transacción básica */
-- Inicia transacción manual
BEGIN;
-- Cambios quedan "pendientes"
UPDATE producto SET precio = precio * 1.05;
-- Confirma cambios (persisten)
COMMIT;

/* Revertir */
BEGIN;
DELETE FROM producto WHERE nombre = 'Púa Jazz';
-- Revierte lo hecho desde BEGIN
ROLLBACK;

/* Savepoints */
BEGIN;
UPDATE producto SET precio = 1 WHERE nombre = 'Cuerda 010';
-- Marca intermedia dentro de la transacción
SAVEPOINT sp1;
UPDATE producto SET precio = -999 WHERE nombre = 'Cuerda 010';
-- Revierte solo desde sp1 (mantiene lo anterior)
ROLLBACK TO SAVEPOINT sp1;
-- Confirma lo restante
COMMIT;

/* Nivel de aislamiento (cuando se necesita control de concurrencia) */
-- Máximo aislamiento (puede fallar por conflictos)
```

```sql
BEGIN ISOLATION LEVEL SERIALIZABLE;
-- ... operaciones ...
-- Confirma o falla (reintentar si es necesario)
COMMIT;
```

5) Comandos psql

Nota: Estos NO son SQL; son comandos del cliente psql.

```
# Muestra ayuda general de comandos \...
\?

# Sale de psql
\q

# Conecta a otra base de datos
\c nombre_db

# Muestra info de conexión actual (DB/usuario/host)
\conninfo

# Lista bases de datos
\l

# Lista esquemas
\dn

# Lista tablas (en el search_path)
\dt

# Lista tablas del esquema public
\dt public.*

# Describe estructura de una tabla (columnas, tipos, índices, etc.)
\d nombre_tabla

# Describe con más detalle (incluye tamaño y info extra)
\d+ nombre_tabla

# Lista índices
\di

# Lista vistas
\dv

# Lista funciones
\df

# Lista roles/usuarios
\du

# Muestra privilegios (ACL) sobre tablas/vistas/secuencias
\dp

# Toggle “expanded display” (resultados en formato vertical)
\x

# Toggle medición de tiempo de ejecución de consultas
\timing

# Activa/desactiva paginador (útil para outputs largos)
\pset pager on|off

# Hace que scripts se detengan al primer error
\set ON_ERROR_STOP on

# Imprime texto (útil en scripts)
\echo 'texto'

# Ejecuta un script SQL desde un archivo
\i ruta/archivo.sql

# Igual que \i pero relativo al script actual
\ir ruta/archivo.sql

# Ejecuta comando del sistema (desde psql)
\! comando_shell

# Exporta a CSV desde el cliente (no requiere permisos del servidor)
\copy tabla TO 'x.csv' CSV HEADER

# Importa CSV desde el cliente
\copy tabla FROM 'x.csv' CSV HEADER


# ------------------------------
```

```
# Búsqueda dentro de psql
# ----------------------------

# Muestra cada comando que se ejecuta (debug en scripts)
\set ECHO all

# Más detalle en errores
\set VERBOSITY verbose
```