psql esencial

```
\l                    lista las bases       \c db     conéctate
\dt  \di  \dv  \df    tablas / índices / vistas / funciones
\d+ name              describe, con índices y almacenamiento
\du  roles            \dn  esquemas          \dx  extensiones
\dp name              privilegios de una tabla
\x                    salida expandida (toggle)
\timing               muestra el tiempo de cada consulta
\e                    edita la consulta en $EDITOR
\i file.sql           ejecuta un archivo SQL
\copy t TO 'out.csv' CSV HEADER    export del lado del cliente
\set VERBOSITY verbose muestra el SQLSTATE con cada error
\watch 2              reejecuta la última consulta cada 2s
```

Qué se está ejecutando

```
SELECT pid, state, now() - query_start AS runtime,
       wait_event_type, left(query, 80) AS query
FROM pg_stat_activity
WHERE state <> 'idle'
ORDER BY query_start;
```

```
-- Quién bloquea a quién:
SELECT w.pid AS waiting, b.pid AS blocking,
       left(b.query, 60) AS blocking_query
FROM pg_stat_activity w
JOIN pg_stat_activity b
  ON b.pid = ANY (pg_blocking_pids(w.pid));
```

`state = 'idle in transaction'` + `xact_start` antiguo = problemas. Más: ver las consultas en ejecución.

Cancelar y matar

```
SELECT pg_cancel_backend(pid);     -- cancela la consulta, mantiene la sesión
SELECT pg_terminate_backend(pid);  -- cierra toda la conexión
```

Escala en ese orden. Nunca hagas `kill -9` a un backend desde la shell: se reinicia el clúster entero. Detalles: matar una consulta con seguridad.

Timeouts

```
SET statement_timeout = '30s';   -- tope de tiempo total (incl. esperas de lock)
SET lock_timeout = '3s';         -- espera máxima por un lock (ideal para DDL)
SET idle_in_transaction_session_timeout = '5min';

BEGIN;
SET LOCAL statement_timeout = '30min';  -- solo para esta transacción
-- ... migración o informe largos ...
COMMIT;
```

Cuando saltan: statement timeout · lock timeout.

Tamaños

```
SELECT pg_size_pretty(pg_total_relation_size('t')); -- heap+TOAST+índices
SELECT pg_size_pretty(pg_table_size('t'));          -- sin índices
SELECT pg_size_pretty(pg_indexes_size('t'));        -- solo índices
SELECT pg_size_pretty(pg_database_size(current_database()));
```

La consulta top-20 y la letra pequeña: tamaños de tablas e índices. Tamaño ≠ datos vivos — ver bloat.

Mantenimiento

```
ANALYZE t;                         -- actualiza las estadísticas del planner
VACUUM (VERBOSE, ANALYZE) t;       -- recupera filas muertas + estadísticas
CREATE INDEX CONCURRENTLY idx ON t (col);  -- sin lock de escritura
REINDEX INDEX CONCURRENTLY idx;

-- vigilancia del wraparound (alerta mucho antes de ~2 mil millones):
SELECT datname, age(datfrozenxid)
FROM pg_database ORDER BY 2 DESC;
```

El contexto: VACUUM, autovacuum y bloat · emergencia: wraparound.

EXPLAIN

```
EXPLAIN SELECT ...;                    -- solo el plan, no ejecuta
EXPLAIN (ANALYZE, BUFFERS) SELECT ...; -- EJECUTA la consulta, tiempos reales

-- escrituras, con seguridad:
BEGIN; EXPLAIN (ANALYZE) UPDATE ...; ROLLBACK;
```

Pega la salida (texto o JSON) en el visualizador EXPLAIN: flame graph y avisos automáticos.

Backup y restauración

```
pg_dump -Fc dbname > db.dump        # formato custom: comprimido,
pg_restore -j4 -d dbname db.dump    #   restauración paralela con -j
pg_dump -Fc -t mytable dbname > table.dump
pg_dumpall --globals-only > roles.sql   # los roles NO están en pg_dump
psql dbname < plain.sql             # restaura dumps SQL planos
```

`pg_dump` es consistente sin bloquear escrituras (se ejecuta en una transacción Repeatable Read).

Privilegios (rol de aplicación típico)

```
GRANT USAGE ON SCHEMA app TO app_user;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON ALL TABLES IN SCHEMA app TO app_user;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA app TO app_user;

-- también las tablas FUTURAS (creadas por el rol de migraciones):
ALTER DEFAULT PRIVILEGES FOR ROLE migrator IN SCHEMA app
  GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO app_user;
```

Los GRANT no cubren las tablas futuras — la trampa clásica. Más: permission denied.

Configuración

```
SHOW work_mem;
SELECT name, setting, source FROM pg_settings
WHERE source <> 'default';        -- qué se ha personalizado

ALTER SYSTEM SET work_mem = '64MB';
SELECT pg_reload_conf();          -- algunos parámetros requieren reinicio
```

`work_mem` es por nodo sort/hash, por consulta — se multiplica. `max_connections` y `shared_buffers` requieren reinicio.

Replicación: comprobaciones rápidas

```
SELECT pg_is_in_recovery();            -- ¿soy una réplica?
SELECT client_addr, state, replay_lag
FROM pg_stat_replication;              -- en el primario

-- slots que retienen WAL (active = f es una señal de alarma):
SELECT slot_name, active, pg_size_pretty(
  pg_wal_lsn_diff(pg_current_wal_lsn(), restart_lsn)) AS retained
FROM pg_replication_slots;
```

Los slots olvidados llenan discos: No space left on device · consultas canceladas en la réplica: conflict with recovery.

Bloqueos y transacciones

```
BEGIN ISOLATION LEVEL SERIALIZABLE;  -- reintenta con SQLSTATE 40001
SELECT ... FOR UPDATE;               -- bloquea las filas que vas a actualizar
SELECT ... FOR UPDATE SKIP LOCKED;   -- colas de trabajos sin contención
SELECT ... FOR UPDATE NOWAIT;        -- falla rápido en vez de esperar
SHOW default_transaction_isolation;  -- el nivel al que realmente ejecutas
```

Qué garantiza cada nivel y cómo reintentar correctamente: niveles de aislamiento en la práctica · deadlock detected.

Conexiones

```
SHOW max_connections;
SELECT count(*) FROM pg_stat_activity;
SELECT usename, application_name, count(*)
FROM pg_stat_activity GROUP BY 1, 2 ORDER BY 3 DESC;

ALTER ROLE app_user CONNECTION LIMIT 50;   -- límite por rol
```

¿Slots agotados? La solución duradera es un pooler, no un límite mayor: too many clients.

SQLSTATE comunes

`40P01` deadlock detected · `40001` serialization failure · `23505` duplicate key · `53300` too many clients · `57014` query canceled · `55P03` lock timeout · `42P01` relation does not exist · `42501` permission denied · `25P02` transaction aborted · `22P02` invalid input syntax

Los 30, con causas y soluciones: referencia de errores. En psql, `\set VERBOSITY verbose` muestra el código con cada error.