Referencia Rápida de PostgreSQL

Tablas, consultas, joins, índices, JSON, roles

Conexión

Línea de Comandos

```
psql -U postgres
psql -h localhost -p 5432 -U user -d mydb
psql "postgresql://user:pass@host:5432/mydb"
```

Metacomandos de psql

`\l`	Listar bases de datos
`\c dbname`	Conectar a la base de datos
`\dt`	Listar tablas
`\d tablename`	Describir la estructura de la tabla
`\dn`	Listar esquemas
`\du`	Listar roles
`\q`	Salir de psql
`\i file.sql`	Ejecutar archivo SQL

Tablas y Esquemas

Crear Tabla

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email TEXT UNIQUE,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

Operaciones de Esquema

```
CREATE SCHEMA app;
CREATE TABLE app.users (id SERIAL PRIMARY KEY);
SET search_path TO app, public;
DROP SCHEMA app CASCADE;
```

Modificar Tabla

```
ALTER TABLE users ADD COLUMN age INT;
ALTER TABLE users ALTER COLUMN name TYPE TEXT;
ALTER TABLE users DROP COLUMN age;
ALTER TABLE users RENAME TO customers;
```

Tipos de Datos

Numéricos

`INTEGER / INT`	Entero de 4 bytes
`BIGINT`	Entero de 8 bytes
`SERIAL`	Entero auto-incremental
`NUMERIC(p,s)`	Numérico exacto (ej., NUMERIC(10,2))
`REAL / DOUBLE PRECISION`	Punto flotante (4 / 8 bytes)
`BOOLEAN`	true / false / null

Cadenas y Binarios

`TEXT`	Texto variable ilimitado
`VARCHAR(n)`	Texto variable hasta n caracteres
`CHAR(n)`	Texto de longitud fija
`BYTEA`	Datos binarios
`UUID`	Identificador único universal de 128 bits

Fecha, JSON y Array

`DATE`	Fecha del calendario
`TIMESTAMPTZ`	Timestamp con zona horaria
`INTERVAL`	Intervalo de tiempo (ej., '2 days')
`JSONB`	JSON binario (indexable)
`INT[] / TEXT[]`	Tipos de array

Consultas

Insertar

```
INSERT INTO users (name, email)
VALUES ('Alice', 'alice@example.com')
RETURNING id;

INSERT INTO users (name, email) VALUES
('Bob', 'bob@ex.com'),
('Carol', 'carol@ex.com');
```

Seleccionar

```
SELECT * FROM users WHERE id = 1;
SELECT name, email FROM users
ORDER BY name LIMIT 10 OFFSET 20;
```

Actualizar

```
UPDATE users SET email = 'new@ex.com'
WHERE id = 1 RETURNING *;
```

Upsert

```
INSERT INTO users (email, name)
VALUES ('a@ex.com', 'Alice')
ON CONFLICT (email) DO UPDATE
SET name = EXCLUDED.name;
```

Eliminar

```
DELETE FROM users WHERE id = 1 RETURNING *;
TRUNCATE TABLE users RESTART IDENTITY;
```

Joins y Subconsultas

Tipos de Join

`INNER JOIN`	Filas coincidentes en ambas tablas
`LEFT JOIN`	Todas las filas izquierdas + coincidencias derechas
`RIGHT JOIN`	Todas las filas derechas + coincidencias izquierdas
`FULL OUTER JOIN`	Todas las filas de ambas tablas
`CROSS JOIN`	Producto cartesiano
`LATERAL JOIN`	Subconsulta que referencia la fila exterior

CTE (Expresión de Tabla Común)

```
WITH active AS (
  SELECT * FROM users WHERE active = true
)
SELECT a.name, o.total
FROM active a
JOIN orders o ON a.id = o.user_id;
```

Subconsulta

```
SELECT name FROM users
WHERE id IN (
  SELECT user_id FROM orders
  WHERE total > 100
);
```

Índices

Crear y Eliminar

```
CREATE INDEX idx_name ON users(name);
CREATE UNIQUE INDEX idx_email ON users(email);
CREATE INDEX idx_gin ON posts USING GIN(tags);
DROP INDEX idx_name;
```

Tipos de Índice

`B-tree`	Predeterminado, ideal para =, <, >, BETWEEN
`Hash`	Solo comparaciones de igualdad
`GIN`	Invertido generalizado — arrays, JSONB, texto completo
`GiST`	Búsqueda generalizada — geométrico, rangos
`BRIN`	Rango de bloques — tablas grandes ordenadas

Análisis de Consulta

```
EXPLAIN ANALYZE
SELECT * FROM users WHERE name = 'Alice';
```

Funciones y Procedimientos

Función SQL

```
CREATE FUNCTION active_count()
RETURNS INTEGER AS $$
  SELECT COUNT(*)::INT FROM users
  WHERE active = true;
$$ LANGUAGE sql;
SELECT active_count();
```

Función PL/pgSQL

```
CREATE FUNCTION greet(name TEXT)
RETURNS TEXT AS $$
BEGIN
  RETURN 'Hello, ' || name;
END;
$$ LANGUAGE plpgsql;
```

Funciones Integradas Útiles

`NOW() / CURRENT_TIMESTAMP`	Timestamp actual con zona horaria
`AGE(ts1, ts2)`	Intervalo entre timestamps
`COALESCE(a, b)`	Primer valor no nulo
`NULLIF(a, b)`	NULL si a = b
`GENERATE_SERIES(1,10)`	Generar filas de valores secuenciales
`STRING_AGG(col, ',')`	Concatenar valores con separador

Roles y Permisos

Gestión de Roles

```
CREATE ROLE app LOGIN PASSWORD 'secret';
ALTER ROLE app SET search_path TO myapp;
DROP ROLE app;
```

Privilegios

```
GRANT ALL ON DATABASE mydb TO app;
GRANT SELECT, INSERT ON users TO reader;
GRANT USAGE ON SCHEMA public TO app;
REVOKE INSERT ON users FROM reader;
```

Seguridad a Nivel de Fila

```
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_own ON users
FOR ALL USING (id = current_setting('app.uid')::INT);
```

Soporte JSON

Operadores JSONB

`->'key'`	Obtener valor JSON por clave (como JSON)
`->>'key'`	Obtener valor JSON por clave (como texto)
`#>'{a,b}'`	Obtener valor anidado por ruta
`@>`	Contiene (izquierda incluye derecha)
`?`	La clave existe
`||`	Concatenar valores JSONB

Referencia Rápida de PostgreSQL

Consultas JSONB

```
SELECT data->>'name' FROM profiles
WHERE data @> '{"active": true}';

SELECT * FROM profiles
WHERE data ? 'email';
```

Funciones JSONB

```
SELECT jsonb_each(data) FROM profiles;
SELECT jsonb_array_elements('[1,2,3]');
SELECT jsonb_set(data, '{name}', '"Alice"')
FROM profiles WHERE id = 1;
```

Patrones Comunes

Transacciones

```
BEGIN;
UPDATE accounts SET balance = balance - 100
WHERE id = 1;
UPDATE accounts SET balance = balance + 100
WHERE id = 2;
COMMIT;  -- or ROLLBACK;
```

Funciones de Ventana

```
SELECT name, salary,
  RANK() OVER (ORDER BY salary DESC),
  AVG(salary) OVER (PARTITION BY dept)
FROM employees;
```

Copiar Datos

```
COPY users TO '/tmp/users.csv'
WITH (FORMAT csv, HEADER);
COPY users FROM '/tmp/users.csv'
WITH (FORMAT csv, HEADER);
```

Respaldo con pg_dump

```
pg_dump -U postgres mydb > backup.sql
pg_dump -Fc mydb > backup.dump
pg_restore -d mydb backup.dump
```