REFERÊNCIA RÁPIDA DE POSTGRESQL

Tabelas, consultas, joins, índices, JSON, roles

Conexão

Linha de Comando

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

Metacomandos do psql

`\l`	Listar bancos de dados
`\c dbname`	Conectar ao banco de dados
`\dt`	Listar tabelas
`\d tablename`	Descrever estrutura da tabela
`\dn`	Listar schemas
`\du`	Listar roles
`\q`	Sair do psql
`\i file.sql`	Executar arquivo SQL

Tabelas e Schemas

Criar Tabela

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

Operações de Schema

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

Alterar Tabela

```
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 Dados

Numérico

`INTEGER / INT`	Inteiro de 4 bytes
`BIGINT`	Inteiro de 8 bytes
`SERIAL`	Inteiro auto-incrementável
`NUMERIC(p,s)`	Numérico exato (ex.: NUMERIC(10,2))
`REAL / DOUBLE PRECISION`	Ponto flutuante (4 / 8 bytes)
`BOOLEAN`	true / false / null

String e Binário

`TEXT`	Texto variável ilimitado
`VARCHAR(n)`	Texto variável até n caracteres
`CHAR(n)`	Texto de comprimento fixo
`BYTEA`	Dados binários
`UUID`	Identificador único universal de 128 bits

Data, JSON e Array

`DATE`	Data do calendário
`TIMESTAMPTZ`	Timestamp com fuso horário
`INTERVAL`	Intervalo de tempo (ex.: '2 days')
`JSONB`	JSON binário (indexável)
`INT[] / TEXT[]`	Tipos de array

Consultas

Inserir

```
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');
```

Selecionar

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

Atualizar

```
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;
```

Excluir

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

Joins e Subconsultas

Tipos de Join

`INNER JOIN`	Linhas correspondentes em ambas as tabelas
`LEFT JOIN`	Todas as linhas da esquerda + correspondentes da direita
`RIGHT JOIN`	Todas as linhas da direita + correspondentes da esquerda
`FULL OUTER JOIN`	Todas as linhas de ambas as tabelas
`CROSS JOIN`	Produto cartesiano
`LATERAL JOIN`	Subconsulta referenciando linha externa

CTE (Expressão de Tabela Comum)

```
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

Criar e Remover

```
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`	Padrão, ideal para =, <, >, BETWEEN
`Hash`	Apenas comparações de igualdade
`GIN`	Invertido generalizado — arrays, JSONB, texto completo
`GiST`	Busca generalizada — geométrico, intervalo
`BRIN`	Intervalo de bloco — tabelas grandes ordenadas

Análise de Consulta

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

Funções e Procedimentos

Função SQL

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

Função PL/pgSQL

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

Funções Integradas Úteis

`NOW() / CURRENT_TIMESTAMP`	Timestamp atual com fuso horário
`AGE(ts1, ts2)`	Intervalo entre timestamps
`COALESCE(a, b)`	Primeiro valor não nulo
`NULLIF(a, b)`	NULL se a = b
`GENERATE_SERIES(1,10)`	Gerar linhas de valores sequenciais
`STRING_AGG(col, ',')`	Concatenar valores com separador

Roles e Permissões

Gerenciamento de Role

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

Concessões

```
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;
```

Segurança em Nível de Linha

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

Suporte a JSON

Operadores JSONB

`->'key'`	Obter valor JSON por chave (como JSON)
`->>'key'`	Obter valor JSON por chave (como texto)
`#>'{a,b}'`	Obter valor aninhado por caminho
`@>`	Contém (esquerda inclui direita)
`?`	Chave existe
`||`	Concatenar valores JSONB

Consultas JSONB

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

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

Funções 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;
```

Padrões Comuns

Transações

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

Funções de Janela

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

Copiar Dados

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

Backup com pg_dump

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