

Référence rapide PostgreSQL

Tables, requêtes, jointures, index, JSON, rôles

Connexion

Ligne de commande

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

Méta-commandes psql

\l	Lister les bases de données
\c dbname	Se connecter à une base de données
\dt	Lister les tables
\d tablename	Décrire la structure de la table
\dn	Lister les schémas
\du	Lister les rôles
\q	Quitter psql
\i file.sql	Exécuter un fichier SQL

Tables et schémas

Créer une table

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

Opérations sur les schémas

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

Modifier une table

```
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;
```

Types de données

Numérique

INTEGER / INT	Entier sur 4 octets
BIGINT	Entier sur 8 octets
SERIAL	Entier auto-incrémenté
NUMERIC(p, s)	Numérique exact (ex. NUMERIC(10,2))
REAL / DOUBLE PRECISION	Virgule flottante (4 / 8 octets)
BOOLEAN	true / false / null

Chaîne et binaire

TEXT	Texte variable illimité
VARCHAR(n)	Texte variable jusqu'à n caractères
CHAR(n)	Texte de longueur fixe
BYTEA	Données binaires
UUID	Identifiant unique universel sur 128 bits

Date, JSON et tableau

DATE	Date calendrier
TIMESTAMPTZ	Horodatage avec fuseau horaire
INTERVAL	Durée (ex. '2 days')
JSONB	JSON binaire (indexable)
INT[] / TEXT[]	Types de tableau

Requêtes

Insert

```
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');
```

Select

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

Update

```
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;
```

Delete

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

Jointures et sous-requêtes

Types de jointures

INNER JOIN	Lignes correspondantes dans les deux tables
LEFT JOIN	Toutes les lignes gauches + correspondances droites
RIGHT JOIN	Toutes les lignes droites + correspondances gauches
FULL OUTER JOIN	Toutes les lignes des deux tables
CROSS JOIN	Produit cartésien
LATERAL JOIN	Sous-requête référençant la ligne externe

CTE (Expression de table commune)

```
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;
```

Sous-requête

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

Index

Créer et supprimer

```
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;
```

Types d'index

B-tree	Par défaut, bon pour =, <, >, BETWEEN
Hash	Comparaisons d'égalité uniquement
GIN	Inversé généralisé — tableaux, JSONB, texte intégral
GIST	Recherche généralisée — géométrie, intervalles
BRIN	Plage de blocs — grandes tables triées

Analyse de requête

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

Fonctions et procédures

Fonction SQL

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

Fonction PL/pgSQL

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

Fonctions intégrées utiles

NOW() / CURRENT_TIMESTAMP	Horodatage courant avec fuseau
AGE(ts1, ts2)	Intervalle entre les horodatages
COALESCE(a, b)	Première valeur non nulle
NULLIF(a, b)	NULL si a = b
GENERATE_SERIES(1, 10)	Générer des lignes de valeurs séquentielles
STRING_AGG(col, ' ', '')	Concaténer des valeurs avec séparateur

Rôles et permissions

Gestion des rôles

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

Grants

```
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;
```

Sécurité au niveau des lignes

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

Support JSON

Opérateurs JSONB

-> 'key'	Obtenir la valeur JSON par clé (en JSON)
->> 'key'	Obtenir la valeur JSON par clé (en texte)
#> '{a, b}'	Obtenir la valeur imbriquée par chemin
@>	Contient (gauche inclut droite)
?	La clé existe
 	Concaténer des valeurs JSONB

Référence rapide PostgreSQL

Requêtes JSONB

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

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

Fonctions 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;
```

Motifs courants

Transactions

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

Fonctions de fenêtrage

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

Copier des données

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

Sauvegarde pg_dump

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