PostgreSQL クイックリファレンス

テーブル、クエリ、ジョイン、インデックス、JSON、ロール

接続

コマンドライン

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

psql メタコマンド

\l	データベース一覧を表示
\c dbname	データベースに接続
\dt	テーブル一覧を表示
\d tablename	テーブル構造を表示
\dn	スキーマ一覧を表示
\du	ロール一覧を表示
\q	psql を終了
\i file.sql	SQL ファイルを実行

テーブルとスキーマ

テーブルの作成

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

スキーマ操作

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

テーブルの変更

```
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;
```

データ型

数値

INTEGER / INT	4 バイト整数
BIGINT	8 バイト整数
SERIAL	自動インクリメント整数
NUMERIC(p,s)	固定小数点数（例：NUMERIC(10,2)）
REAL / DOUBLE PRECISION	浮動小数点数（4 / 8 バイト）
BOOLEAN	true / false / null

文字列とバイナリ

TEXT	可変長の無制限テキスト
VARCHAR(n)	n 文字までの可変長テキスト
CHAR(n)	固定長テキスト
BYTEA	バイナリデータ
UUID	128 ビットの汎用一意識別子

日付、JSON と配列

DATE	カレンダー日付
TIMESTAMPTZ	タイムゾーン付きタイムスタンプ
INTERVAL	時間間隔（例：'2 days'）
JSONB	バイナリ JSON（インデックス可能）
INT[] / TEXT[]	配列型

クエリ

挿入

```
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');
```

選択

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

更新

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

アップサート

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

削除

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

ジョインとサブクエリ

ジョインの種類

INNER JOIN	両テーブルで一致する行
LEFT JOIN	左の全行 + 一致する右の行
RIGHT JOIN	右の全行 + 一致する左の行
FULL OUTER JOIN	両テーブルの全行
CROSS JOIN	直積（デカルト積）
LATERAL JOIN	外部行を参照するサブクエリ

CTE（共通テーブル式）

```
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;
```

サブクエリ

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

インデックス

作成と削除

```
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;
```

インデックスの種類

B-tree	デフォルト、=、<、>、BETWEEN に有効
Hash	等価比較のみ
GIN	汎用転置インデックス — 配列、JSONB、全文検索
GiST	汎用検索ツリー — 幾何、範囲
BRIN	ブロック範囲 — ソート済みの大きなテーブル

クエリ分析

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

関数とプロシージャ

SQL 関数

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

PL/pgSQL 関数

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

便利な組み込み関数

NOW() / CURRENT_TIMESTAMP	タイムゾーン付きの現在のタイムスタンプ
AGE(ts1, ts2)	タイムスタンプ間の間隔
COALESCE(a, b)	最初の null でない値
NULLIF(a, b)	a = b の場合は NULL
GENERATE_SERIES(1,10)	連続値の行を生成
STRING_AGG(col, ',')	区切り文字で値を結合

ロールと権限

ロールの管理

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

権限の付与

```
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;
```

行レベルセキュリティ

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

PostgreSQL クイックリファレンス

JSON サポート

JSONB 演算子

`->'key'`	キーで JSON 値を取得（JSON として）
`->>'key'`	キーで JSON 値を取得（テキストとして）
`#>'{a,b}'`	パスでネストした値を取得
`@>`	包含（左が右を含む）
`?`	キーが存在する
`||`	JSONB 値を結合

JSONB クエリ

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

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

JSONB 関数

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

よくあるパターン

トランザクション

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

ウィンドウ関数

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

データのコピー

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

pg_dump バックアップ

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