Referensi Cepat PostgreSQL

Tabel, query, join, index, JSON, dan role

Menghubungkan

Command Line

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

Meta-Command psql

`\l`	Daftar database
`\c dbname`	Hubungkan ke database
`\dt`	Daftar tabel
`\d tablename`	Tampilkan struktur tabel
`\dn`	Daftar schema
`\du`	Daftar role
`\q`	Keluar dari psql
`\i file.sql`	Jalankan file SQL

Tabel & Schema

Buat Tabel

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

Operasi Schema

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

Alter 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;
```

Tipe Data

Numerik

`INTEGER / INT`	Integer 4-byte
`BIGINT`	Integer 8-byte
`SERIAL`	Integer auto-increment
`NUMERIC(p,s)`	Numerik eksak (mis. NUMERIC(10,2))
`REAL / DOUBLE PRECISION`	Floating-point (4 / 8 byte)
`BOOLEAN`	true / false / null

String & Biner

`TEXT`	Teks panjang tak terbatas
`VARCHAR(n)`	Teks variabel hingga n karakter
`CHAR(n)`	Teks panjang tetap
`BYTEA`	Data biner
`UUID`	ID unik universal 128-bit

Tanggal, JSON & Array

`DATE`	Tanggal kalender
`TIMESTAMPTZ`	Timestamp dengan zona waktu
`INTERVAL`	Rentang waktu (mis. '2 days')
`JSONB`	JSON biner (dapat diindeks)
`INT[] / TEXT[]`	Tipe array

Query

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;
```

Join & Subquery

Tipe Join

`INNER JOIN`	Baris yang cocok di kedua tabel
`LEFT JOIN`	Semua baris kiri + yang cocok di kanan
`RIGHT JOIN`	Semua baris kanan + yang cocok di kiri
`FULL OUTER JOIN`	Semua baris dari kedua tabel
`CROSS JOIN`	Produk kartesian
`LATERAL JOIN`	Subquery yang mereferensikan baris luar

CTE (Common Table Expression)

```
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;
```

Subquery

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

Index

Buat & Hapus

```
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;
```

Tipe Index

`B-tree`	Default, cocok untuk =, <, >, BETWEEN
`Hash`	Hanya perbandingan kesamaan
`GIN`	Generalized inverted — array, JSONB, full-text
`GiST`	Generalized search — geometri, range
`BRIN`	Block range — tabel besar yang terurut

Analisis Query

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

Fungsi & Prosedur

Fungsi SQL

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

Fungsi PL/pgSQL

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

Fungsi Bawaan Berguna

`NOW() / CURRENT_TIMESTAMP`	Timestamp saat ini dengan TZ
`AGE(ts1, ts2)`	Interval antara dua timestamp
`COALESCE(a, b)`	Nilai non-null pertama
`NULLIF(a, b)`	NULL jika a = b
`GENERATE_SERIES(1,10)`	Hasilkan baris nilai berurutan
`STRING_AGG(col, ',')`	Gabungkan nilai dengan separator

Role & Izin

Manajemen Role

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

Grant

```
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;
```

Row-Level Security

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

Dukungan JSON

Operator JSONB

`->'key'`	Ambil nilai JSON berdasarkan key (sebagai JSON)
`->>'key'`	Ambil nilai JSON berdasarkan key (sebagai teks)
`#>'{a,b}'`	Ambil nilai bersarang berdasarkan path
`@>`	Mengandung (kiri mencakup kanan)
`?`	Key ada
`||`	Gabungkan nilai JSONB

Referensi Cepat PostgreSQL

Query JSONB

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

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

Fungsi 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;
```

Pola Umum

Transaksi

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

Window Function

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

Salin Data

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

Backup pg_dump

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