PostgreSQL 快速参考

表、查询、JOIN、索引、JSON 与角色管理

连接

命令行连接

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

psql 元命令

`\l`	列出所有数据库
`\c dbname`	连接到数据库
`\dt`	列出所有表
`\d tablename`	查看表结构
`\dn`	列出 schema
`\du`	列出角色
`\q`	退出 psql
`\i file.sql`	执行 SQL 文件

表与 Schema

创建表

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

Schema 操作

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

修改表

```
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;
```

数据类型

数值类型

`INTEGER / INT`	4 字节整数
`BIGINT`	8 字节整数
`SERIAL`	自增整数
`NUMERIC(p,s)`	精确数值（如 NUMERIC(10,2)）
`REAL / DOUBLE PRECISION`	浮点数（4 / 8 字节）
`BOOLEAN`	true / false / null

字符串与二进制

`TEXT`	不限长度的可变文本
`VARCHAR(n)`	最多 n 个字符的可变文本
`CHAR(n)`	固定长度文本
`BYTEA`	二进制数据
`UUID`	128 位通用唯一标识符

日期、JSON 与数组

`DATE`	日历日期
`TIMESTAMPTZ`	带时区的时间戳
`INTERVAL`	时间跨度（如 '2 days'）
`JSONB`	二进制 JSON（可建索引）
`INT[] / TEXT[]`	数组类型

查询

插入

```
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');
```

查询

```
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 *;
```

Upsert

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

删除

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

JOIN 与子查询

JOIN 类型

`INNER JOIN`	两表中均匹配的行
`LEFT JOIN`	左表全部行 + 右表匹配行
`RIGHT JOIN`	右表全部行 + 左表匹配行
`FULL OUTER JOIN`	两表全部行
`CROSS JOIN`	笛卡尔积
`LATERAL JOIN`	可引用外部行的子查询

CTE（公共表表达式）

```
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;
```

子查询

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

索引

创建与删除

```
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;
```

索引类型

`B-tree`	默认，适合 =、<、>、BETWEEN
`Hash`	仅用于等值比较
`GIN`	通用倒排索引，适合数组、JSONB、全文搜索
`GiST`	通用搜索树，适合几何、范围类型
`BRIN`	块范围索引，适合大型有序表

查询分析

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

函数与存储过程

SQL 函数

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

PL/pgSQL 函数

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

常用内置函数

`NOW() / CURRENT_TIMESTAMP`	当前带时区时间戳
`AGE(ts1, ts2)`	两个时间戳之间的间隔
`COALESCE(a, b)`	返回第一个非 null 值
`NULLIF(a, b)`	若 a = b 则返回 NULL
`GENERATE_SERIES(1,10)`	生成连续值序列
`STRING_AGG(col, ',')`	用分隔符拼接值

角色与权限

角色管理

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

授权

```
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;
```

行级安全

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

JSON 支持

JSONB 运算符

`->'key'`	按键获取 JSON 值（JSON 类型）
`->>'key'`	按键获取 JSON 值（文本类型）
`#>'{a,b}'`	按路径获取嵌套值
`@>`	包含（左侧包含右侧）
`?`	键是否存在
`||`	拼接两个 JSONB 值

PostgreSQL 快速参考

JSONB 查询

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

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

JSONB 函数

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

常用模式

事务

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

窗口函数

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

数据导入导出

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

pg_dump 备份

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