Pg

PostgreSQL 常用命令汇总

数据库管理员与开发者速查手册

版本：PostgreSQL 14+

2026年2月

目录

一、连接与断开数据库

1.1 连接数据库

使用 psql 命令行工具连接数据库

```
# 基本连接方式
psql -U username -d database_name -h hostname -p port

# 示例：连接本地数据库
psql -U postgres -d mydb

# 连接远程数据库
psql -U postgres -h 192.168.1.100 -p 5432 -d mydb

# 使用完整连接字符串
psql "postgresql://username:password@hostname:port/database"
```

1.2 断开连接

在 psql 交互式终端中退出

```
\q          -- 退出 psql
\quit       -- 同上
Ctrl+D      -- 快捷键退出
```

1.3 常用 psql 元命令

表 1 psql 元命令速查

命令	说明
`\l` 或 `\list`	列出所有数据库
`\c [dbname]`	切换到指定数据库

`\dt`	列出当前数据库的所有表
`\d table_name`	查看表结构
`\du`	列出所有用户
`\dn`	列出所有 schema
`\df`	列出所有函数
`\dv`	列出所有视图
`\di`	列出所有索引
`\timing`	开启/关闭执行时间显示
`\?`	显示所有元命令帮助
`\h [command]`	显示 SQL 命令帮助

二、数据库操作

2.1 创建数据库

```
-- 创建数据库
CREATE DATABASE database_name;

-- 带参数创建数据库
CREATE DATABASE database_name
    WITH
    OWNER = username
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

-- 从模板创建数据库
CREATE DATABASE new_db TEMPLATE template_db;
```

2.2 修改数据库

```
-- 修改数据库所有者
ALTER DATABASE database_name OWNER TO new_owner;

-- 修改数据库名称
ALTER DATABASE old_name RENAME TO new_name;

-- 修改连接限制
ALTER DATABASE database_name CONNECTION LIMIT 100;

-- 设置数据库参数
ALTER DATABASE database_name SET parameter_name = value;
```

2.3 删除数据库

```
-- 删除数据库
DROP DATABASE database_name;

-- 强制删除（终止连接后删除）
DROP DATABASE database_name WITH (FORCE);
```

> **注意**
>
> 删除数据库前请确保没有活动连接，否则删除会失败。使用 `WITH (FORCE)` 选项可以强制终止所有连接后删除。

三、表操作

3.1 创建表

```
-- 基本创建表
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE
);

-- 带外键的表
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    total_amount DECIMAL(10, 2),
    order_date DATE DEFAULT CURRENT_DATE
);

-- 创建临时表
CREATE TEMP TABLE temp_data AS
SELECT * FROM users WHERE is_active = TRUE;
```

常用数据类型

表 2 PostgreSQL 常用数据类型

类型	说明	示例
`INTEGER / INT`	4字节整数	42
`BIGINT`	8字节大整数	9223372036854775807
`SMALLINT`	2字节小整数	32767
`NUMERIC(p,s)`	精确小数	NUMERIC(10,2)
`VARCHAR(n)`	变长字符串	'Hello'
`TEXT`	无限制文本	长文本内容
`TIMESTAMP`	日期时间	'2026-02-28 10:30:00'
`DATE`	日期	'2026-02-28'

`BOOLEAN`	布尔值	TRUE/FALSE
`JSON / JSONB`	JSON数据	'{"key": "value"}'
`ARRAY`	数组	ARRAY[1,2,3]
`UUID`	UUID	'a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11'

3.2 修改表

```
-- 添加列
ALTER TABLE users ADD COLUMN phone VARCHAR(20);

-- 删除列
ALTER TABLE users DROP COLUMN phone;

-- 修改列类型
ALTER TABLE users ALTER COLUMN phone TYPE VARCHAR(30);

-- 修改列名
ALTER TABLE users RENAME COLUMN phone TO mobile;

-- 添加 NOT NULL 约束
ALTER TABLE users ALTER COLUMN email SET NOT NULL;

-- 删除 NOT NULL 约束
ALTER TABLE users ALTER COLUMN email DROP NOT NULL;

-- 添加默认值
ALTER TABLE users ALTER COLUMN is_active SET DEFAULT FALSE;

-- 添加主键
ALTER TABLE users ADD CONSTRAINT pk_users PRIMARY KEY (id);

-- 添加外键
ALTER TABLE orders ADD CONSTRAINT fk_orders_users
    FOREIGN KEY (user_id) REFERENCES users(id);

-- 删除约束
ALTER TABLE users DROP CONSTRAINT constraint_name;

-- 重命名表
ALTER TABLE users RENAME TO customers;
```

3.3 删除表

```
-- 删除表
DROP TABLE table_name;

-- 如果存在则删除
DROP TABLE IF EXISTS table_name;

-- 级联删除（删除依赖对象）
DROP TABLE table_name CASCADE;
```

四、数据操作

4.1 插入数据

```
-- 插入单行
INSERT INTO users (username, email)
VALUES ('john_doe', 'john@example.com');

-- 插入多行
INSERT INTO users (username, email) VALUES
    ('alice', 'alice@example.com'),
    ('bob', 'bob@example.com'),
    ('charlie', 'charlie@example.com');

-- 插入并返回数据
INSERT INTO users (username, email)
VALUES ('dave', 'dave@example.com')
RETURNING id, created_at;

-- 从其他表插入
INSERT INTO users_backup
SELECT * FROM users WHERE is_active = FALSE;
```

4.2 更新数据

```
-- 更新单条记录
UPDATE users
SET email = 'newemail@example.com'
WHERE id = 1;

-- 更新多条记录
UPDATE users
SET is_active = FALSE
WHERE created_at < '2025-01-01';

-- 更新并返回
UPDATE users
SET email = 'updated@example.com'
WHERE id = 1
RETURNING *;

-- 使用子查询更新
UPDATE users
SET status = 'premium'
WHERE id IN (SELECT user_id FROM orders WHERE total > 1000);
```

4.3 删除数据

```
-- 删除指定记录
DELETE FROM users WHERE id = 1;

-- 删除多条记录
DELETE FROM users WHERE is_active = FALSE;

-- 删除所有记录（谨慎使用）
DELETE FROM users;

-- 清空表并重置自增（更快）
TRUNCATE TABLE users RESTART IDENTITY;

-- 级联清空相关表
TRUNCATE TABLE users, orders CASCADE;
```

> **提示**
>
> `TRUNCATE` 比 `DELETE` 更快，因为它不记录单行删除操作。但 `TRUNCATE` 不能带 WHERE 条件。

五、查询操作

5.1 基本查询

```
-- 查询所有列
SELECT * FROM users;

-- 查询指定列
SELECT username, email FROM users;

-- 去重查询
SELECT DISTINCT status FROM users;

-- 条件查询
SELECT * FROM users WHERE is_active = TRUE;

-- 多条件查询
SELECT * FROM users
WHERE is_active = TRUE AND created_at > '2025-01-01';

-- 范围查询
SELECT * FROM users WHERE id BETWEEN 1 AND 100;

-- IN 查询
SELECT * FROM users WHERE status IN ('active', 'pending');

-- 模糊查询
SELECT * FROM users WHERE username LIKE '%john%';
```

5.2 排序与分页

```
-- 排序
SELECT * FROM users ORDER BY created_at DESC;

-- 多字段排序
SELECT * FROM users ORDER BY status ASC, created_at DESC;

-- 限制结果数量
SELECT * FROM users LIMIT 10;

-- 分页查询
SELECT * FROM users ORDER BY id LIMIT 10 OFFSET 20;

-- 简写分页（PostgreSQL 特有）
SELECT * FROM users ORDER BY id FETCH FIRST 10 ROWS ONLY;
SELECT * FROM users ORDER BY id OFFSET 20 ROWS FETCH NEXT 10 ROWS ONLY;
```

5.3 聚合查询

```
-- 计数
SELECT COUNT(*) FROM users;
SELECT COUNT(DISTINCT status) FROM users;

-- 求和、平均值、最大最小值
SELECT
    SUM(total_amount) as total_sales,
    AVG(total_amount) as avg_sales,
    MAX(total_amount) as max_sale,
    MIN(total_amount) as min_sale
FROM orders;

-- 分组统计
SELECT status, COUNT(*) as count
FROM users
GROUP BY status;

-- 分组过滤
SELECT status, COUNT(*) as count
FROM users
GROUP BY status
HAVING COUNT(*) > 10;
```

5.4 连接查询

```
-- 内连接
SELECT u.username, o.total_amount
FROM users u
INNER JOIN orders o ON u.id = o.user_id;

-- 左连接
SELECT u.username, o.total_amount
FROM users u
LEFT JOIN orders o ON u.id = o.user_id;

-- 右连接
SELECT u.username, o.total_amount
FROM users u
RIGHT JOIN orders o ON u.id = o.user_id;

-- 全外连接
SELECT u.username, o.total_amount
FROM users u
FULL OUTER JOIN orders o ON u.id = o.user_id;

-- 交叉连接
SELECT * FROM users CROSS JOIN roles;
```

5.5 子查询

```
-- WHERE 子查询
SELECT * FROM users
WHERE id IN (SELECT user_id FROM orders WHERE total_amount > 1000);

-- EXISTS 子查询
SELECT * FROM users u
WHERE EXISTS (SELECT 1 FROM orders o WHERE o.user_id = u.id);

-- 相关子查询
SELECT * FROM users u
WHERE (SELECT COUNT(*) FROM orders o WHERE o.user_id = u.id) > 5;

-- WITH 子句（CTE）
WITH active_users AS (
    SELECT * FROM users WHERE is_active = TRUE
)
SELECT * FROM active_users WHERE created_at > '2025-01-01';
```

六、用户与权限管理

6.1 用户管理

```
-- 创建用户
CREATE USER username WITH PASSWORD 'password';

-- 创建超级用户
CREATE USER admin WITH PASSWORD 'password' SUPERUSER;

-- 修改用户密码
ALTER USER username WITH PASSWORD 'newpassword';

-- 重命名用户
ALTER USER username RENAME TO newname;

-- 删除用户
DROP USER username;

-- 删除用户及其拥有的对象
DROP USER username CASCADE;
```

6.2 权限管理

```
-- 授予表权限
GRANT SELECT, INSERT, UPDATE ON table_name TO username;

-- 授予所有权限
GRANT ALL PRIVILEGES ON table_name TO username;

-- 授予数据库权限
GRANT CREATE ON DATABASE database_name TO username;

-- 授予 Schema 权限
GRANT USAGE ON SCHEMA schema_name TO username;
GRANT CREATE ON SCHEMA schema_name TO username;

-- 授予序列权限
GRANT USAGE, SELECT ON SEQUENCE sequence_name TO username;

-- 撤销权限
REVOKE INSERT ON table_name FROM username;

-- 查看权限
\dp table_name        -- 查看表权限
\du username          -- 查看用户权限
```

6.3 角色管理

```
-- 创建角色
CREATE ROLE readonly;

-- 给角色授权
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;

-- 将角色授予用户
GRANT readonly TO username;

-- 设置默认角色
ALTER USER username SET ROLE readonly;

-- 删除角色
DROP ROLE readonly;
```

七、索引管理

7.1 创建索引

```
-- 单列索引
CREATE INDEX idx_users_email ON users(email);

-- 多列索引
CREATE INDEX idx_users_name_email ON users(username, email);

-- 唯一索引
CREATE UNIQUE INDEX idx_users_email_unique ON users(email);

-- 部分索引
CREATE INDEX idx_active_users ON users(email) WHERE is_active = TRUE;

-- 函数索引
CREATE INDEX idx_users_lower_email ON users(LOWER(email));

-- GIN 索引（用于数组、JSONB）
CREATE INDEX idx_users_tags ON users USING GIN(tags);

-- GiST 索引（用于几何数据、全文搜索）
CREATE INDEX idx_docs_content ON documents USING GiST(content);
```

7.2 管理索引

```
-- 查看表的所有索引
\di table_name

-- 查看索引定义
\d index_name

-- 重命名索引
ALTER INDEX old_name RENAME TO new_name;

-- 删除索引
DROP INDEX index_name;

-- 重建索引
REINDEX INDEX index_name;

-- 分析索引
ANALYZE table_name;

-- 查看索引使用情况
SELECT * FROM pg_stat_user_indexes WHERE relname = 'users';
```

八、备份与恢复

8.1 使用 pg_dump 备份

```
# 备份整个数据库
pg_dump -U postgres -d mydb > mydb_backup.sql

# 备份为自定义格式（可压缩、可选择性恢复）
pg_dump -U postgres -d mydb -Fc > mydb_backup.dump

# 备份特定表
pg_dump -U postgres -d mydb -t users -t orders > tables_backup.sql

# 备份时排除特定表
pg_dump -U postgres -d mydb -T temp_table > backup.sql

# 压缩备份
pg_dump -U postgres -d mydb | gzip > mydb_backup.sql.gz

# 远程备份
pg_dump -h remote_host -U postgres -d mydb > remote_backup.sql
```

8.2 使用 pg_restore 恢复

```
# 恢复 SQL 格式备份
psql -U postgres -d mydb < mydb_backup.sql

# 恢复自定义格式备份
pg_restore -U postgres -d mydb mydb_backup.dump

# 仅恢复特定表
pg_restore -U postgres -d mydb -t users mydb_backup.dump

# 恢复到新数据库
pg_restore -U postgres -C -d postgres mydb_backup.dump

# 查看备份内容
pg_restore -l mydb_backup.dump
```

8.3 全局备份（pg_dumpall）

```
# 备份所有数据库和全局对象
pg_dumpall -U postgres > all_databases.sql

# 仅备份全局对象（用户、表空间等）
pg_dumpall -U postgres -g > globals.sql
```

8.4 物理备份（WAL 归档）

```
# 创建基础备份
pg_basebackup -U postgres -D /backup/path -Fp -Xs -P -v

# 参数说明：
# -D: 备份目录
# -Fp: 普通格式（非 tar）
# -Xs: 流式 WAL
# -P: 显示进度
# -v: 详细输出
```

九、系统维护

9.1 数据库统计

```
-- 更新表统计信息
ANALYZE table_name;

-- 更新所有表的统计信息
ANALYZE;

-- 详细分析
ANALYZE VERBOSE table_name;
```

9.2 表维护

```
-- 清理死元组
VACUUM table_name;

-- 清理并分析
VACUUM ANALYZE table_name;

-- 完全清理（需要排他锁）
VACUUM FULL table_name;

-- 自动清理所有表
VACUUM;

-- 查看表膨胀情况
SELECT
    schemaname,
    relname,
    n_dead_tup,
    n_live_tup,
    round(n_dead_tup::numeric/nullif(n_live_tup,0)*100, 2) as dead_pct
FROM pg_stat_user_tables
ORDER BY n_dead_tup DESC;
```

9.3 监控与诊断

```
-- 查看当前活动连接
SELECT * FROM pg_stat_activity;

-- 查看慢查询
SELECT * FROM pg_stat_activity
WHERE state = 'active' AND now() - query_start > interval '5 seconds';

-- 查看锁等待
SELECT * FROM pg_locks WHERE NOT granted;

-- 查看数据库大小
SELECT pg_size_pretty(pg_database_size('mydb'));

-- 查看表大小
SELECT pg_size_pretty(pg_total_relation_size('users'));

-- 查看表空间使用情况
SELECT spcname, pg_size_pretty(pg_tablespace_size(spcname))
FROM pg_tablespace;
```

9.4 配置参数

```
-- 查看所有参数
SHOW ALL;

-- 查看特定参数
SHOW max_connections;
SHOW shared_buffers;

-- 修改参数（当前会话）
SET work_mem = '256MB';

-- 修改参数（当前事务）
SET LOCAL work_mem = '256MB';

-- 重置参数
RESET work_mem;

-- 查看配置文件位置
SHOW config_file;
SHOW hba_file;
```

十、常用系统表与视图

10.1 系统目录表

表 3 常用系统目录表

表名	说明
`pg_database`	数据库信息
`pg_tables`	所有表信息
`pg_indexes`	所有索引信息
`pg_user`	用户信息
`pg_roles`	角色信息

`pg_constraint`	约束信息
`pg_trigger`	触发器信息
`pg_proc`	函数/存储过程信息
`pg_class`	表、索引、序列等对象
`pg_attribute`	表的列信息

10.2 统计视图

表 4 常用统计视图

视图名	说明
`pg_stat_activity`	当前活动连接和查询
`pg_stat_database`	数据库级统计信息
`pg_stat_user_tables`	用户表统计信息
`pg_stat_user_indexes`	用户索引统计信息
`pg_stat_statements`	SQL 语句统计（需安装扩展）
`pg_locks`	当前锁信息
`pg_stat_replication`	流复制统计

10.3 常用查询示例

```
-- 查看所有表及其大小
SELECT
    schemaname,
    relname as table_name,
    pg_size_pretty(pg_total_relation_size(relid)) as total_size
FROM pg_stat_user_tables
ORDER BY pg_total_relation_size(relid) DESC;

-- 查看所有索引及其大小
SELECT
    schemaname,
    relname as table_name,
    indexrelname as index_name,
    pg_size_pretty(pg_relation_size(indexrelid)) as index_size
FROM pg_stat_user_indexes
ORDER BY pg_relation_size(indexrelid) DESC;

-- 查看未使用索引
SELECT
    schemaname,
    relname as table_name,
    indexrelname as index_name
FROM pg_stat_user_indexes
WHERE idx_scan = 0;

-- 查看表的外键
SELECT
    tc.table_name,
    kcu.column_name,
    ccu.table_name AS foreign_table_name,
    ccu.column_name AS foreign_column_name
FROM information_schema.table_constraints tc
JOIN    information_schema.key_column_usage    kcu    ON    tc.constraint_name    =
kcu.constraint_name
JOIN  information_schema.constraint_column_usage  ccu  ON  ccu.constraint_name  =
tc.constraint_name
WHERE tc.constraint_type = 'FOREIGN KEY';
```

本文档汇总了 PostgreSQL 数据库管理中最常用的命令和操作

建议结合官方文档使用：https://www.postgresql.org/docs/