PostgreSQL 常用命令速查表

不剪发的 Tony 老师

PostgreSQL 是世界上最先进的开源对象-关系型数据库，遵循类似 BSD/MIT 的开源协议，以其强大功能、高性能、可靠性以及可扩展性在企业中被广泛应用。PostgreSQL 支持各种主流的平台，例如 Linux、BSD、AIX、HP-UX、Mac OS X 以及 Windows 等。PostgreSQL 遵循事务的 ACID 原则，高度兼容 SQL 标准；同时支持自定义数据类型、索引类型、过程语言扩展（PL/pgSQL、PL/Python、PL/Java 等）。

连接服务器

```
psql -h hostname -p port -U username dbname
postgres=# \? [ topic ]              -- 获取 psql 工具帮助
postgres=# \h command;               -- 获取 SQL 命令帮助
postgres=# \conninfo                 -- 显示当前连接信息
postgres=# \l[+]                     -- 列出所有的数据库
postgres=# \q                        -- 退出 psql 客户端
SELECT version();                    -- 查看服务器版本
SHOW { ALL | name }                  -- 查看运行时参数
SET [ SESSION | LOCAL ] name
TO { value | 'value' | DEFAULT }     -- 设置运行时参数
SELECT pg_reload_conf();             -- 重新加载配置文件
SELECT * FROM pg_stat_activity;      -- 查询所有连接
SELECT pg_cancel_backend(pid); -- 取消运行中的查询
SELECT pg_terminate_backend(pid);  -- 终止指定连接
```

角色、用户和组

```
CREATE USER name PASSWORD 'password';
                                     -- 创建用户
SELECT * FROM pg_user;               -- 查看所有用户
SELECT user, current_user;           -- 查看当前用户
ALTER USER name PASSWORD 'password' ;
                                     -- 修改密码
ALTER USER name VALID UNTIL 'timestamp' ;
                                     -- 设置密码失效时间
DROP USER [IF EXISTS] name;          -- 删除用户
GRANT privilege_list | ALL           -- 授予权限
ON object_type TO user;
REVOKE privilege_list | ALL          -- 撤销权限
ON object_type FROM user;
SELECT * FROM role_table_grants;     -- 查看表的授权
```

备注：PostgreSQL 角色又包含了两种概念，具有登录权限的角色称为用户，包含其他成员的角色称为组（group）。因此，以上语句中 USER 关键字可以替换为 ROLE 或者 GROUP。

数据库和模式

```
CREATE DATABASE name
[WITH OWNER = user];                 -- 创建数据库
SELECT * FROM pg_database;           -- 查看数据库
ALTER DATABASE name ...;             -- 修改数据库
DROP DATABASE [IF EXISTS] name;      -- 删除数据库
postgres=# \dn                       -- 查看当前数据库中的模式
SELECT * FROM pg_namespace;          -- 查看所有模式
CREATE SCHEMA [IF NOT EXISTS] name
[ AUTHORIZATION user ];              -- 创建模式
ALTER SCHEMA name RENAME TO new_name;
ALTER SCHEMA name OWNER TO new_owner;
                                     -- 修改模式
SHOW search_path;                    -- 查看模式搜索路径
SET search_path TO schema1, shema2, ...;
                                     -- 设置模式搜索路径
DROP SCHEMA [ IF EXISTS ] name;      -- 删除模式
```

PostgreSQL 常用命令速查表

不剪发的 Tony 老师

PostgreSQL 常用的数据类型包括： CHAR(n)、VARCHAR(n) 、TEXT 等字符类型，SMALLINT、INTEGER、BIGINT、NUMERIC (p, s)、REAL、DOUBLE PRECISION 等数字类型，DATE、TIME、TIMESTAMP 等日期时间类型。

PostgreSQL 支持 SQL 标准中的所有字段和表级完整性约束，包括：主键约束、外键约束、唯一约束、非空约束、检查约束以及默认值。

管理数据表

```
CREATE TABLE [IF NOT EXISTS] name          -- 创建表
( column_name data_type column_constraint,
  ...,
  table_constraint );

CREATE TABLE [IF NOT EXISTS] name          -- 复制表
AS SELECT ...;

SELECT *
FROM information_schema.tables     -- 查看所有的表
WHERE table_type = 'BASE TABLE';

postgres=# \d table_name;                  -- 查看表结构

ALTER TABLE table_name                     -- 增加字段
ADD COLUMN name data_type column_constraint;

ALTER TABLE table_name                     -- 修改字段类型
ALTER COLUMN name TYPE data_type;

ALTER TABLE table_name                     -- 修改字段名称
RENAME COLUMN name TO new_name;

ALTER TABLE table_name
DROP COLUMN [IF EXISTS] name;              -- 删除字段
```

管理数据表

```
ALTER TABLE [IF EXISTS] name               -- 重命名表
RENAME TO new_name;

DROP TABLE [IF EXISTS] name;               --删除表

ALTER TABLE table_name                     -- 增加约束
ADD CONSTRAINT table_constraint;

ALTER TABLE table_name                     --删除约束
DROP CONSTRAINT [IF EXISTS] name;

ALTER TABLE table_name          -- 设置/删除非空约束
ALTER COLUMN name {SET | DROP} NOT NULL;

CREATE INDEX name ON table_name      -- 创建索引
(column1 [ASC | DESC], ...);

SELECT * FROM pg_indexes;                  -- 查看索引

ALTER INDEX name RENAME TO new_name;
ALTER INDEX name SET TABLESPACE
tablespace_name;                           -- 修改索引

REINDEX name;                              -- 重建索引

DROP INDEX [IF EXISTS] name;               -- 删除索引
```

管理表空间

```
CREATE TABLESPACE name OWNER user
LOCATION  'directory' ;                    -- 创建表空间

SELECT * FROM pg_tablespace;               -- 查看表空间

ALTER TABLESPACE name ...;                 -- 修改表空间

DROP TABLESPACE [IF EXISTS] name;          -- 删除表空间
```

备份与还原

```
pg_dump db_name > file_name.sql      -- 备份数据库
pg_dump -Fc db_name -f file_name.dmp
pg_dump -Fd db_name -f file_dir
pg_dump -Ft db_name -f file_name.tar

psql newdb -f file_name.sql                -- 还原数据库
pg_restore -d newdb file_name.dmp
pg_restore -d newdb file_dir
pg_restore -d newdb file_name.tar

pg_dumpall -f cluster.sql;                 -- 备份整个集群

COPY table_name FROM  'filename' ;         -- 导入表

COPY table_name TO  'filename' ;           -- 导出表
```

PostgreSQL 常用命令速查表

不剪发的 Tony 老师

常用的查询条件包括：=、!=、<>、<、<=、>、>=、BETWEEN、IN、EXISTS、LIKE、AND、OR、NOT、IS [NOT] NULL 等。

常用的聚合函数：AVG、COUNT、MIN、MAX、SUM、STRING_AGG 等。

除了基本的分组操作之外，PostgreSQL 还支持 3 种高级的分组选项：GROUPING SETS、ROLLUP 以及 CUBE。

查询语句

```
SELECT col1, col2, ... FROM t;                  -- 单表查询

SELECT * FROM t;                                -- 查询所有字段

SELECT DISTINCT col1, col2, ...                 -- 排除重复结果
FROM t;

SELECT col1, col2, ...
FROM t
WHERE conditions;                               -- 使用查询条件

SELECT col1, col2, ...
FROM t
ORDER BY col1 ASC, col2 DESC;                   -- 排序显示

SELECT col1, col2, ...
FROM t
ORDER BY col1 ASC, col2 DESC
OFFSET m ROWS
FETCH NEXT n ROWS ONLY;                         -- 限定数量

SELECT col1, col2, agg_func()                   -- 聚合函数
FROM t
GROUP BY col1, col2                             -- 分组操作
HAVING conditions;                              -- 分组过滤
```

复杂查询

```
SELECT t1.col1, t2.col1, ...                    -- 连接查询
FROM table1 AS t1                                  -- 别名
INNER | LEFT | RIGHT | FULL | CROSS JOIN t2
ON conditions;

SELECT t.col1, t.col2, ...                      -- 子查询
FROM (SELECT ...) t;                               -- 派生表

SELECT t1.col1, t1.col2, ...                    -- 关联子查询
FROM t1
WHERE EXISTS ( SELECT 1
               FROM t2
               WHERE t2.col1 = t1.col1);

SELECT col1, col2, ...                          -- 集合运算
FROM t1
UNION | INTERSECT | EXCEPT [ALL]
SELECT c1, c2, ...
FROM t2;

WITH RECURSIVE cte AS (                         -- 通用表表达式
   SELECT ...                                      -- 初始查询
   UNION ALL
   SELECT ...)                                  --递归查询，可以引用 cte 自身
SELECT * FROM cte;
```

DML 语句

```
INSERT INTO table(col1,col2,...)                -- 插入数据
VALUES (val1,val2,...)
[RETUNRING ...];

INSERT INTO table(col1,col2)                    -- 插入多条记录
VALUES (val11,val12), (val21,val22), ...;

INSERT INTO table(col1,col2,...)                -- 插入查询结果
SELECT ...;

UPDATE table                                    -- 更新数据
SET col1 = val1, col2 = val2, ...
WHERE conditions
[RETUNRING ...];

DELETE FROM table                               -- 删除数据
WHERE conditions
[RETUNRING ...];

TRUNCATE [TABLE] table;                         --快速清空表

INSERT INTO table(col1,col2,...)                -- 合并数据
...
ON CONFLICT [conflict_target]
DO NOTHING | DO UPDATE SET ...;
```

PostgreSQL 常用命令速查表

不剪发的 Tony 老师

视图是预定义的查询语句，可以用于替代复杂查询，减少复杂性；提供一致性接口，实现业务规则；控制对于表的访问，提高访问安全性。

存储过程/函数是一种存储在数据库中的程序，可以减少应用和数据库之间的网络传输，提高应用的性能，实现代码的可重用性。

触发器是一种特殊的函数，当表中的数据发生变化时自动触发并执行预定义的操作。

事务控制

```
BEGIN;                                  -- 开始一个事务
COMMIT;                                 -- 提交事务
ROLLBACK;                               -- 回滚事务
SAVEPOINT name;                         -- 设置事务保存点
ROLLBACK TO SAVEPOINT name;             -- 回滚到保存点
RELEASE SAVEPOINT name;                 -- 释放保存点
```

视图

```
CREATE [OR REPLACE] VIEW name           -- 创建视图
AS SELECT ...
[WITH CHECK OPTION];

SELECT view_definition                  -- 查看视图定义
FROM information_schema.views ;

ALTER VIEW [IF EXISTS] name
RENAME TO  new_name;                    -- 重命名视图

DROP VIEW [IF EXISTS] name;             -- 删除视图
```

存储过程/函数

```
CREATE OR REPLACE PROCEDURE
name ( argmode argname argtype, ...)
AS $$
procedure_body$$
LANGUAGE plpgsql;                       -- 创建存储过程

CALL procedure_name (argument, ...);    -- 调用过程

CREATE OR REPLACE FUNCTION
name ([parameter, ...])
RETURNS type
AS $$
procedure_body$$
LANGUAGE plpgsql;                       -- 创建存储函数

SELECT function_name(argument, ...);    -- 使用函数

ALTER PROCEDURE | FUNCTION name
   RENAME TO new_name;

DROP PROCEDURE | FUNCTION
[IF EXISTS] name;                       -- 删除存储过程/函数
```

触发器

```
CREATE OR REPLACE FUNCTION name()
RETURNS trigger                         -- 创建触发器函数
...;

CREATE TRIGGER name                     -- 创建触发器
BEFORE | AFTER | INSTEAD OF event
  ON table_name
  FOR EACH {ROW | STATEMENT}
  WHEN ( condition )
  EXECUTE FUNCTION trigger_function;

SELECT *                                -- 查看触发器
FROM information_schema.triggers ;

ALTER TRIGGER name ON table_name
RENAME TO new_name;                     --重命名触发器

ALTER TABLE table_name                  -- 启用/禁用触发器
ENABLE | DISABLE TRIGGER name;


DROP TRIGGER [IF EXISTS] name
ON table_name;                          -- 删除触发器
```