学习

postgresql

#postgresql

1: postgresql

postgresql。

postgresql。 postgresql。

		EOL
9.6	2016929	202191
9.5	201617	202111
9.4		2019121
9.3	201399	201891
9.2	2012910	201791
9.1	2011-09-12	201691
9	2010-09-20	201591
8.4	2009-07-01	201471

Examples

GNU + Linux

GNU + LinuxPostgreSQL。

https //yum.postgresql.org/repopackages.php

```
yum -y install https://download.postgresql.org/pub/repos/yum/X.X/redhat/rhel-7-x86_64/pgdg-
redhatXX-X.X-X.noarch.rpm
```

```
yum list available | grep postgres*
```

postgresqlXX postgresqlXX-server postgresqlXX-libs postgresqlXX-contrib

yum -y install postgresqlXX postgresqlXX-server postgresqlXX-libs postgresqlXX-contrib

postgres。 pg_ctl。

```
sudo -su postgres
./usr/pgsql-X.X/bin/pg_ctl -D /var/lib/pgsql/X.X/data start
```

CLIDB`psql`

Debian

Debian

```
sudo apt-get install postgresql
```

PostgreSQL。

。

PostgreSQLPGDG Yum。 。

OSXMacPortsPostgreSQL

OSXPostgreSQL。

。

```
sudo port list | grep "^postgresql[[:digit:]]\{2\}[[:space:]]"
```

```
postgresql80                   @8.0.26         databases/postgresql80
postgresql81                   @8.1.23         databases/postgresql81
postgresql82                   @8.2.23         databases/postgresql82
postgresql83                   @8.3.23         databases/postgresql83
postgresql84                   @8.4.22         databases/postgresql84
postgresql90                   @9.0.23         databases/postgresql90
postgresql91                   @9.1.22         databases/postgresql91
postgresql92                   @9.2.17         databases/postgresql92
postgresql93                   @9.3.13         databases/postgresql93
postgresql94                   @9.4.8          databases/postgresql94
postgresql95                   @9.5.3          databases/postgresql95
postgresql96                   @9.6beta2       databases/postgresql96
```

9.6PostgreSQL。

```
sudo port install postgresql96-server postgresql96
```

```
--->  Computing dependencies for postgresql96-server
--->  Dependencies to be installed: postgresql96
--->  Fetching archive for postgresql96
--->  Attempting to fetch postgresql96-9.6beta2_0.darwin_15.x86_64.tbz2 from
https://packages.macports.org/postgresql96
--->  Attempting to fetch postgresql96-9.6beta2_0.darwin_15.x86_64.tbz2.rmd160 from
https://packages.macports.org/postgresql96
--->  Installing postgresql96 @9.6beta2_0
--->  Activating postgresql96 @9.6beta2_0

To use the postgresql server, install the postgresql96-server port
```

```
--->  Cleaning postgresql96
--->  Fetching archive for postgresql96-server
--->  Attempting to fetch postgresql96-server-9.6beta2_0.darwin_15.x86_64.tbz2 from
https://packages.macports.org/postgresql96-server
--->  Attempting to fetch postgresql96-server-9.6beta2_0.darwin_15.x86_64.tbz2.rmd160 from
https://packages.macports.org/postgresql96-server
--->  Installing postgresql96-server @9.6beta2_0
--->  Activating postgresql96-server @9.6beta2_0

To create a database instance, after install do
 sudo mkdir -p /opt/local/var/db/postgresql96/defaultdb
 sudo chown postgres:postgres /opt/local/var/db/postgresql96/defaultdb
 sudo su postgres -c '/opt/local/lib/postgresql96/bin/initdb -D
/opt/local/var/db/postgresql96/defaultdb'

--->  Cleaning postgresql96-server
--->  Computing dependencies for postgresql96
--->  Cleaning postgresql96
--->  Updating database of binaries
--->  Scanning binaries for linking errors
--->  No broken files found.
```

。

```
sudo mkdir -p /opt/local/var/db/postgresql96/defaultdb
sudo chown postgres:postgres /opt/local/var/db/postgresql96/defaultdb
sudo su postgres -c '/opt/local/lib/postgresql96/bin/initdb -D
/opt/local/var/db/postgresql96/defaultdb'
```

```
sudo port load -w postgresql96-server
```

```
su postgres -c psql
```

postgres

```
psql (9.6.1)
Type "help" for help.

postgres=#
```

。

```
postgres=#SELECT setting FROM pg_settings WHERE name='data_directory';
```

```
                setting
------------------------------------------
/opt/local/var/db/postgresql96/defaultdb
(1 row)
postgres=#
```

\ q

```
postgres=#\q
```

shell。

OS / XPostgreSQL。

Mac OSXPostgres.app

Postgres.appMacPostgreSQL。
PostgreSQL。

WindowsPostgreSQL

UnixLinuxBSDWindowsPostgreSQL。

EnterpriseDBWindows http //www.enterprisedb.com/products-services-training/pgdownload PostgreSQLWindows。

Beta9.5.3。 Win x86-6432WindowsWin x86-32。

Beta。 。

- > - > - >Windows3264“## - bit Operating System”。 Windows 7Windows。

。

- pgAdmin https://www.pgadmin.org GUI。 9.6。
- PostGIS http://postgis.net GPSGIS。
- PL / PythonPL / PerlPL / Tcl。
- pgAgentpgBouncerSlony。

“Application Stack Builder”。

PL / V8 PL / Lua PL / Java。

pgAdmin。 “PostgreSQL 9.5localhost5432。

PostgreSQLUp and Running2nd Edition http://shop.oreilly.com/product/0636920032144.do 。

PostgreSQL。 Web。 。

PostgreSQLPCPostegreSQL。

。

“”“”。 ...

“” - >“” - >“”。

“”“” - >“”。

“”“”。

postgresql-x ## - 9.“postgresql-x64-9.5”。

postgresProperties - > Startup type - > Manual - > Apply - > OK。 。

PostgreSQL“pgbouncer”“PostgreSQL Scheduling Agent - pgAgent”PostgreSQL。 。 PostgreSQL。

“”。

“”。 。 。 Windows。

postgres“”。

。

EDB PostgreSQLPostgreSQLpythonEBD 。

Macbrewpostgresql

Homebrew' *macOS* '。 。 `brew`PostgreSQL

```
brew update
brew install postgresql
```

Homebrew。 `brew search postgresql`。 PostgreSQL`brew info postgresql`。 Homebrew。

```
brew services start postgresql
```

PostgreSQL

```
psql
```

psql`createdb` 。

LinuxSourcePostgreSQL

- GNU Make Version> 3.80
- ISO / ANSI Cgcc
- targzip
- zlib-devel
- readline-devel oder libedit-devel

9.6.3

```
tar -xzvf postgresql-9.6.3.tar.gz
```

PostgreSQL

- `--prefix=PATH--prefix=PATH`
- `--exec-prefix=PATH` architectur-dependet`--exec-prefix=PATH`
- `--bindir=PATH--bindir=PATH`
- `--sysconfdir=PATH--sysconfdir=PATH`
- `--with-pgport=NUMBER`
- `--with-perl` add perl support
- `--with-python`python
- `--with-openssl`openssl
- `--with-ldap`ldap
- `--with-blocksize=BLOCKSIZE`KBpagesize
  - `BLOCKSIZE`2132
- `--with-wal-segsize=SEGSIZE`WAL-SegmentMB
  - `SEGSIZE`1642

cofigure

```
./configure --exec=/usr/local/pgsql
```

`make`

`make install`PostgreSQL

`make clean`

`cd contrib` `make``make install`

postgresql https://riptutorial.com/zh-CN/postgresql/topic/885/postgresql

2: EXTENSION dblinkpostgres_fdw

- dblink'dbname = name_db_distance port = PortOfDB host = HostOfDB user = usernameDB password = passwordDB''MY QUESRY'
- dbname =
- port =
- host =
- user =
- password ='
- =SELECTINSERT...

Examples

dblink

dblink EXTENSION

1 - dblink

```
CREATE EXTENSION dblink;
```

2 -

```
SELECT * FROM
dblink ('dbname = bd_distance port = 5432 host = 10.6.6.6 user = username
password = passw@rd', 'SELECT id, code FROM schema.table')
AS newTable(id INTEGER, code character varying);
```

FDW

FDWdblink

1 -

```
CREATE EXTENSION postgres_fdw;
```

2 -

```
CREATE SERVER name_srv FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host 'hostname',
dbname 'bd_name', port '5432');
```

3 - postgres

```
CREATE USER MAPPING FOR postgres SERVER name_srv OPTIONS(user 'postgres', password
'password');
```

4 -

```
CREATE FOREIGN TABLE table_foreign (id INTEGER, code character varying)
SERVER name_srv OPTIONS(schema_name 'schema', table_name 'table');
```

5 -

```
SELECT * FROM table_foreign;
```

db。

1. EXTENSION

```
    CREATE EXTENSION postgres_fdw;
```

2.

```
    CREATE SERVER server_name FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host 'host_ip',
    dbname 'db_name', port 'port_number');
```

3.

```
    CREATE USER MAPPING FOR CURRENT_USER
    SERVER server_name
    OPTIONS (user 'user_name', password 'password');
```

4. DB

```
   CREATE SCHEMA schema_name;
```

5.

```
     IMPORT FOREIGN SCHEMA schema_name_to_import_from_remote_db
     FROM SERVER server_name
     INTO schema_name;
```

6.

```
    SELECT * FROM schema_name.table_name;
```

。

EXTENSION dblinkpostgres_fdw https://riptutorial.com/zh-CN/postgresql/topic/6970/extension-dblinkpostgres-fdw

3: JSON

JSON - JavaPostgresql9.2JSON。 JSON。 ->JSON。 ->>JSON Column。

Examples

JSON

JSON`JSONB`

```
CREATE TABLE mytable (data JSONB NOT NULL);
```

```
CREATE INDEX mytable_idx ON mytable USING gin (data jsonb_path_ops);
```

。

JSON

JSON

```
CREATE TABLE mytable (data JSONB NOT NULL);
CREATE INDEX mytable_idx ON mytable USING gin (data jsonb_path_ops);
INSERT INTO mytable VALUES($$
{
    "name": "Alice",
    "emails": [
        "alice1@test.com",
        "alice2@test.com"
    ],
    "events": [
        {
            "type": "birthday",
            "date": "1970-01-01"
        },
        {
            "type": "anniversary",
            "date": "2001-05-05"
        }
    ],
    "locations": {
        "home": {
            "city": "London",
            "country": "United Kingdom"
        },
        "work": {
            "city": "Edinburgh",
            "country": "United Kingdom"
        }
    }
}
$$);
```

```
SELECT data->>'name' FROM mytable WHERE data @> '{"name":"Alice"}';
```

```
SELECT data->>'name' FROM mytable WHERE data @> '{"emails":["alice1@test.com"]}';
```

```
SELECT data->>'name' FROM mytable WHERE data @> '{"events":[{"type":"anniversary"}]}';
```

```
SELECT data->>'name' FROM mytable WHERE data @> '{"locations":{"home":{"city":"London"}}}';
```

@>->->>

WHERE@> ->->>。

```
SELECT data FROM mytable WHERE data @> '{"name":"Alice"}';
SELECT data FROM mytable WHERE data->'name' = '"Alice"';
SELECT data FROM mytable WHERE data->>'name' = 'Alice';
```

。

->

```
SELECT data->'locations'->'work' FROM mytable WHERE data @> '{"name":"Alice"}';
SELECT data->'locations'->'work'->>'city' FROM mytable WHERE data @> '{"name":"Alice"}';
```

JSONb

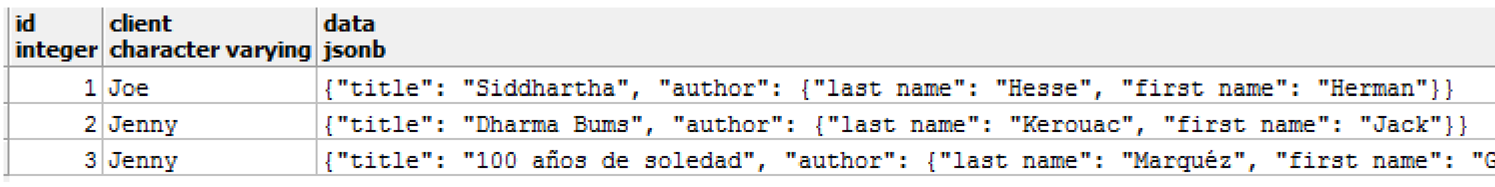
```
DROP DATABASE IF EXISTS books_db;
CREATE DATABASE books_db WITH ENCODING='UTF8' TEMPLATE template0;

DROP TABLE IF EXISTS books;

CREATE TABLE books (
  id SERIAL PRIMARY KEY,
  client TEXT NOT NULL,
  data JSONb NOT NULL
);
```

```
INSERT INTO books(client, data) values (
    'Joe',
    '{ "title": "Siddhartha", "author": { "first_name": "Herman", "last_name": "Hesse" } }'
),(
    'Jenny',
    '{ "title": "Dharma Bums", "author": { "first_name": "Jack", "last_name": "Kerouac" } }'
),(
    'Jenny',
    '{ "title": "100 años de soledad", "author": { "first_name": "Gabo", "last_name":
"Marquéz" } }'
);
```

```
SELECT * FROM books;
```

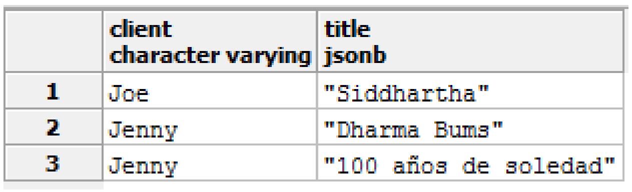


id integer	client character varying	data jsonb
1	Joe	{"title": "Siddhartha", "author": {"last name": "Hesse", "first name": "Herman"}}
2	Jenny	{"title": "Dharma Bums", "author": {"last name": "Kerouac", "first name": "Jack"}}
3	Jenny	{"title": "100 años de soledad", "author": {"last name": "Marquéz", "first name": "G

->JSON

1

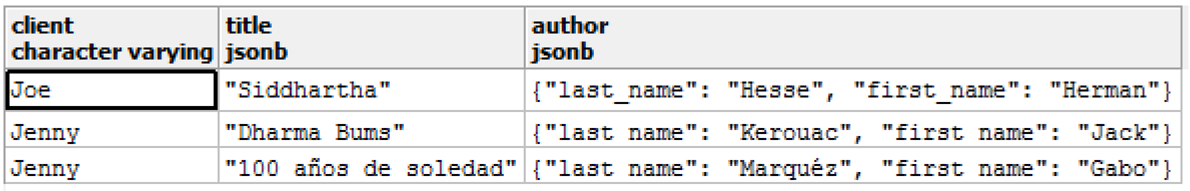
```
SELECT client,
    data->'title' AS title
    FROM books;
```



	client character varying	title jsonb
1	Joe	"Siddhartha"
2	Jenny	"Dharma Bums"
3	Jenny	"100 años de soledad"

2

```
SELECT client,
   data->'title' AS title, data->'author' AS author
   FROM books;
```



client character varying	title jsonb	author jsonb
Joe	"Siddhartha"	{"last_name": "Hesse", "first_name": "Herman"}
Jenny	"Dharma Bums"	{"last name": "Kerouac", "first name": "Jack"}
Jenny	"100 años de soledad"	{"last name": "Marquéz", "first name": "Gabo"}

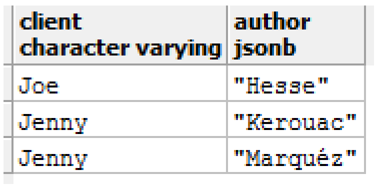
-> VS ->>

->JSON->>。

NESTED

->

```
SELECT client,
   data->'author'->'last_name' AS author
   FROM books;
```



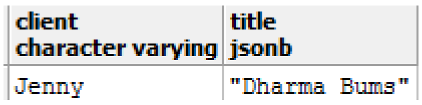
client character varying	author jsonb
Joe	"Hesse"
Jenny	"Kerouac"
Jenny	"Marquéz"

JSON

```
 SELECT
 client,
 data->'title' AS title
 FROM books
  WHERE data->'title' = '"Dharma Bums"';
```

WHERE->JSON `'"Dharma Bums"'`

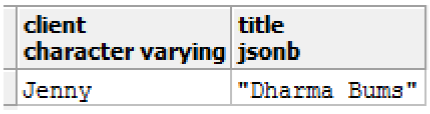
`->>'Dharma Bums'`



client character varying	title jsonb
Jenny	"Dharma Bums"

JSON

```
SELECT
 client,
 data->'title' AS title
 FROM books
  WHERE data->'author'->>'last_name' = 'Kerouac';
```



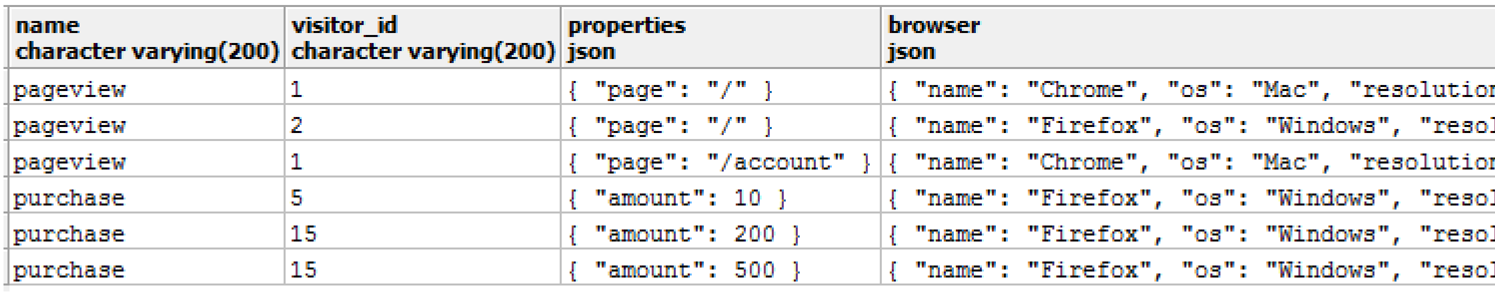
client character varying	title jsonb
Jenny	"Dharma Bums"

```
CREATE TABLE events (
  name varchar(200),
  visitor_id varchar(200),
  properties json,
  browser json
);
```

。 。 。

```
INSERT INTO events (name, visitor_id, properties, browser) VALUES
(
  'pageview', '1',
  '{ "page": "/" }',
  '{ "name": "Chrome", "os": "Mac", "resolution": { "x": 1440, "y": 900 } }'
),(
  'pageview', '2',
  '{ "page": "/" }',
  '{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1920, "y": 1200 } }'
),(
  'pageview', '1',
  '{ "page": "/account" }',
  '{ "name": "Chrome", "os": "Mac", "resolution": { "x": 1440, "y": 900 } }'
),(
  'purchase', '5',
  '{ "amount": 10 }',
  '{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1024, "y": 768 } }'
),(
  'purchase', '15',
  '{ "amount": 200 }',
  '{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1280, "y": 800 } }'
),(
  'purchase', '15',
  '{ "amount": 500 }',
  '{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1280, "y": 800 } }'
);
```

```
SELECT * FROM events;
```

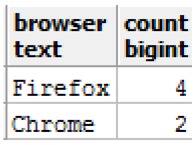


name character varying(200)	visitor_id character varying(200)	properties json	browser json
pageview	1	{ "page": "/" }	{ "name": "Chrome", "os": "Mac", "resolutio
pageview	2	{ "page": "/" }	{ "name": "Firefox", "os": "Windows", "resol
pageview	1	{ "page": "/account" }	{ "name": "Chrome", "os": "Mac", "resolutio
purchase	5	{ "amount": 10 }	{ "name": "Firefox", "os": "Windows", "resol
purchase	15	{ "amount": 200 }	{ "name": "Firefox", "os": "Windows", "resol
purchase	15	{ "amount": 500 }	{ "name": "Firefox", "os": "Windows", "resol

JSON+ PostgreSQL

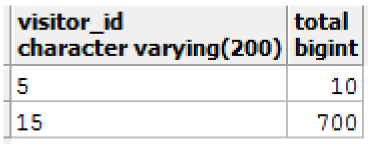
JSONPostgreSQL。 RDBMS。

-
  ```
  SELECT browser->>'name' AS browser,
    count(browser)
    FROM events
    GROUP BY browser->>'name';
  ```



browser text	count bigint
Firefox	4
Chrome	2

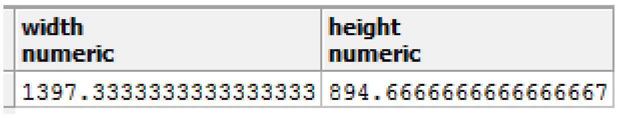
-
  ```
  SELECT visitor_id, SUM(CAST(properties->>'amount' AS integer)) AS total
  FROM events
  WHERE CAST(properties->>'amount' AS integer) > 0
  GROUP BY visitor_id;
  ```



visitor_id character varying(200)	total bigint
5	10
15	700

-

```
SELECT AVG(CAST(browser->'resolution'->>'x' AS integer)) AS width,
  AVG(CAST(browser->'resolution'->>'y' AS integer)) AS height
FROM events;
```



width numeric	height numeric
1397.3333333333333333	894.6666666666666667

。

JSON https://riptutorial.com/zh-CN/postgresql/topic/1034/json

4: postgresql

COMMMENT。

。 COMMENT。

COMMENT ON ROLECREATEROLE。

- database_object";

http //www.postgresql.org/docs/current/static/sql-comment.html

Examples

table_name IS";

TABLE;

。

postgresql https://riptutorial.com/zh-CN/postgresql/topic/8191/postgresql

5: PostgreSQL

Examples

PostgreSQL

- ◦
    - ◦
    - ◦ WAL
  - ◦ `createuser -U postgres replication -P -c 5 --replication`

    ```
    + option -P will prompt you for new password
    + option -c is for maximum connections. 5 connections are enough for replication
    + -replication will grant replication privileges to the user
    ```

  - ◦ `mkdir $PGDATA/archive`
  - ◦ **pg_hba.conf**

    ◦

    ```
        #hosttype    database_name     user_name       hostname/IP      method
         host        replication       replication     <slave-IP>/32    md5
    ```

  - ◦ **postgresql.conf**

    PostgreSQL。

    `wal_level = hot_standby`

    ◦

    ```
    `hot_standby` logs what is required to accept read only queries on slave server.

    `streaming` logs what is required to just apply the WAL's on slave.

    `archive` which logs what is required for archiving.
    ```

    `archive_mode=on`

    `archive_command`WAL。

    `archive_command = 'test ! -f /path/to/archivedir/%f && cp %p /path/to/archivedir/%f'`

    `archive_command`WAL。

    `wal_senders = 5`WAL。

    ◦

-

primay

。

。 。 。

- **pg_basebackup**

  `pg_basebackup。`

  `$ pg_basebackup -h <primary IP> -D /var/lib/postgresql/<version>/main -U replication -v -P --xlog-method=stream`

```
    -D: This is tells pg_basebackup where to the initial backup

    -h: Specifies the system where to look for the primary server

    -xlog-method=stream: This will force the pg_basebackup to open another connection and
stream enough xlog while backup is running.
                         It also ensures that fresh backup can be started without failing back
to using an archive.
```

- postgresql.confrecovery.conf。

  `hot_standby = on`

  - **recovery.conf**

    `standby_mode = on`

  - IP。 replication

    `primary_conninfo ='host = port = 5432 user = replication password ='

    `trigger_file = '/tmp/postgresql.trigger.5432'`

    `trigger_file。` “”。 pg_ctl promote。

-

PostgreSQL。 3.0。

PostgreSQL https://riptutorial.com/zh-CN/postgresql/topic/5478/postgresql

6: Postgres

Postgrespgcrypto。 CREATE EXTENSION pgcrypto;

Examples

`DIGEST()`。 。

`digest(data text, type text) returns bytea`

`digest(data bytea, type text) returns bytea`

- `SELECT DIGEST('1', 'sha1')`
- `SELECT DIGEST(CONCAT(CAST(current_timestamp AS TEXT), RANDOM()::TEXT), 'sha1')`

Postgres https://riptutorial.com/zh-CN/postgresql/topic/9230/postgres

7: Postgres

Examples

PostgresDATEADD

- `SELECT CURRENT_DATE + '1 day'::INTERVAL`
- `SELECT '1999-12-11'::TIMESTAMP + '19 days'::INTERVAL`
- `SELECT '1 month'::INTERVAL + '1 month 3 days'::INTERVAL`

```
SELECT
    string_agg(<TABLE_NAME>.<COLUMN_NAME>, ',')
FROM
    <SCHEMA_NAME>.<TABLE_NAME> T
```

postgres

```
DELETE
    FROM <SCHEMA_NAME>.<Table_NAME>
WHERE
    ctid NOT IN
        (
        SELECT
            MAX(ctid)
        FROM
            <SCHEMA_NAME>.<TABLE_NAME>
        GROUP BY
            <SCHEMA_NAME>.<TABLE_NAME>.*
        )
;
```

Postresql。

```
    update <SCHEMA_NAME>.<TABLE_NAME_1> AS A
    SET <COLUMN_1> = True
    FROM <SCHEMA_NAME>.<TABLE_NAME_2> AS B
    WHERE
        A.<COLUMN_2> = B.<COLUMN_2> AND
        A.<COLUMN_3> = B.<COLUMN_3>
```

```
select
    (
        (DATE_PART('year', AgeonDate) - DATE_PART('year', tmpdate)) * 12
        +
        (DATE_PART('month', AgeonDate) - DATE_PART('month', tmpdate))
    )
from dbo."Table1"
```

```
select (DATE_PART('year', AgeonDate) - DATE_PART('year', tmpdate)) from dbo."Table1"
```

//

```
CREATE EXTENSION DBLINK;
```

```
INSERT INTO
    <SCHEMA_NAME>.<TABLE_NAME_1>
SELECT *
FROM
    DBLINK(
    'HOST=<IP-ADDRESS> USER=<USERNAME> PASSWORD=<PASSWORD> DBNAME=<DATABASE>',
    'SELECT * FROM <SCHEMA_NAME>.<TABLE_NAME_2>')
    AS <TABLE_NAME>
    (
    <COLUMN_1> <DATATYPE_1>,
    <COLUMN_1> <DATATYPE_2>,
    <COLUMN_1> <DATATYPE_3>
    );
```

Postgres https://riptutorial.com/zh-CN/postgresql/topic/7433/postgres

8: UPDATE

Examples

column_name = value

```
UPDATE person SET planet = 'Earth';
```

```
UPDATE person SET state = 'NY' WHERE city = 'New York';
```

col=val

```
UPDATE person
   SET country = 'USA',
       state = 'NY'
WHERE city = 'New York';
```

```
UPDATE person
SET state_code = cities.state_code
FROM cities
WHERE cities.city = city;
```

person citycities city。 personstate_code。

UPDATE https://riptutorial.com/zh-CN/postgresql/topic/3136/update

9:

。

PostgreSQL

https://www.postgresql.org/docs/9.3/static/event-trigger-definition.html

Examples

DDL

-

- DDL_COMMAND_START
- DDL_COMMAND_END
- SQL_DROP

DDL_COMMAND_STARTDDL_COMMAND_START 。

```
CREATE TABLE TAB_EVENT_LOGS(
  DATE_TIME TIMESTAMP,
  EVENT_NAME TEXT,
  REMARKS TEXT
);

CREATE OR REPLACE FUNCTION FN_LOG_EVENT()
  RETURNS EVENT_TRIGGER
  LANGUAGE SQL
  AS
  $main$
    INSERT INTO TAB_EVENT_LOGS(DATE_TIME,EVENT_NAME,REMARKS)
      VALUES(NOW(),TG_TAG,'Event Logging');
  $main$;

CREATE EVENT TRIGGER TRG_LOG_EVENT ON DDL_COMMAND_START
  EXECUTE PROCEDURE FN_LOG_EVENT();
```

https://riptutorial.com/zh-CN/postgresql/topic/9255/

10: JavaPostgreSQL

JavaAPIJDBC。

APIJDBC。

JARJAVA。

JDBC。

JDBC URL

JDBC URL

- `jdbc:postgresql://` *`host`* `[:` *`port`* `]/[` *`database`* `][` *`parameters`* `]`

  `host``localhost` *`port`*5432。
  *`host`*IPv6。
  。

  `host` `[:` *`port`* `]`。
  。

- `jdbc:postgresql:` *`database`* `[` *`parameters`* `]`

- `jdbc:postgresql:/[` *`parameters`* `]`

  `localhost` 。

`parameters``key` `[=` *`value`* `]??&`。 *`value`* `true` 。

```
jdbc:postgresql://localhost/test?user=fred&password=secret&ssl&sslfactory=org.postgresql.ssl.NonValidat
```

- JDBC http //download.oracle.com/otndocs/jcp/jdbc-4_2-mrel2-eval-spec/
- PostgreSQL JDBC https //jdbc.postgresql.org/
- PostgreSQL JDBC https //jdbc.postgresql.org/documentation/head/index.html

Examples

java.sql.DriverManager

。

`java.sql.DriverManager` 。

`java.lang.Class.forname(` *`<driver class name>`* `)` 。

```
/**
 * Connect to a PostgreSQL database.
 * @param url the JDBC URL to connect to; must start with "jdbc:postgresql:"
 * @param user the username for the connection
 * @param password the password for the connection
 * @return a connection object for the established connection
 * @throws ClassNotFoundException if the driver class cannot be found on the Java class path
 * @throws java.sql.SQLException if the connection to the database fails
 */
private static java.sql.Connection connect(String url, String user, String password)
    throws ClassNotFoundException, java.sql.SQLException
{
    /*
     * Register the PostgreSQL JDBC driver.
     * This may throw a ClassNotFoundException.
     */
    Class.forName("org.postgresql.Driver");
    /*
     * Tell the driver manager to connect to the database specified with the URL.
     * This may throw an SQLException.
     */
    return java.sql.DriverManager.getConnection(url, user, password);
}
```

JDBC URL`getConnection`。

java.sql.DriverManagerProperties

`java.util.Properties`URL

```
/**
 * Connect to a PostgreSQL database.
 * @param url the JDBC URL to connect to. Must start with "jdbc:postgresql:"
 * @param user the username for the connection
 * @param password the password for the connection
 * @return a connection object for the established connection
 * @throws ClassNotFoundException if the driver class cannot be found on the Java class path
 * @throws java.sql.SQLException if the connection to the database fails
 */
private static java.sql.Connection connect(String url, String user, String password)
    throws ClassNotFoundException, java.sql.SQLException
{
    /*
     * Register the PostgreSQL JDBC driver.
     * This may throw a ClassNotFoundException.
     */
    Class.forName("org.postgresql.Driver");
    java.util.Properties props = new java.util.Properties();
    props.setProperty("user", user);
    props.setProperty("password", password);
    /* don't use server prepared statements */
    props.setProperty("prepareThreshold", "0");
    /*
     * Tell the driver manager to connect to the database specified with the URL.
     * This may throw an SQLException.
     */
    return java.sql.DriverManager.getConnection(url, props);
}
```

javax.sql.DataSource

JNDI`javax.sql.DataSource`。

```
/**
 * Create a data source with connection pool for PostgreSQL connections
 * @param url the JDBC URL to connect to. Must start with "jdbc:postgresql:"
 * @param user the username for the connection
 * @param password the password for the connection
 * @return a data source with the correct properties set
 */
private static javax.sql.DataSource createDataSource(String url, String user, String password)
{
    /* use a data source with connection pooling */
    org.postgresql.ds.PGPoolingDataSource ds = new org.postgresql.ds.PGPoolingDataSource();
    ds.setUrl(url);
    ds.setUser(user);
    ds.setPassword(password);
    /* the connection pool will have 10 to 20 connections */
    ds.setInitialConnections(10);
    ds.setMaxConnections(20);
    /* use SSL connections without checking server certificate */
    ds.setSslMode("require");
    ds.setSslfactory("org.postgresql.ssl.NonValidatingFactory");

    return ds;
}
```

```
/* get a connection from the connection pool */
java.sql.Connection conn = ds.getConnection();

/* do some work */

/* hand the connection back to the pool - it will not be closed */
conn.close();
```

JavaPostgreSQL https://riptutorial.com/zh-CN/postgresql/topic/9633/javapostgresql

11:

Examples

Npgsql.NETPostgresql

Postgresql.NETNpgsql ADO.NET.NET。

。 C

```
var connString = "Host=myserv;Username=myuser;Password=mypass;Database=mydb";
using (var conn = new NpgsqlConnection(connString))
{
    var querystring = "INSERT INTO data (some_field) VALUES (@content)";

    conn.Open();
    // Create a new command with CommandText and Connection constructor
    using (var cmd = new NpgsqlCommand(querystring, conn))
    {
        // Add a parameter and set its type with the NpgsqlDbType enum
        var contentString = "Hello World!";
        cmd.Parameters.Add("@content", NpgsqlDbType.Text).Value = contentString;

        // Execute a query that returns no results
        cmd.ExecuteNonQuery();


        /* It is possible to reuse a command object and open connection instead of creating
new ones */

        // Create a new query and set its parameters
        int keyId = 101;
        cmd.CommandText = "SELECT primary_key, some_field FROM data WHERE primary_key =
@keyId";
        cmd.Parameters.Clear();
        cmd.Parameters.Add("@keyId", NpgsqlDbType.Integer).Value = keyId;

        // Execute the command and read through the rows one by one
        using (NpgsqlDataReader reader = cmd.ExecuteReader())
        {
            while (reader.Read())   // Returns false for 0 rows, or after reading the last row
of the results
            {
                // read an integer value
                int primaryKey = reader.GetInt32(0);
                // or
                primaryKey = Convert.ToInt32(reader["primary_key"]);

                // read a text value
                string someFieldText = reader["some_field"].ToString();
            }
        }
    }
}    // the C# 'using' directive calls conn.Close() and conn.Dispose() for us
```

C-APIPostgreSQL

C-APIPostgreSQL。

`pg_config --includedir`PostgreSQL。
PostgreSQLUNIX`libpq.so` Windows`libpq.dll` 。 PostgreSQL`pg_config --libdir`。

`libpq.so` `libpg.so` 。

`coltype.c`

```
gcc -Wall -I "$(pg_config --includedir)" -L "$(pg_config --libdir)" -o coltype coltype.c -lpq
```

GNU C`-Wl,-rpath,"$(pg_config --libdir)"`

```
cl /MT /W4 /I <include directory> coltype.c <path to libpq.lib>
```

WindowsMicrosoft Visual C.

```
/* necessary for all PostgreSQL client programs, should be first */
#include <libpq-fe.h>

#include <stdio.h>
#include <string.h>

#ifdef TRACE
#define TRACEFILE "trace.out"
#endif

int main(int argc, char **argv) {
#ifdef TRACE
    FILE *trc;
#endif
    PGconn *conn;
    PGresult *res;
    int rowcount, colcount, i, j, firstcol;
    /* parameter type should be guessed by PostgreSQL */
    const Oid paramTypes[1] = { 0 };
    /* parameter value */
    const char * const paramValues[1] = { "pg_database" };

    /*
     * Using an empty connectstring will use default values for everything.
     * If set, the environment variables PGHOST, PGDATABASE, PGPORT and
     * PGUSER will be used.
     */
    conn = PQconnectdb("");

    /*
     * This can only happen if there is not enough memory
     * to allocate the PGconn structure.
     */
    if (conn == NULL)
    {
        fprintf(stderr, "Out of memory connecting to PostgreSQL.\n");
        return 1;
    }

    /* check if the connection attempt worked */
    if (PQstatus(conn) != CONNECTION_OK)
    {
        fprintf(stderr, "%s\n", PQerrorMessage(conn));
        /*
         * Even if the connection failed, the PGconn structure has been
         * allocated and must be freed.
         */
        PQfinish(conn);
        return 1;
    }

#ifdef TRACE
    if (NULL == (trc = fopen(TRACEFILE, "w")))
    {
        fprintf(stderr, "Error opening trace file \"%s\"!\n", TRACEFILE);
        PQfinish(conn);
        return 1;
    }

    /* tracing for client-server communication */
    PQtrace(conn, trc);
#endif

    /* this program expects the database to return data in UTF-8 */
    PQsetClientEncoding(conn, "UTF8");

    /* perform a query with parameters */
    res = PQexecParams(
        conn,
        "SELECT column_name, data_type "
            "FROM information_schema.columns "
            "WHERE table_name = $1",
        1,            /* one parameter */
        paramTypes,
        paramValues,
        NULL,        /* parameter lengths are not required for strings */
        NULL,        /* all parameters are in text format */
        0            /* result shall be in text format */
    );

    /* out of memory or sever communication broken */
    if (NULL == res)
    {
        fprintf(stderr, "%s\n", PQerrorMessage(conn));
        PQfinish(conn);
#ifdef TRACE
        fclose(trc);
#endif
        return 1;
    }

    /* SQL statement should return results */
    if (PGRES_TUPLES_OK != PQresultStatus(res))
    {
        fprintf(stderr, "%s\n", PQerrorMessage(conn));
        PQfinish(conn);
#ifdef TRACE
        fclose(trc);
#endif
        return 1;
    }

    /* get count of result rows and columns */
    rowcount = PQntuples(res);
    colcount = PQnfields(res);

    /* print column headings */
    firstcol = 1;

    printf("Description of the table \"pg_database\"\n");

    for (j=0; j<colcount; ++j)
    {
        if (firstcol)
            firstcol = 0;
        else
            printf(": ");

        printf(PQfname(res, j));
    }

    printf("\n\n");

    /* loop through rosult rows */
    for (i=0; i<rowcount; ++i)
    {
        /* print all column data */
        firstcol = 1;

        for (j=0; j<colcount; ++j)
        {
            if (firstcol)
                firstcol = 0;
            else
                printf(": ");

            printf(PQgetvalue(res, i, j));
        }

        printf("\n");
    }

    /* this must be done after every statement to avoid memory leaks */
    PQclear(res);
    /* close the database connection and release memory */
    PQfinish(conn);
#ifdef TRACE
    fclose(trc);
#endif
    return 0;
}
```

psycopg2pythonPostgreSQL

。

```
import psycopg2

db_host = 'postgres.server.com'
db_port = '5432'
db_un = 'user'
db_pw = 'password'
db_name = 'testdb'

conn = psycopg2.connect("dbname={} host={} user={} password={}".format(
                         db_name,  db_host, db_un, db_pw),
                         cursor_factory=RealDictCursor)
cur = conn.cursor()
sql = 'select * from testtable where id > %s and id < %s'
args = (1, 4)
cur.execute(sql, args)

print(cur.fetchall())
```

```
[{'id': 2, 'fruit': 'apple'}, {'id': 3, 'fruit': 'orange'}]
```

Pomm2PHPPostgreSQL

pomm 。 /。

Pommcomposer

```
<?php
use PommProject\Foundation\Pomm;
$loader = require __DIR__ . '/vendor/autoload.php';
$pomm = new Pomm(['my_db' => ['dsn' => 'pgsql://user:pass@host:5432/db_name']]);

// TABLE comment (
// comment_id uuid PK, created_at timestamptz NN,
// is_moderated bool NN default false,
// content text NN CHECK (content !~ '^\s+$'), author_email text NN)
$sql = <<<SQL
SELECT
  comment_id,
  created_at,
  is_moderated,
  content,
  author_email
FROM comment
  INNER JOIN author USING (author_email)
WHERE
  age(now(), created_at) < $*::interval
ORDER BY created_at ASC
SQL;

// the argument will be converted as it is cast in the query above
$comments = $pomm['my_db']
    ->getQueryManager()
    ->query($sql, [DateInterval::createFromDateString('1 day')]);

if ($comments->isEmpty()) {
    printf("There are no new comments since yesterday.");
} else {
    foreach ($comments as $comment) {
        printf(
                "%s has posted at %s. %s\n",
                $comment['author_email'],
                $comment['created_at']->format("Y-m-d H:i:s"),
                $comment['is_moderated'] ? '[OK]' : '');
    }
}
```

PommSQL。 PHPPostgres。 。 \ DateTime。

https://riptutorial.com/zh-CN/postgresql/topic/2014/

12: WITH

Examples

SELECT

。

```
WITH sales AS (
  SELECT
    orders.ordered_at,
    orders.user_id,
    SUM(orders.amount) AS total
  FROM orders
  GROUP BY orders.ordered_at, orders.user_id
)
SELECT
  sales.ordered_at,
  sales.total,
  users.name
FROM sales
JOIN users USING (user_id)
```

WITH RECURSIVE

```
create table empl (
    name text primary key,
    boss text null
        references name
            on update cascade
            on delete cascade
        default null
);

insert into empl values ('Paul',null);
insert into empl values ('Luke','Paul');
insert into empl values ('Kate','Paul');
insert into empl values ('Marge','Kate');
insert into empl values ('Edith','Kate');
insert into empl values ('Pam','Kate');
insert into empl values ('Carol','Luke');
insert into empl values ('John','Luke');
insert into empl values ('Jack','Carol');
insert into empl values ('Alex','Carol');

with recursive t(level,path,boss,name) as (
        select 0,name,boss,name from empl where boss is null
    union
        select
            level + 1,
            path || ' > ' || empl.name,
            empl.boss,
            empl.name
        from
```

```
             empl join t
                 on empl.boss = t.name
) select * from t order by path;
```

WITH https://riptutorial.com/zh-CN/postgresql/topic/1973/-with-

13:

Coalescenone null。 null。 nullnull。

Examples

null

```
PGSQL> SELECT COALESCE(NULL, NULL, 'HELLO WORLD');

coalesce
--------
'HELLO WORLD'
```

null

PGSQL> SELECT COALESCENULLNULL'first non null'nullnull'second non null';

```
coalesce
--------
'first non null'
```

```
PGSQL> SELECT COALESCE(NULL, NULL, NULL);

coalesce
--------
```

https://riptutorial.com/zh-CN/postgresql/topic/10576/

14:

pg_dumpallpg_dump

pg_start_backup()pg_stop_backup()。 ;ZFSFreeBSD。

PostgresPostgres。 。

Examples

```
pg_dump -Fc -f DATABASE.pgsql DATABASE
```

-Fc“”SQL;pg_restore 。 vanilla SQL

```
pg_dump -f DATABASE.sql DATABASE
```

```
pg_dump DATABASE > DATABASE.sql
```

```
psql < backup.sql
```

-1。 -fshell。

```
psql -1f backup.sql
```

-dpg_restore

```
pg_restore -d DATABASE DATABASE.pgsql
```

SQL

```
pg_restore backup.pgsql > backup.sql
```

。

postgresqlpg_dumppg_restore。

```
$ pg_dumpall -f backup.sql
```

pg_dump。

cron

```
$ postgres-backup-$(date +%Y-%m-%d).sql
```

◦ Postgresql - WAL

pg_dumpallPostgres`$PGDATA` `pg_hba.conf` `postgresql.conf` ◦

```
postgres=# SELECT pg_start_backup('my-backup');
postgres=# SELECT pg_stop_backup();
```

Postgres◦

CSV

```
COPY <tablename> FROM '<filename with path>';
```

`/home/user/user_data.csv` `user`

```
COPY user FROM '/home/user/user_data.csv';
```

```
COPY user FROM '/home/user/user_data' WITH DELIMITER '|';
```

`with delimiter,`

```
COPY user FROM '/home/user/user_data' WITH DELIMITER '|' HEADER;
```

◦ `QUOTE`;CSV◦

o / p

COPY <tablename> TO STDOUTDELIMITER'|';

COPYTO STDOUTDELIMITER'|';

COPYFROM'/ home / user / user_data'with DELIMITER'|';

SQL

COPYsqlTO'<>';

COPYSELECT * FROM user WHERE user_name LIKE'A'TO'/ home / user / user_data';

COPYTO PROGRAM'gzip> /home/user/user_data.gz';

gzip。

psql

psql。

csvcsv

```
psql -p \<port> -U \<username> -d \<database> -A -F<delimiter> -c\<sql to execute> \> \<output
filename with path>

psql -p 5432 -U postgres -d test_database -A -F, -c "select * from user" >
/home/user/user_data.csv
```

-A-F。

`-F`

```
-A or --no-align
```

。 。

https://riptutorial.com/zh-CN/postgresql/topic/2291/

15: PostgreSQLCSV

Adminermysqlcsvpostgresql。 postgresqlCSV。

Examples

PostgreSQLcsv

```
COPY products(is_public, title, discount) TO 'D:\csv_backup\products_db.csv' DELIMITER ',' CSV
HEADER;

COPY categories(name) TO 'D:\csv_backup\categories_db.csv' DELIMITER ',' CSV HEADER;
```

csv

```
COPY products TO 'D:\csv_backup\products_db.csv' DELIMITER ',' CSV HEADER;

COPY categories TO 'D:\csv_backup\categories_db.csv' DELIMITER ',' CSV HEADER;
```

```
copy (select oid,relname from pg_class limit 5) to stdout;
```

PostgreSQLCSV https://riptutorial.com/zh-CN/postgresql/topic/8643/postgresqlcsv

16:

Examples

INSERT

person

```
CREATE TABLE person (
    person_id BIGINT,
    name VARCHAR(255).
    age INT,
    city VARCHAR(255)
);
```

```
INSERT INTO person VALUES (1, 'john doe', 25, 'new york');
```

```
INSERT INTO person (name, age) VALUES ('john doe', 25);
```

NOT NULL。

```
INSERT INTO person (name, age) VALUES
  ('john doe', 25),
  ('jane doe', 20);
```

select

select

```
INSERT INTO person SELECT * FROM tmp_person WHERE age < 30;
```

select。 `tmp_personperson` 。

COPY

COPYPostgreSQL。 `INSERT`。

。

```
cat > samplet_data.csv

1,Yogesh
2,Raunak
3,Varun
4,Kamal
5,Hari
6,Amit
```

。

```
CREATE TABLE copy_test(id int, name varchar(8));
```

。

```
COPY copy_test FROM '/path/to/file/sample_data.csv' DELIMITER ',';
```

`stdin`

```
COPY copy_test FROM stdin DELIMITER ',';
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself.
>> 7,Amol
>> 8,Amar
>> \.
Time: 85254.306 ms

SELECT * FROM copy_test ;
 id |  name
----+--------
  1 | Yogesh
  3 | Varun
  5 | Hari
  7 | Amol
  2 | Raunak
  4 | Kamal
  6 | Amit
  8 | Amar
```

```
COPY copy_test TO 'path/to/file/sample_data.csv'  DELIMITER ',';
```

COPY

INSERTRETURING

。

`my_table`

```
CREATE TABLE my_table
(
id serial NOT NULL, -- serial data type is auto incrementing four-byte integer
name character varying,
contact_number integer,
CONSTRAINT my_table_pkey PRIMARY KEY (id)
);
```

`my_table`id

```
INSERT INTO my_table(name, contact_number) VALUES ( 'USER', 8542621) RETURNING id;
```

id。

◦

◦

```
postgres=# select * from my_table;
 c1 | c2 | c3
----+----+----
  1 |  1 |  1
  2 |  2 |  2
  3 |  3 |  3
  4 |  4 |  4
  5 |  5 |
(5 rows)

postgres=# copy my_table to '/home/postgres/my_table.txt' using delimiters '|' with null as
'null_string' csv header;
COPY 5
postgres=# \! cat my_table.txt
c1|c2|c3
1|1|1
2|2|2
3|3|3
4|4|4
5|5|null_string
```

UPSERT - INSERT ....

9.5 postgres`INSERTUPSERT`◦

my_table◦ PK

```
b=# INSERT INTO my_table (name,contact_number) values ('one',333) RETURNING id;
 id
----
  2
(1 row)

INSERT 0 1
```

```
b=# INSERT INTO my_table values (2,'one',333);
ERROR:  duplicate key value violates unique constraint "my_table_pkey"
DETAIL:  Key (id)=(2) already exists.
```

Upsert

```
b=# INSERT INTO my_table values (2,'one',333) ON CONFLICT (id) DO UPDATE SET name =
my_table.name||' changed to: "two" at '||now() returning *;
 id |                        name                              | contact_number
----+-----------------------------------------------------------------------------------------------
------------------+----------------
  2 | one changed to: "two" at 2016-11-23 08:32:17.105179+00 |            333
(1 row)

INSERT 0 1
```

https://riptutorial.com/zh-CN/postgresql/topic/2561/

17:

PostgreSQL。 CREATE TYPEPostgreSQL。

https://www.postgresql.org/docs/9.6/static/datatype.html

Examples

`smallint`	2		-32768+32767
`integer`	4		-2147483648+2147483647
`bigint`	8		-9223372036854775808+9223372036854775807
`decimal`			131072;16383
`numeric`			131072;16383
`real`	4		6
`double precision`	8		15
`smallserial`	2		132767
`serial`	4		12147483647
`bigserial`	8		19223372036854775807
`int4range`			
`int8range`		bigint	
`numrange`			

/

`timestamp`	8		4713	294276	1/ 14
`timestamp`	8		4713	294276	1/ 14
`date`	4		4713	5874897	1
`time`	8		00:00:00	24:00:00	1/ 14
`time`	12		000000 + 1459	240000-1459	1/ 14

`interval`	16		-178000000	1.78	1/ 14
`tsrange`					
`tstzrange`					
`daterange`					

`point`	16		XY
`line`	32		{ABC}
`lseg`	32		X1Y1X2Y2
`box`	32		X1Y1X2Y2
`path`	16 + 16n		X1Y1...
`path`	16 + 16n		[X1Y1...]
`polygon`	40 + 16n		X1Y1...
`circle`	24		<xyr>

`cidr`	719	IPv4IPv6
`inet`	719	IPv4IPv6
`macaddr`	6	MAC

`character varying(n)` `varchar(n)`	
`character(n)` `char(n)`	
`text`	

PostgreSQL。 Array。

```
SELECT integer[];
SELECT integer[3];
SELECT integer[][];
SELECT integer[3][3];
SELECT integer ARRAY;
SELECT integer ARRAY[3];
SELECT '{0,1,2}';
SELECT '{{0,1},{1,2}}';
SELECT ARRAY[0,1,2];
SELECT ARRAY[ARRAY[0,1],ARRAY[1,2]];
```

PostgreSQLn`array[1]array[n]`。

```
--accesing a spefific element
WITH arr AS (SELECT ARRAY[0,1,2] int_arr) SELECT int_arr[1] FROM arr;

int_arr
---------
        0
(1 row)

--sclicing an array
WITH arr AS (SELECT ARRAY[0,1,2] int_arr) SELECT int_arr[1:2] FROM arr;

int_arr
---------
    {0,1}
(1 row)
```

```
--array dimensions (as text)
with arr as (select ARRAY[0,1,2] int_arr) select array_dims(int_arr) from arr;

array_dims
------------
       [1:3]
(1 row)

--length of an array dimension
 WITH arr AS (SELECT ARRAY[0,1,2] int_arr) SELECT array_length(int_arr,1) FROM arr;

 array_length
--------------
             3
 (1 row)

--total number of elements across all dimensions
 WITH arr AS (SELECT ARRAY[0,1,2] int_arr) SELECT cardinality(int_arr) FROM arr;

 cardinality
-------------
            3
 (1 row)
```

https://riptutorial.com/zh-CN/postgresql/topic/8976/

18:

Examples

to_char()timestampinterval

```
SELECT to_char('2016-08-12 16:40:32'::timestamp, 'DD Mon YYYY HH:MI:SSPM');
```

“201681204:40:32 PM”。 ;。

```
SELECT to_char('2016-08-12 16:40:32'::timestamp,
               '"Today is "FMDay", the "DDth" day of the month of "FMMonth" of "YYYY');
```

“2016812”。 - “I”“D”“W”。 。

TM。 PostgreSQL。

```
SELECT to_char('2016-08-12 16:40:32'::timestamp, 'TMDay, DD" de "TMMonth" del año "YYYY');
```

“Sábado12 de Agostodelaño2016”。

。

```
SELECT (date_trunc('MONTH', ('201608'||'01')::date) + INTERVAL '1 MONTH - 1 day')::DATE;
```

201608。

SELECT date_trunc'week'<>AS“Week”count*FROM <> GROUP BY 1 ORDER BY 1;

https://riptutorial.com/zh-CN/postgresql/topic/4227/-

19: /

“”“”char_lengthcharacter_length。

Examples

1 `SELECT char_length('ABCDE')`

2 `SELECT character_length('ABCDE')`

/ https://riptutorial.com/zh-CN/postgresql/topic/9695/-

20:

- +
- prodDir22-11-2016-19h55
- **+**
-
- dbprod22-11-2016-19h55.backup
- dbprod22-11-2016-19h55.sql **sql**
- **19-1155**
- /save_bd/prodDir22-11-2016-19h55/dbprod22-11-2016-19h55.backup
- /save_bd/prodDir22-11-2016-19h55/dbprod22-11-2016-19h55.sql

SAVE_DB	
dbProd	
dbprod	
/opt/postgres/9.0/bin/pg_dump	pg_dump
-H	localhost
-p	TCPUnix5432
-U	。

1. HDPSSymantec Backup。

。

3。

```
rm -R / save_db / *
```

2. cron 。

cron。

```
crontab -e
```

11。

```
0 23 * * * /saveProdDb.sh
```

Examples

saveProdDb.sh

pgAdmin。 linuxsh

- **SQL** PostgreSQL。
- 。

```
#!/bin/sh
cd /save_db
#rm -R /save_db/*
DATE=$(date +%d-%m-%Y-%Hh%M)
echo -e "Sauvegarde de la base du ${DATE}"
mkdir prodDir${DATE}
cd prodDir${DATE}

#dump file
/opt/postgres/9.0/bin/pg_dump -i -h localhost -p 5432 -U postgres -F c -b -w -v -f
"dbprod${DATE}.backup" dbprod

#SQL file
/opt/postgres/9.0/bin/pg_dump -i -h localhost -p 5432 -U postgres --format plain --verbose  -f
"dbprod${DATE}.sql" dbprod
```

https://riptutorial.com/zh-CN/postgresql/topic/7974/

21: PL / pgSQL

PL / pgSQLPostgreSQL。 SQL。 。

PostgreSQLPL / PythonPL / PerlPLV8PL / pgSQLPostgreSQLSQL。 PL / SQLOraclePL / SQL OraclePL / pgSQLPL / SQL。

PL / pgSQLPL / pgSQLPostgreSQL。 PL / pgSQLPL。

PL / pgSQL

- https //www.postgresql.org/docs/current/static/plpgsql.html
- w3resource.com http //www.w3resource.com/PostgreSQL/pl-pgsql-tutorial.php
- postgres.cz http //postgres.cz/wiki/PL/pgSQL_en
- PostgreSQL2 https //www.packtpub.com/big-data-and-business-intelligence/postgresql-server-programming-second-edition
- PostgreSQL https //www.packtpub.com/big-data-and-business-intelligence/postgresql-developers-guide

Examples

PL / pgSQL

PL / pgSQL

```
CREATE FUNCTION active_subscribers() RETURNS bigint AS $$
DECLARE
    -- variable for the following BEGIN ... END block
    subscribers integer;
BEGIN
    -- SELECT must always be used with INTO
    SELECT COUNT(user_id) INTO subscribers FROM users WHERE subscribed;
    -- function result
    RETURN subscribers;
EXCEPTION
    -- return NULL if table "users" does not exist
    WHEN undefined_table
    THEN RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

SQL。

```
select active_subscribers();
```

PL / pgSQL

```
CREATE [OR REPLACE] FUNCTION functionName (someParameter 'parameterType')
```

```
RETURNS 'DATATYPE'
AS $_block_name_$
DECLARE
    --declare something
BEGIN
    --do something
    --return something
END;
$_block_name_$
LANGUAGE plpgsql;
```

PL / pgSQL

- `Datatype Datatype`
- `Table(column_name column_type, ...)`
- `Setof 'Datatype' or 'table_column'`

'P2222'

```
create or replace function s164() returns void as
$$
begin
raise exception using message = 'S 164', detail = 'D 164', hint = 'H 164', errcode = 'P2222';
end;
$$ language plpgsql
;
```

errm

```
create or replace function s165() returns void as
$$
begin
raise exception '%','nothing specified';
end;
$$ language plpgsql
;
```

```
t=# do
$$
declare
 _t text;
begin
  perform s165();
  exception when SQLSTATE 'P0001' then raise info '%','state P0001 caught: '||SQLERRM;
  perform s164();

end;
$$
;
INFO:  state P0001 caught: nothing specified
ERROR:  S 164
DETAIL:  D 164
HINT:  H 164
CONTEXT:  SQL statement "SELECT s164()"
PL/pgSQL function inline_code_block line 7 at PERFORM
```

P0001P2222。

http //stackoverflow.com/a/2700312/5315974

PL / pgSQL https://riptutorial.com/zh-CN/postgresql/topic/5299/pl---pgsql

22:

Examples

```
create table wf_example(i int, t text,ts timestamptz,b boolean);
insert into wf_example select 1,'a','1970.01.01',true;
insert into wf_example select 1,'a','1970.01.01',false;
insert into wf_example select 1,'b','1970.01.01',false;
insert into wf_example select 2,'b','1970.01.01',false;
insert into wf_example select 3,'b','1970.01.01',false;
insert into wf_example select 4,'b','1970.02.01',false;
insert into wf_example select 5,'b','1970.03.01',false;
insert into wf_example select 2,'c','1970.03.01',true;
```

```
select *
  , dense_rank() over (order by i) dist_by_i
  , lag(t) over () prev_t
  , nth_value(i, 6) over () nth
  , count(true) over (partition by i) num_by_i
  , count(true) over () num_all
  , ntile(3) over() ntile
from wf_example
;
```

```
 i | t |           ts            | b | dist_by_i | prev_t | nth | num_by_i | num_all | ntile
---+---+------------------------+---+-----------+--------+-----+----------+---------+-------
 1 | a | 1970-01-01 00:00:00+01 | f |         1 |        |   3 |        3 |       8 |     1
 1 | a | 1970-01-01 00:00:00+01 | t |         1 | a      |   3 |        3 |       8 |     1
 1 | b | 1970-01-01 00:00:00+01 | f |         1 | a      |   3 |        3 |       8 |     1
 2 | c | 1970-03-01 00:00:00+01 | t |         2 | b      |   3 |        2 |       8 |     2
 2 | b | 1970-01-01 00:00:00+01 | f |         2 | c      |   3 |        2 |       8 |     2
 3 | b | 1970-01-01 00:00:00+01 | f |         3 | b      |   3 |        1 |       8 |     2
 4 | b | 1970-02-01 00:00:00+01 | f |         4 | b      |   3 |        1 |       8 |     3
 5 | b | 1970-03-01 00:00:00+01 | f |         5 | b      |   3 |        1 |       8 |     3
(8 rows)
```

dist_by_i `dense_rank() over (order by i)` row_number。 i `count(DISTINCT i)` wold。 。

prev_t `lag(t) over ()` t。 。

nth `nth_value(i, 6) over ()` i

num_by_i `count(true) over (partition by i)` i

num_all `count(true) over ()`

ntile `ntile(3) over()` 3

vs dense_rank vs rank vs row_number

。

wf_example

```
select i
  , dense_rank() over (order by i)
  , row_number() over ()
  , rank() over (order by i)
from wf_example
```

```
 i | dense_rank | row_number | rank
---+------------+------------+------
 1 |          1 |          1 |    1
 1 |          1 |          2 |    1
 1 |          1 |          3 |    1
 2 |          2 |          4 |    4
 2 |          2 |          5 |    4
 3 |          3 |          6 |    6
 4 |          4 |          7 |    7
 5 |          5 |          8 |    8
```

- *dense_rank*i**VALUES** 。 `i=1`dense_ranknexti`dense_rank`*1* - FIRST。 `i=2` `dense_rank`2。 6`i=3` 3.*i* 。 `dense_rank`*i*。
- *row_number***ROWS** 。
- *rank*`dense_rank`**iROW NUMBER** 。 44`i=2` 。 6`i=3`。

https://riptutorial.com/zh-CN/postgresql/topic/7421/

23:

Examples

minmaxavg

。

individuals

	17
	14
	20

```
SELECT min(age), max(age), avg(age)
FROM individuals;
```

14	20	17

string_agg

string_agg()。

individuals

	15	
	14	
	20	

SELECT ... GROUP BY/

```
SELECT string_agg(name, ', ') AS names, country
FROM individuals
GROUP BY country;
```

GROUP BYstring_agg()。

PostgreSQL

regr_slopeYXXY

regr_slopeYX。 Java。 postgres。

。

```
     ^
     |
s    |  Legend:
i    |  *  - data point
z    |  -- - trend
e    |
(    |
b    |               *
y    |                   --
t    |                --
e    |           * --    *
s    |         --
)    |     *--       *
     |   --    *
     |  -- *
    --------------------------------------->
                      time
```

```
CREATE TABLE heap_histogram (
    -- when the heap histogram was taken
    histwhen timestamp without time zone NOT NULL,
    -- the object type bytes are referring to
    -- ex: java.util.String
    class character varying NOT NULL,
    -- the size in bytes used by the above class
    bytes integer NOT NULL
);
```

。 HAVING> 0。 。

```
-- epoch returns seconds
SELECT class, REGR_SLOPE(bytes,extract(epoch from histwhen)) as slope
    FROM public.heap_histogram
    GROUP BY class
    HAVING REGR_SLOPE(bytes,extract(epoch from histwhen)) > 0
    ORDER BY slope DESC ;
```

```
         class             |        slope
---------------------------+----------------------
 java.util.ArrayList       |     71.7993806279174
 java.util.HashMap         |     49.0324576155785
 java.lang.String          |     31.7770770326123
 joe.schmoe.BusinessObject |     23.2036817108056
 java.lang.ThreadLocal     |     20.9013528767851
```

java.util.ArrayList71.799。

https://riptutorial.com/zh-CN/postgresql/topic/4803/

24:

Examples

```
CREATE TABLE person (
    person_id BIGINT NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    first_name VARCHAR(255),
    address VARCHAR(255),
    city VARCHAR(255),
    PRIMARY KEY (person_id)
);
```

`PRIMARY KEY`

```
CREATE TABLE person (
    person_id BIGINT NOT NULL PRIMARY KEY,
    last_name VARCHAR(255) NOT NULL,
    first_name VARCHAR(255),
    address VARCHAR(255),
    city VARCHAR(255)
);
```

。 `Person` `"Person"` PostgreSQL。

`psql`。

```
\d tablename
```

```
\d+ tablename
```

psql\ dd。

select

person

```
CREATE TABLE person (
    person_id BIGINT NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    first_name VARCHAR(255),
    age INT NOT NULL,
    PRIMARY KEY (person_id)
);
```

30

```
CREATE TABLE people_over_30 AS SELECT * FROM person WHERE age > 30;
```

◦ write-ahead◦

```
CREATE UNLOGGED TABLE person (
    person_id BIGINT NOT NULL PRIMARY KEY,
    last_name VARCHAR(255) NOT NULL,
    first_name VARCHAR(255),
    address VARCHAR(255),
    city VARCHAR(255)
);
```

◦

Agency◦

```
CREATE TABLE agencies ( -- first create the agency table
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL
)

CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  agency_id NOT NULL INTEGER REFERENCES agencies(id) DEFERRABLE INITIALLY DEFERRED -- this is
going to references your agency table.
)
```

https://riptutorial.com/zh-CN/postgresql/topic/2430/

25:

- `CREATE ROLE name [ [ WITH ] option [ ... ] ]`
- `CREATE USER name [ [ WITH ] option [ ... ] ]`
- `where option can be: SUPERUSER | NOSUPERUSER | CREATEDB | NOCREATEDB | CREATEROLE | NOCREATEROLE | CREATEUSER | NOCREATEUSER | INHERIT | NOINHERIT | LOGIN | NOLOGIN | CONNECTION LIMIT connlimit | [ENCRYPTED | UNENCRYPTED] PASSWORD 'password' | VALID UNTIL 'timestamp' | IN ROLE role_name [, ...] | IN GROUP role_name [, ...] | ROLE role_name [, ...] | ADMIN role_name [, ...] | USER role_name [, ...] | SYSID uid`

Examples

`postgres` 。 。 `niceusername` `very-strong-password`

```
CREATE ROLE niceusername with PASSWORD 'very-strong-password' LOGIN;
```

`psql` `.psql_history` PostgreSQL。

`\password`。 。

```
CREATE ROLE niceusername with LOGIN;
\password niceusername
```

。

shell

```
$ createuser -P blogger
Enter password for the new role: ********
Enter it again: ********

$ createdb -O blogger blogger
```

`pg_hba.conf`

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    sameuser        all             localhost               md5
local   sameuser        all                                     md5
```

。

1. > admin
2. > read_write
3. > read_only

```
--ACCESS DB
REVOKE CONNECT ON DATABASE nova FROM PUBLIC;
```

```
GRANT  CONNECT ON DATABASE nova  TO user;
```

。

```
--ACCESS SCHEMA
REVOKE ALL     ON SCHEMA public FROM PUBLIC;
GRANT  USAGE   ON SCHEMA public  TO user;
```

`read_write`。

```
--ACCESS TABLES
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM PUBLIC ;
GRANT SELECT                         ON ALL TABLES IN SCHEMA public TO read_only ;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO read_write ;
GRANT ALL                            ON ALL TABLES IN SCHEMA public TO admin ;

--ACCESS SEQUENCES
REVOKE ALL   ON ALL SEQUENCES IN SCHEMA public FROM PUBLIC;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO read_only; -- allows the use of CURRVAL
GRANT UPDATE ON ALL SEQUENCES IN SCHEMA public TO read_write; -- allows the use of NEXTVAL and
SETVAL
GRANT USAGE  ON ALL SEQUENCES IN SCHEMA public TO read_write; -- allows the use of CURRVAL and
NEXTVAL
GRANT ALL    ON ALL SEQUENCES IN SCHEMA public TO admin;
```

search_path

search_path。

1. 。

```
postgres=# \c postgres user1
You are now connected to database "postgres" as user "user1".
postgres=> show search_path;
  search_path
----------------
 "$user",public
(1 row)
```

2. `alter usersearch_pathmy_schema`

```
postgres=> \c postgres postgres
You are now connected to database "postgres" as user "postgres".
postgres=# alter user user1 set search_path='my_schema, "$user", public';
ALTER ROLE
```

3. 。

```
postgres=# \c postgres user1
Password for user user1:
You are now connected to database "postgres" as user "user1".
postgres=> show search_path;
 search_path
```

```
-------------
 my_schema, "$user", public
(1 row)
```

```
postgres=# set role user1;
postgres=# show search_path;
 search_path
-------------
 my_schema, "$user", public
(1 row)
```

。

three users

1. admin > admin
2. > read_write
3. > read_only

。

```
ALTER DEFAULT PRIVILEGES IN SCHEMA myschema GRANT SELECT                      ON TABLES TO
read_only;
ALTER DEFAULT PRIVILEGES IN SCHEMA myschema GRANT SELECT,INSERT,DELETE,UPDATE ON TABLES TO
read_write;
ALTER DEFAULT PRIVILEGES IN SCHEMA myschema GRANT ALL                         ON TABLES TO
admin;
```

。

```
ALTER DEFAULT PRIVILEGES FOR ROLE admin GRANT SELECT  ON TABLES TO read_only;
```

```
CREATE USER readonly WITH ENCRYPTED PASSWORD 'yourpassword';
GRANT CONNECT ON DATABASE <database_name> to readonly;

GRANT USAGE ON SCHEMA public to readonly;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;
```

https://riptutorial.com/zh-CN/postgresql/topic/1572/

26:

function_name。

- https //www.postgresql.org/docs/current/static/sql-createtrigger.html
- https //www.postgresql.org/docs/current/static/plpgsql-trigger.html

Examples

PL / pgSQL

。

```
CREATE OR REPLACE FUNCTION my_simple_trigger_function()
RETURNS trigger AS
$BODY$

BEGIN
    -- TG_TABLE_NAME :name of the table that caused the trigger invocation
IF (TG_TABLE_NAME = 'users') THEN

    --TG_OP : operation the trigger was fired
  IF (TG_OP = 'INSERT') THEN
    --NEW.id is holding the new database row value (in here id is the id column in users
table)
    --NEW will return null for DELETE operations
    INSERT INTO log_table (date_and_time, description) VALUES (now(), 'New user inserted. User
ID: '|| NEW.id);
    RETURN NEW;

  ELSIF (TG_OP = 'DELETE') THEN
    --OLD.id is holding the old database row value (in here id is the id column in users
table)
    --OLD will return null for INSERT operations
    INSERT INTO log_table (date_and_time, description) VALUES (now(), 'User deleted.. User ID:
' || OLD.id);
    RETURN OLD;

  END IF;

RETURN null;
END IF;

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

users

```
CREATE TRIGGER my_trigger
AFTER INSERT OR DELETE
ON users
FOR EACH ROW
```

```
EXECUTE PROCEDURE my_simple_trigger_function();
```

- BEFOREBEFORE - ;
- AFTER-;
- INSTEAD OF。

- FOR EACH ROW;
- FOR EACH STATEMENTonde。

```
CREATE TABLE company (
    id          SERIAL PRIMARY KEY NOT NULL,
    name        TEXT NOT NULL,
    created_at  TIMESTAMP,
    modified_at TIMESTAMP DEFAULT NOW()
)

CREATE TABLE log (
    id          SERIAL PRIMARY KEY NOT NULL,
    table_name  TEXT NOT NULL,
    table_id    TEXT NOT NULL,
    description TEXT NOT NULL,
    created_at  TIMESTAMP DEFAULT NOW()
)
```

1

```
CREATE OR REPLACE FUNCTION add_created_at_function()
  RETURNS trigger AS $BODY$
BEGIN
  NEW.created_at := NOW();
  RETURN NEW;
END $BODY$
LANGUAGE plpgsql;
```

2

```
CREATE TRIGGER add_created_at_trigger
BEFORE INSERT
ON company
FOR EACH ROW
EXECUTE PROCEDURE add_created_at_function();
```

3

```
INSERT INTO company (name) VALUES ('My company');
SELECT * FROM company;
```

1

```
CREATE OR REPLACE FUNCTION add_log_function()
  RETURNS trigger AS $BODY$
DECLARE
  vDescription TEXT;
  vId INT;
  vReturn RECORD;
BEGIN
    vDescription := TG_TABLE_NAME || ' ';
     IF (TG_OP = 'INSERT') THEN
        vId := NEW.id;
        vDescription := vDescription || 'added. Id: ' || vId;
        vReturn := NEW;
    ELSIF (TG_OP = 'UPDATE') THEN
        vId := NEW.id;
        vDescription := vDescription || 'updated. Id: ' || vId;
        vReturn := NEW;
    ELSIF (TG_OP = 'DELETE') THEN
        vId := OLD.id;
        vDescription := vDescription || 'deleted. Id: ' || vId;
        vReturn := OLD;
    END IF;

    RAISE NOTICE 'TRIGER called on % - Log: %', TG_TABLE_NAME, vDescription;

    INSERT INTO log
        (table_name, table_id, description, created_at)
        VALUES
        (TG_TABLE_NAME, vId, vDescription, NOW());

    RETURN vReturn;
END $BODY$
  LANGUAGE plpgsql;
```

2

```
CREATE TRIGGER add_log_trigger
AFTER INSERT OR UPDATE OR DELETE
ON company
FOR EACH ROW
EXECUTE PROCEDURE add_log_function();
```

3

```
INSERT INTO company (name) VALUES ('Company 1');
INSERT INTO company (name) VALUES ('Company 2');
INSERT INTO company (name) VALUES ('Company 3');
UPDATE company SET name='Company new 2' WHERE name='Company 2';
```

```
DELETE FROM company WHERE name='Company 1';
SELECT * FROM log;
```

https://riptutorial.com/zh-CN/postgresql/topic/6957/

27:

Examples

WHERESELECT

```
CREATE TABLE sch_test.user_table
(
  id serial NOT NULL,
  username character varying,
  pass character varying,
  first_name character varying(30),
  last_name character varying(30),
  CONSTRAINT user_table_pkey PRIMARY KEY (id)
)

+----+------------+-----------+----------+------+
| id | first_name | last_name | username | pass |
+----+------------+-----------+----------+------+
| 1  | hello      | world     | hello    | word |
+----+------------+-----------+----------+------+
| 2  | root       | me        | root     | toor |
+----+------------+-----------+----------+------+
```

```
SELECT * FROM schema_name.table_name WHERE <condition>;
```

```
SELECT field1, field2 FROM schema_name.table_name WHERE <condition>;
```

```
-- SELECT every thing where id = 1
SELECT * FROM schema_name.table_name WHERE id = 1;

-- SELECT id where username = ? and pass = ?
SELECT id FROM schema_name.table_name WHERE username = 'root' AND pass = 'toor';

-- SELECT first_name where id not equal 1
SELECT first_name FROM schema_name.table_name WHERE id != 1;
```

https://riptutorial.com/zh-CN/postgresql/topic/9528/

28:

Examples

```
WITH RECURSIVE t(n) AS (
    VALUES (1)
  UNION ALL
    SELECT n+1 FROM t WHERE n < 100
)
SELECT sum(n) FROM t;
```

https://riptutorial.com/zh-CN/postgresql/topic/9025/

29:

PostgreSQL http //stackoverflow.com/a/3075248/653378

Examples

```
CREATE TABLE users (username text, email text);
CREATE TABLE simple_users () INHERITS (users);
CREATE TABLE users_with_password (password text) INHERITS (users);
```


simple_users

users_with_password


```
CREATE TABLE users (username text, email text);
CREATE TABLE simple_users () INHERITS (users);
```

```
ALTER TABLE simple_users ADD COLUMN password text;
```

simple_users


```
ALTER TABLE users ADD COLUMN password text;
```

“simple_users”“password”

```
ALTER TABLE users DROP COLUMN password;
```

simple_users

`simple_users` PostgreSQL。

`password`。

https://riptutorial.com/zh-CN/postgresql/topic/5429/

S. No		Contributors
1	postgresql	a_horse_with_no_name, Alison S, AndrewCichocki, Ben, Ben H, bignose, Community, Dakota Wagner, DeadEye, Demircan Celebi, Dmitri Goldring, e4c5, , jasonszhao, Kirk Roybal, Marek Skiba, Mokadillion, Patrick, user_0
2	EXTENSION dblink postgres_fdw	Riya Bansal, YCF_L
3	JSON	Clodoaldo Neto, commonSenseCode, jgm, KIRAN KUMAR MATAM, mnoronha, Peter Krauss
4	postgresql	Ben, KIRAN KUMAR MATAM
5	PostgreSQL	gpdude_, Patrick
6	Postgres	Ben H, skj123
7	Postgres	Ben H, skj123, user_0, YCF_L
8	UPDATE	frlan, leeor
9		Ben H, Tajinder, Udlei Nati
10	JavaPostgreSQL	Laurenz Albe
11		AstraSerg, brichins, greg, Laurenz Albe
12	WITH	Daniel Lyons, Jakub Fedyczak, Kevin Sylvestre
13		Mokadillion
14		ankidaemon, Ben H, Daniel Lyons, e4c5, Laurel, mnoronha
15	PostgreSQLCSV	Vao Tsun, wOwhOw
16		chalitha geekiyanage, e4c5, gpdude_, KIM, lamorach, leeor, Nathaniel Waisbrot, Patrick, Vao Tsun
17		Ben H, user_0
18		KIM, Nuri Tasdemir, Patrick, Tom Gerken
19	/	Mohamed Navas
20		bilelovitch

21	PL / pgSQL	AndrewCichocki, Ben H, Goerman, Laurenz Albe, Vao Tsun
22		mnoronha, Vao Tsun
23		Alison S, joseph, Kirill Sokolov, Patrick
24		e4c5, Jefferson, KIM, leeor, Patrick
25		Ben, Ben H, bilelovitch, Blackus, Daniel Lyons, e4c5, greg, KIM, Laurenz Albe, mnoronha, Reboot
26		chalitha geekiyanage, mnoronha, Udlei Nati
27		YCF_L
28		Ben H
29		evuez