PostgreSQL 11 Administration CookBook

Chapter-2: Exploring the Database

Upcode Software
Engineer Team

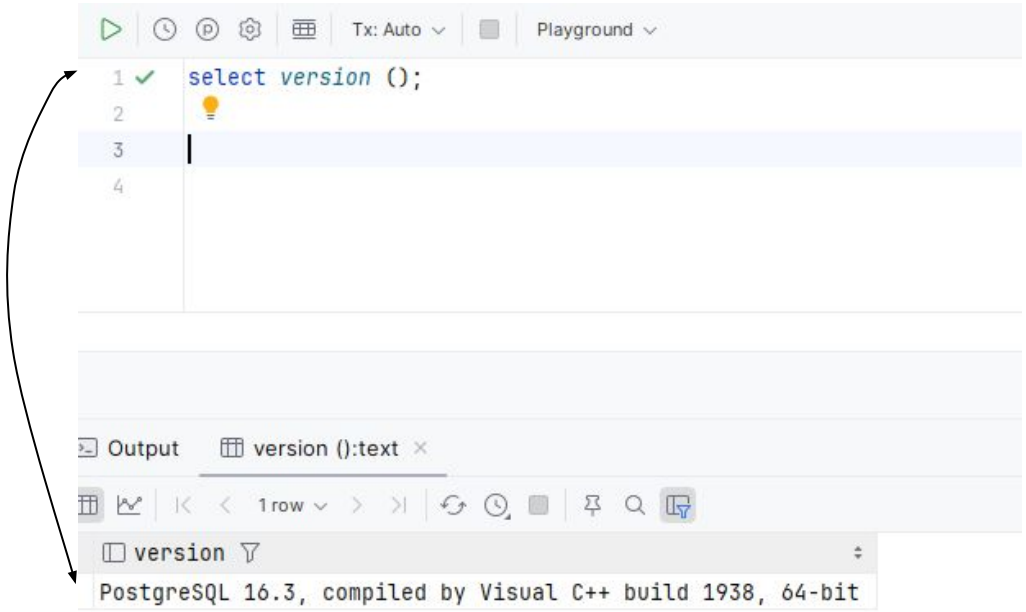
-2024-

CONTENT

1. What version is the server?
2. What is the server uptime?
3. Locating the database server files
4. Locating the database server's message log
5. How much disk space does a database use?
6. How much disk space does a table use?
7. How many rows are there in a table?
8. Quickly estimating the number of rows in a table
9. Reference

What version is the server?

We will find out the version by directly querying the database server:



How it works…

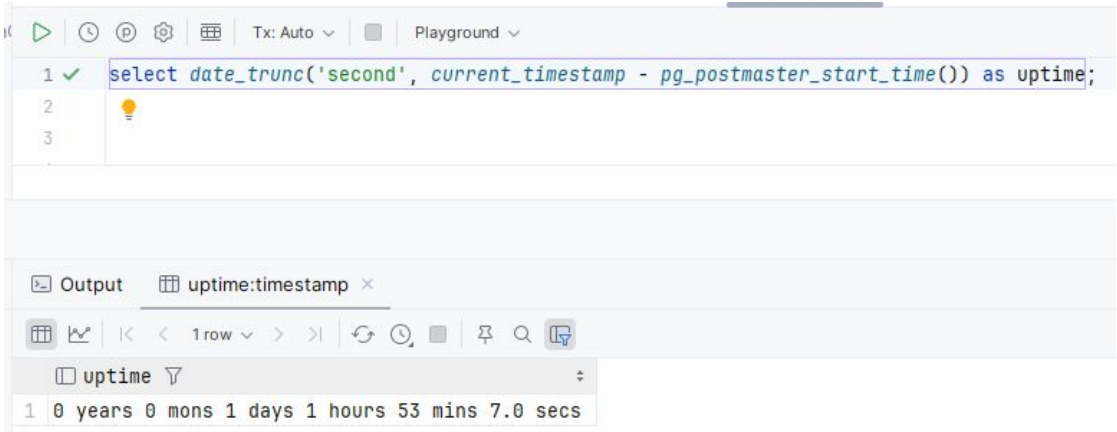
- The current PostgreSQL server version format is composed of two numbers;
- The first number indicates the major release, and the second one denotes subsequent maintenance releases for that major release.
- It is common to mention just the major release when discussing what features are supported, as they are unchanged on a maintenance release.

	version
1	PostgreSQL 16.3, compiled by Visual C++ build 1938, 64-bit

What is the server uptime?(1/2)

You may wonder, how long has it been since the server started?

For instance, you might want to verify that there was no server crash if your server is not monitored; or to see when the server was last restarted, for instance, to change the configuration.
We will find this out by asking the database server.



What is the server uptime?(2/2)

Postgres stores the server start time, so we can access it directly, as follows:

```
4 ✓ select pg_postmaster_start_time();
5
```

Output | pg_postmaster_start_time():timestamp

1 row

pg_postmaster_start_time
2024-06-04 09:00:16.215121 +00:00

Then, we can write a SQL query to get the uptime, like this :

```
7 ✓ select current_timestamp - pg_postmaster_start_time();
8
9
```

Output | current_timestamp - ...tart_time():timestamp

1 row

?column?
0 years 0 mons 1 days 2 hours 5 mins 9.298083 secs

Locating the database server files(1/4)

How to do it...

The following are the system default data directory locations:

- Debian or Ubuntu systems: /var/lib/postgresql/MAJOR_RELEASE/main
- Red Hat RHEL, CentOS, and Fedora: /var/lib/pgsql/data/
- Windows: C:\Program Files\PostgreSQL\MAJOR_RELEASE\data

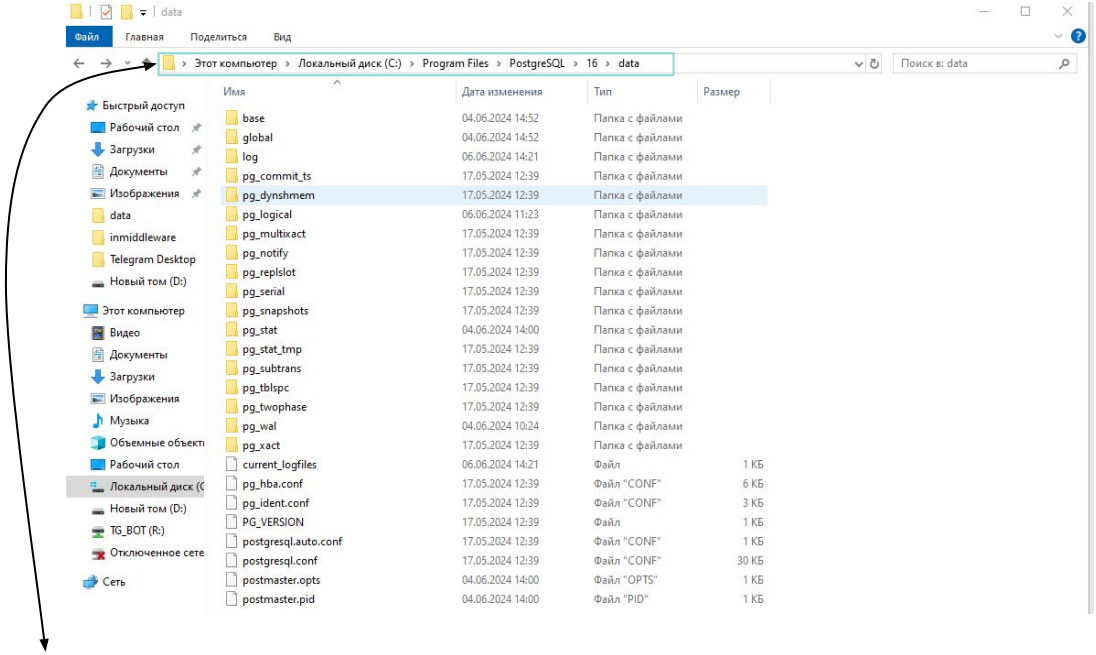
Locating the database server files(2/4)

Subdirectory	Purpose
base	This is the main table storage. Beneath this directory, each database has its own directory, within which are the files for each database table or index.
global	Here are the tables that are shared across all databases, including the list of databases.
pg_commit_ts	Here we store transaction commit timestamp data (from 9.5 onward).
pg_dynshmem	This includes dynamic shared memory information (from 9.4 onward).
pg_logical	This includes logical decoding status data.
pg_multixact	This includes files used for shared row-level locks.
pg_notify	This includes the LISTEN/NOTIFY status files.
pg_replslot	This includes information about replication slots (from 9.4 onward).
pg_serial	This includes information on committed serializable transactions.

Locating the database server files(3/4)

pg_snapshots	This includes exported snapshot files.
pg_stat	This includes permanent statistics data.
pg_stat_tmp	This includes transient statistics data.
pg_subtrans	This includes subtransaction status data.
pg_tblspc	This includes symbolic links to tablespace directories.
pg_twophase	This includes state files for prepared transactions.
pg_wal	This includes the transaction log or **Write-Ahead Log** (**WAL**) (formerly pg_xlog).
pg_xact	This includes the transaction status files (formerly pg_clog).

Locating the database server files(4/4)



PostgreSQL database server.

Locating the database server's message log(1/3)

- The database server's message log is a record of all messages recorded by the database server.
- This is the first place to look if you have server problems, and a good place to check regularly.

This log will include messages that look something like the following:

```
2016-09-01 19:37:41 GMT [2507-1] LOG:  database system was shut down at
2016-09-01 19:37:38 GMT

2016-09-01 19:37:41 GMT [2506-1] LOG:  database system is ready to accept
connections
```

Locating the database server's message log(2/3)

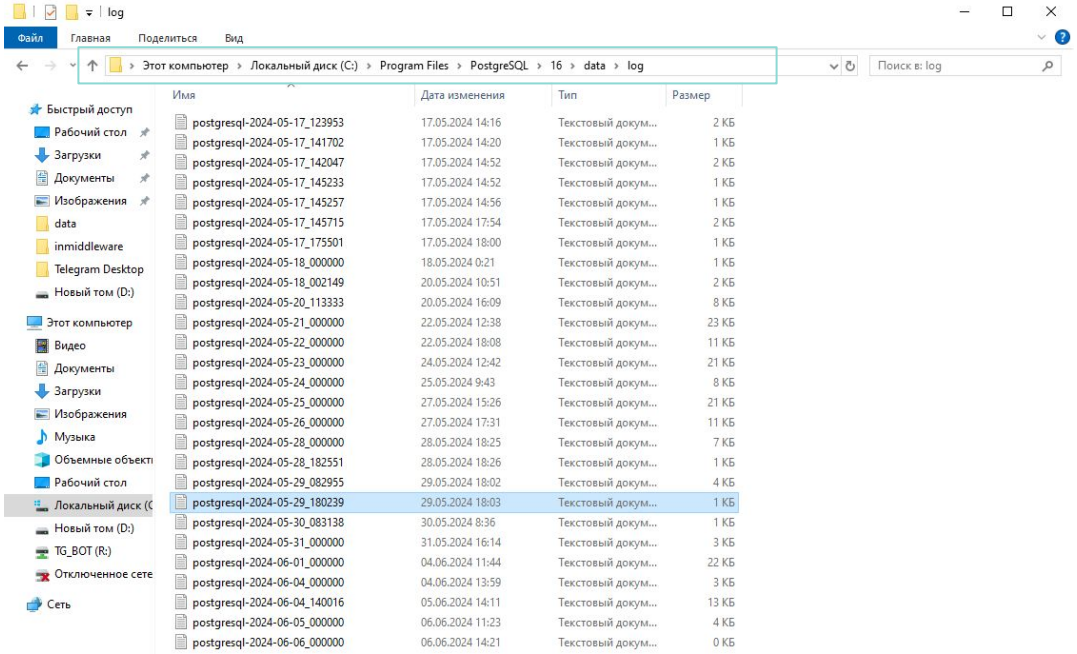
The following are the default server log locations:

- **Debian or Ubuntu systems:** /var/log/postgresql
- **Red Hat, RHEL, CentOS, and Fedora:** /var/lib/pgsql/data/pg_log
- **Windows systems:** The messages are sent to the Windows Event Log

PostgreSQL severity	Meaning	Syslog severity	Windows Event Log
DEBUG 1 to DEBUG 5	This comprises the internal diagnostics.	DEBUG	INFORMATION
INFO	This is the command output for the user.	INFO	INFORMATION
NOTICE	This is helpful information.	NOTICE	INFORMATION
WARNING	This warns of likely problems.	NOTICE	WARNING
ERROR	This is the current command that is aborted.	WARNING	ERROR
LOG	This is useful for sysadmins.	INFO	INFORMATION
FATAL	This is the event that disconnects one session only.	ERR	ERROR
PANIC	This is the event that crashes the server.	CRIT	ERROR

Locating the database server's message log(3/3)

The following are the default server log locations:



Listing databases on the database server

You can create your own databases as well, like this: CREATE DATABASE my_database;

postgres=# \x (1) —---> first step

postgres=# select * from pg_database;
(2) —---> second step

```
daopattern=# \x
Расширенный вывод включён.
daopattern=# select * from pg_database;
-[ RECORD 1 ]--+-----------------------------------
oid            | 5
datname        | postgres
datdba         | 10
encoding       | 6
datlocprovider | c
datistemplate  | f
datallowconn   | t
datconnlimit   | -1
datfrozenxid   | 723
datminmxid     | 1
dattablespace  | 1663
datcollate     | Russian_Russia.1251
datctype       | Russian_Russia.1251
daticulocale   |
daticurules    |
datcollversion |
datacl         |
-[ RECORD 2 ]--+-----------------------------------
oid            | 1
datname        | template1
datdba         | 10
encoding       | 6
datlocprovider | c
datistemplate  | t
datallowconn   | t
datconnlimit   | -1
datfrozenxid   | 723
datminmxid     | 1
-- Далее  --
```

How much disk space does a database use?

The easiest way is to ask the database a simple query, like this:

```
SELECT pg_database_size(current_database());
```

```
daopattern=# SELECT pg_database_size(current_database());
-[ RECORD 1 ]----+--------
pg_database_size | 8163811
```

We can also see the total size of a table, including indexes and other related spaces, as follows:

```
postgres=# select pg_total_relation_size('person');
```

```
daopattern=#  select pg_total_relation_size('person');
 pg_total_relation_size
------------------------
                  32768
(1 ёЄЁюър)
```

Multiversion Concurrency Control (MVCC).

- Multiversion Concurrency Control (MVCC) is a database management technique that allows multiple transactions to access the database concurrently without conflicting with each other.
- It achieves this by maintaining multiple versions of each data item, enabling readers to see a consistent snapshot of the data while writers can make changes without blocking the readers.

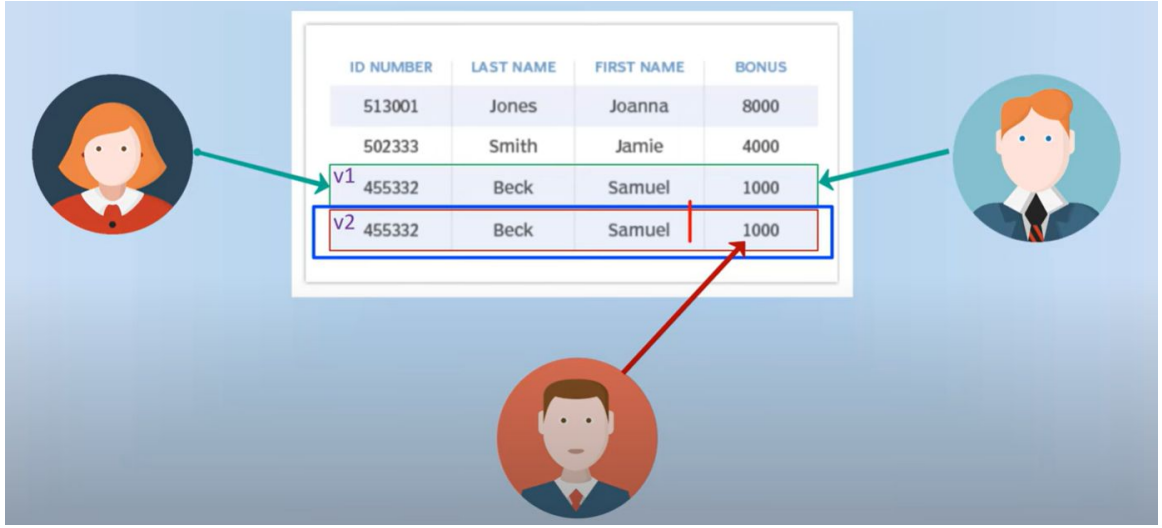
Here's a simple breakdown:

- Concurrency: Allows multiple transactions to happen at the same time.
- Versioning: Keeps different versions of data to manage changes.
- Consistency: Ensures each transaction sees a consistent view of the data.

In essence, MVCC improves performance and reduces conflicts in environments with many simultaneous transactions.

Multiversion Concurrency Control (MVCC).

- **Multiversion Concurrency Control (MVCC)** is a method of controlling the consistency of data accessed by multiple users concurrently.
- MVCC implements the snapshot isolation guarantee which ensures that each transaction always sees a consistent snapshot of data.



Quickly estimating the number of rows in a table

We can get a quick estimate of the number of rows in a table using roughly the same calculation that Postgres optimizer uses:

```
SELECT (CASE WHEN reltuples > 0 THEN
pg_relation_size(oid)*reltuples/(8192*relpages)
ELSE 0
END)::bigint AS estimated_row_count
FROM pg_class
WHERE oid = 'pg_class'::regclass;
```

```
daopattern=# SELECT (CASE WHEN reltuples > 0 THEN pg_relation_size(oid)*reltuples/(8192*relpages) ELSE 0
daopattern(# END)::bigint AS estimated_row_count
daopattern-# FROM pg_class
daopattern-# WHERE oid = 'pg_class'::regclass;
 estimated_row_count
---------------------
                 413
(1 ёЄЁюър)
```

Quickly estimating the number of rows in a table

First, get some details on the table from **pg_class:**

```
SELECT reltablespace, relfilenode FROM pg_class
WHERE  oid = 'person'::regclass;
```

```
daopattern=# SELECT reltablespace, relfilenode FROM pg_class
daopattern-# WHERE oid = 'person'::regclass;
-[ RECORD 1 ]-+------
reltablespace | 0
relfilenode   | 17770
```

REFERENCE

1: Medium .org

2.Java Guides(DAO)

Thank you!

Presented by

Tokhirjon Sadullaev

(tohirjonsadullayev387@gmail.com)