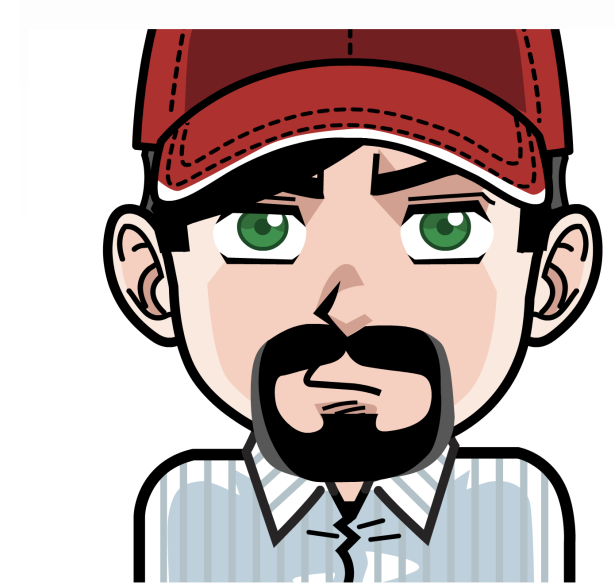
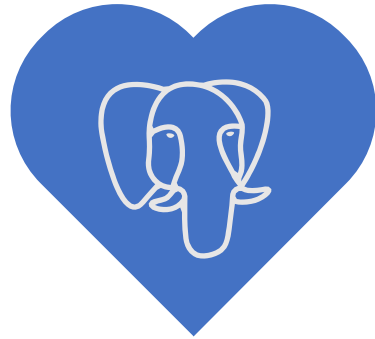


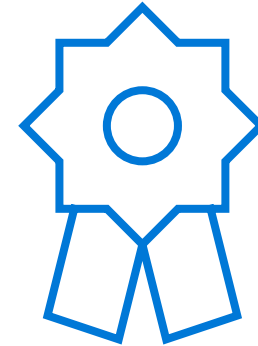
What's wrong with Postgres



Postgres is great

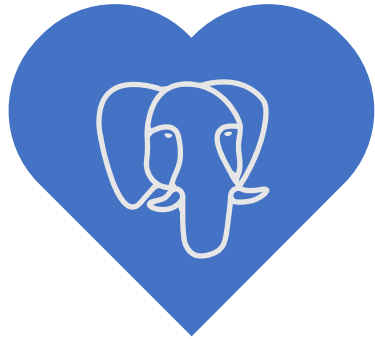


One of the most **loved** and **wanted** databases in Stack Overflow's 2019 Developer Survey

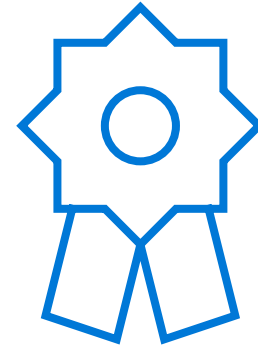


Ranked 2017 and 2018 **DBMS of the Year** by DB-Engine

Postgres is great, but it's not perfect



One of the most **loved** and **wanted** databases in Stack Overflow's 2019 Developer Survey



Ranked 2017 and 2018 **DBMS of the Year** by DB-Engine

Who am I

Helped build and grow Heroku Postgres

Over 1.5 million databases across team of 8 individuals

Lead product and cloud teams at Citus

Currently running product for Azure Postgres

Write a lot about Postgres – craigkerstiens.com

Curate postgres weekly

@craigkerstiens on twitter

But first



Biggest mistake

Michael Fuhr <mike(at)fuhr(dot)org> writes:

> On Wed, Jul 12, 2006 at 07:39:05PM +0300, Petronenko D.S. wrote:

>> Can i get data in postgre from non-postgre db?

> The name is PostgreSQL or Postgres, not postgre.

It might help to explain that the pronunciation is "post-gres" or "post-gres-cue-ell", not "post-gray-something".

I heard people making this same mistake in presentations at this past weekend's Postgres Anniversary Conference :- (Arguably, the 1996 decision to call it PostgreSQL instead of reverting to plain Postgres was the single worst mistake this project ever made. It seems far too late to change now, though.

regards, tom lane

Biggest mistake

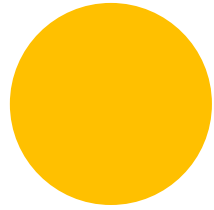
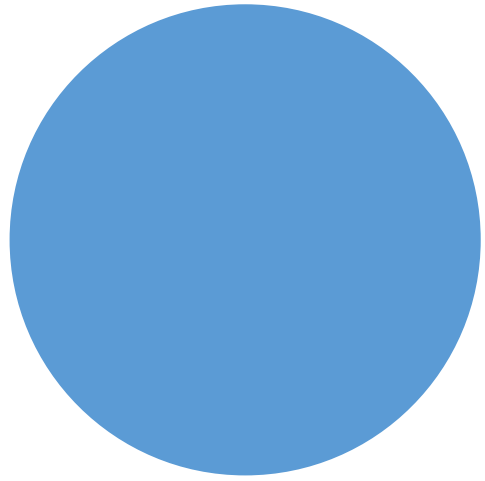
Michael Fuhr <mike(at)fuhr(dot)org> writes:
> On Wed, Jul 12, 2006 at 07:39:05PM +0300, Petronenko D.S. wrote:
>> Can i get data in postgre from non-postgre db?

> The name is PostgreSQL or Postgres, not postgre.

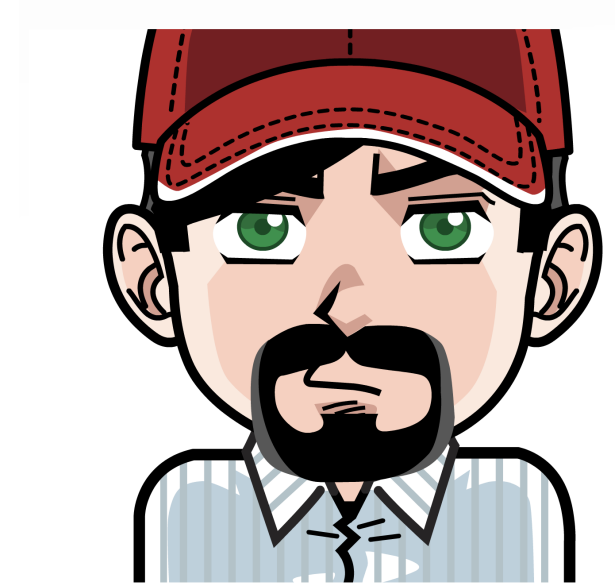
It might help to explain that the pronunciation is "post-gres" or "post-gres-cue-ell", not "post-gray-something".

I heard people making this same mistake in presentations at this past weekend's Postgres Anniversary Conference :- (Arguably, the 1996 decision to call it PostgreSQL instead of reverting to plain Postgres was the single worst mistake this project ever made. It seems far too late to change now, though.

regards, tom lane



What's wrong with Postgre



What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

05

Other areas

What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

05

Other areas

Documentation



Reference docs



On boarding



Tutorials/guides

Did you know Postgres
has a tutorial?

The tutorial

Before you can use PostgreSQL you need to install it, of course. It is possible that PostgreSQL is already installed at your site, either because it was included in your operating system distribution or because the system administrator already installed it. If that is the case, you should obtain information from the operating system documentation or your system administrator about how to access PostgreSQL.

If you are not sure whether PostgreSQL is already available or whether you can use it for your experimentation then you can install it yourself. Doing so is not hard and it can be a good exercise. PostgreSQL can be installed by any unprivileged user; no superuser (root) access is required.

The tutorial

Before you can use PostgreSQL you need to install it, of course. It is possible that PostgreSQL is already **installed at your site**, either because it was included in your operating system distribution or because the **system administrator already installed it**. If that is the case, you should obtain information from the operating system documentation or your system administrator about how to access PostgreSQL.

If you are not sure whether PostgreSQL is already available or whether you can use it for your experimentation then you can install it yourself. Doing so is not hard and it can be a good exercise. PostgreSQL can be installed by any unprivileged user; no superuser (root) access is required.

Chapter 16. Installation from Source Code

Table of Contents

[16.1. Short Version](#)

[16.2. Requirements](#)

[16.3. Getting the Source](#)

[16.4. Installation Procedure](#)

[16.5. Post-Installation Setup](#)

[16.5.1. Shared Libraries](#)

[16.5.2. Environment Variables](#)

[16.6. Supported Platforms](#)

[16.7. Platform-Specific Notes](#)

[16.7.1. AIX](#)

[16.7.2. Cygwin](#)

[16.7.3. macOS](#)

[16.7.4. MinGW/Native Windows](#)

[16.7.5. Solaris](#)

Go to chapter 16 to install

This chapter describes the installation of PostgreSQL using the source code distribution. If you are installing a pre-packaged distribution, such as an RPM or Debian package, ignore this chapter and read the packager's instructions instead.

16.7.3. macOS

On recent macOS releases, it's necessary to embed the "sysroot" path in the include switches used to find some system header files. This results in a configure script varying depending on which SDK version was used during configure. That shouldn't pose any problem in simple scenarios, but in something like building an extension on a different machine than the server code was built on, you may need to force use of a different sysroot. For example, use `PG_SYSROOT`, for example

```
make PG_SYSROOT=/desired/path all
```

To find out the appropriate path on your machine, run

```
xcodebuild -version -sdk macosk Path
```

Installing for mac

Note that building an extension using a different sysroot version than was used to build the core server is not really recommended; in the worst case, it can lead to hard-to-debug ABI inconsistencies.

You can also select a non-default sysroot path when configuring, by specifying `PG_SYSROOT` to configure:

```
./configure ... PG_SYSROOT=/desired/path
```

macOS's "System Integrity Protection" (SIP) feature breaks `make check`, because it prevents passing the needed setting of `DYLD_LIBRARY_PATH` to the executables being tested. You can work around that by doing `make install` before `make check`. Most PostgreSQL developers just turn off SIP

Install from source?

- Okay, so installing is wrong, but we can get past that
- I return to google and it helps

What's next

- Architecture fundamentals
- Creating a database
- Accessing the database

Accessing the database

Running the PostgreSQL interactive terminal program, called `psql`, which allows you to interactively enter, edit, and execute SQL commands.

Using an existing graphical frontend tool like `pgAdmin` or an office suite with ODBC or JDBC support to create and manipulate a database. These possibilities are not covered in this tutorial.

Writing a custom application, using one of the several available language bindings. These possibilities are discussed further in Part IV.

Accessing the database

Running the PostgreSQL interactive terminal program, called `psql`, which allows you to interactively enter, edit, and execute SQL commands.

Using an existing graphical frontend tool like `pgAdmin` or an office suite with ODBC or JDBC support to create and manipulate a database. These possibilities are not covered in this tutorial.

Writing a custom application, using one of the several available language bindings. These possibilities are discussed further in Part IV.

Part IV. Client Interfaces

This part describes the client programming interfaces distributed with PostgreSQL. Each of these chapters can be read independently. Note that there are also programming interfaces for client programs that are distributed separately and contain their own documentation ([Appendix H](#) lists some of the more popular ones). Readers of this part should be familiar with using SQL commands to manipulate and query the database (see [Part II](#)) and of course with the programming language that the interface uses.

Table of Contents

33. libpq - C Library

- 33.1. Database Connection Control Functions
- 33.2. Connection Status Functions
- 33.3. Command Execution Functions
- 33.4. Asynchronous Command Processing
- 33.5. Retrieving Query Results Row-by-Row
- 33.6. Canceling Queries in Progress
- 33.7. The Fast-Path Interface
- 33.8. Asynchronous Notification
- 33.9. Functions Associated with the `COPY` Command
- 33.10. Control Functions
- 33.11. Miscellaneous Functions
- 33.12. Notice Processing
- 33.13. Event System
- 33.14. Environment Variables
- 33.15. The Password File
- 33.16. The Connection Service File

Libpq huh?

Accessing the database

Running the PostgreSQL interactive terminal program, called psql, which allows you to interactively enter, edit, and execute SQL commands.

Using an existing graphical frontend tool like pgAdmin or an office suite with ODBC or JDBC support to create and manipulate a database. These possibilities are not covered in this tutorial.

Writing a custom application, using one of the several available language bindings. These possibilities are discussed further in **Part IV**.

Accessing the database

Running the PostgreSQL interactive terminal program, called psql, which allows you to interactively enter, edit, and execute SQL commands.

Using an existing graphical frontend tool like pgAdmin or an office suite with ODBC or JDBC support to create and manipulate a database. These possibilities are not covered in this tutorial.

Writing a custom application in C, using one of the several available language bindings. These possibilities are discussed further in **Part IV**.

What if:

- Getting started with
 - Ruby
 - Ruby and Rails
 - Ruby and Sequel
 - Python
 - Django
 - Python and SQLAlchemy
 - Java
 - Node
 - C
 - Go
 - PHP
 - Perl

Django getting started

Install Postgres

```
pip install django psycopg2
```

```
django-admin.py startproject myproject .
```

```
~/myproject/myproject/settings.py
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'myproject',  
        'USER': 'myprojectuser',  
        'PASSWORD': 'password',  
        'HOST': 'localhost',  
        'PORT': '',  
    }  
}
```

Documentation



Reference docs



On boarding



Tutorials/guides

How do I know what to look for

 postgresql docs speeding up a query



Chapter 14. Performance Tips

Table of Contents

- 14.1. Using `EXPLAIN`
- 14.2. Statistics Used by the Planner
- 14.3. Controlling the Planner with Explicit `JOIN` Clauses
- 14.4. Populating a Database
 - 14.4.1. Disable Autocommit
 - 14.4.2. Use `COPY`
 - 14.4.3. Remove Indexes
 - 14.4.4. Remove Foreign Key Constraints
 - 14.4.5. Increase `maintenance_work_mem`
 - 14.4.6. Increase `checkpoint_segments`
 - 14.4.7. Turn off `archive_mode`
 - 14.4.8. Run `ANALYZE` Afterwards
 - 14.4.9. Some Notes About `pg_dump`

Query performance can be affected by many things. Some of these can be manipulated by the user, while others are fundamental to the underlying design of the system. This chapter provides some hints about understanding and tuning PostgreSQL performance.

How do I know what to look for

🔍 postgresql docs slow query 

Documentation: 9.5: auto_explain - PostgreSQL

<https://www.postgresql.org/docs/auto-explain> ▼

PostgreSQL 9.5.19 Documentation ... To use it, simply load it into the **server**. ... Then you can track unexpectedly **slow queries** no matter when they happen.

Documentation: 9.5: EXPLAIN - PostgreSQL

<https://www.postgresql.org/docs/sql-explain> ▼

Sep 5, 2019 - For most **queries** the total cost is what matters, but in contexts such as a ... reading the system clock can **slow** down the **query** significantly on ...

Documentation: 9.2: Error Reporting and Logging - PostgreSQL

<https://www.postgresql.org/docs/static/runtime-config-logging>

This parameter can only be set in the **postgresql.conf** file or on the **server** command line. If csvlog is included in log_destination, **log** entries are output in "comma ...

Documentation: 10: 19.8. Error Reporting and ... - PostgreSQL

<https://www.postgresql.org/docs/runtime-config-logging> ▼

Where To **Log**. log_destination (string). **PostgreSQL** supports several methods for logging **server** messages, including stderr, csvlog and syslog. On Windows ...

How do we fix it?



Dedicated tutorial section



Can we pattern match for common searches?



Do we analyze our google traffic to see where people land?



Ask if docs were helpful?

What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

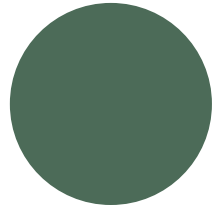
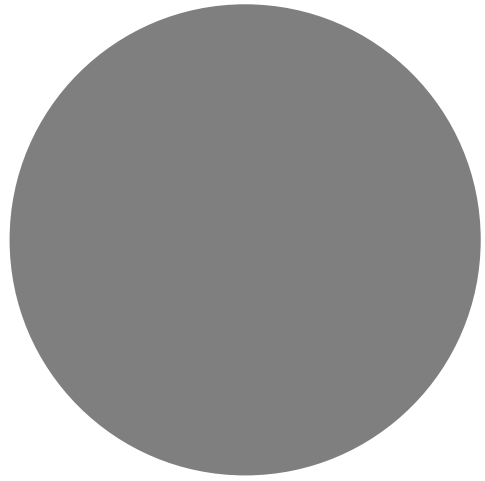
05

Other areas

Some assembly
required

- Config
- High availability
- Recovery





Django

Batteries included

Config

- postgresql.conf – settings
- Pg_hba.conf – access

Postgresql.conf

- Logging
- Memory
- Checkpoints
- Planner

Logging

- If you have syslog use it
 - People that have syslog know what it is
- If someone doesn't know what syslog is, shouldn't we give them something useful?

Shared buffers

- Default of 128MB Do we ever want this?
- Below 2GB of memory: This looks like an if statement to me
 - Set to 20%
- Below 32GB of memory:
 - Set to 25%
- Above 32GB of memory:
 - Set to 8GB

Can we have machine learning for this?

Work_mem

- Start low at 32-64 MB
- Look for 'temporary file'
 - Then raise it
- If you raise it too high look for OOMs
 - Then lower

Can we have machine learning for this?

But there are tools

- <https://postgresqlco.nf/en/doc/param/>
- <https://pgtune.leopard.in.ua/#/>
- <https://github.com/jfcoz/postgresqtuner>

And guides

- <https://thebuild.com/presentations/not-your-job-pgconf-us-2017.pdf>

It's up and running, ready for production!

I want availability

- Read replica?
- Primary/stand-by
- Active/active
- Load balancer?

Primary/stand-by

- Patroni
- PAF
- Repmgr
- Stolon
- Pg_auto_failover

Postgres
doesn't have
HA

NO EXIT

© Andy Singer





Recovery time still not
consistent

Backups

Do we all agree we need them?

Backups

- We give users pg_dump up to 100/500GB
- Do we give them something else beyond that?
- Same thing as HA, we don't have a solution, we allow you to pick a solution

Backups

- Postgres can know when it was last backed up
- Should we tell users when it hasn't been?

What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

05

Other areas

Vacuum



Connections

1. Establishing a connection
2. Connection overhead
3. The ceiling is too low



Establishing a connection

- Many databases running in cloud
- Want to securely communicate
- SSL or TLS negotiation aren't free
- 10-100ms to get a new connection



Connection overhead

- Every connection consumes 10MB of overhead
- Applications grab a “pool” of connections
- Developers: I need 4,000 connections
- Me: I see 4 active queries and 3996 idle queries



Low ceiling

- So we need to handle idle connection with a pooler
- Web apps start small, can scale massively, even the average app you haven't heard of, can need over 1,000 connections

What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

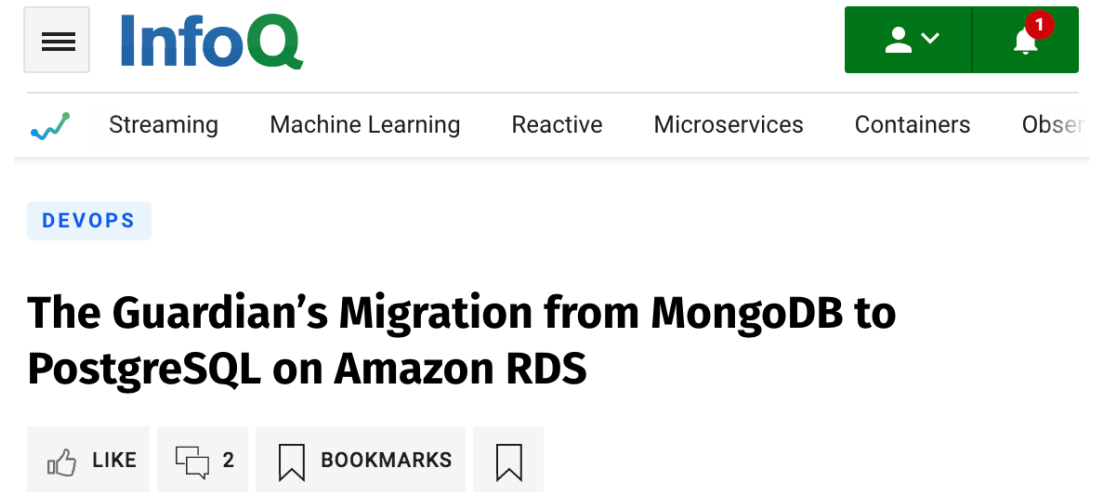
The community

05

Other areas

The project

- Businesses are betting on Postgres
- But businesses have certain expectations



The screenshot shows the top portion of a web page. At the top left is a hamburger menu icon. Next to it is the 'InfoQ' logo in blue and green. On the right side of the header, there are two green buttons: one with a white user icon and a dropdown arrow, and another with a white bell icon and a red notification bubble containing the number '1'. Below the header is a horizontal navigation bar with a blue line graph icon followed by the text 'Streaming', 'Machine Learning', 'Reactive', 'Microservices', 'Containers', and 'Observability'. Underneath this is a light blue pill-shaped button with the text 'DEVOPS'. The main content area features the article title 'The Guardian's Migration from MongoDB to PostgreSQL on Amazon RDS' in bold black text. At the bottom of the article preview, there are four buttons: 'LIKE' with a thumbs-up icon, a comment icon with the number '2', 'BOOKMARKS' with a bookmark icon, and another bookmark icon.



Roadmap

- When is the next release happening
- Can we commit before we commit?
- Roadmaps don't have to be feature checklists, they can be directional

Funding source

- What is needed

What's wrong with Postgres

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

05

Other areas

JSON

- JSONB vs. JSON
 - Could these be one?
- JSON users just want to insert data, they don't want to even create a table

Extensions

- What can't they do:
- Change the grammar
- We still have a lot of missing hooks

Extensions

- How do I discover them?
- How do I vet them?
- How do I install them?

- Right now only a power user feature

What's wrong with Postgre

01

Getting started

02

Getting to production

03

Operating at scale

04

The community

05

Other areas

Thanks!