PostgreSQL

Lesson Objectives

- Overview of PostgreSQL
- Data organization
- DDL
- DML
- Views
- Indexes
- Prog with PostgreSQL

PostgreSQL or Postgres

A Brief History of the Name

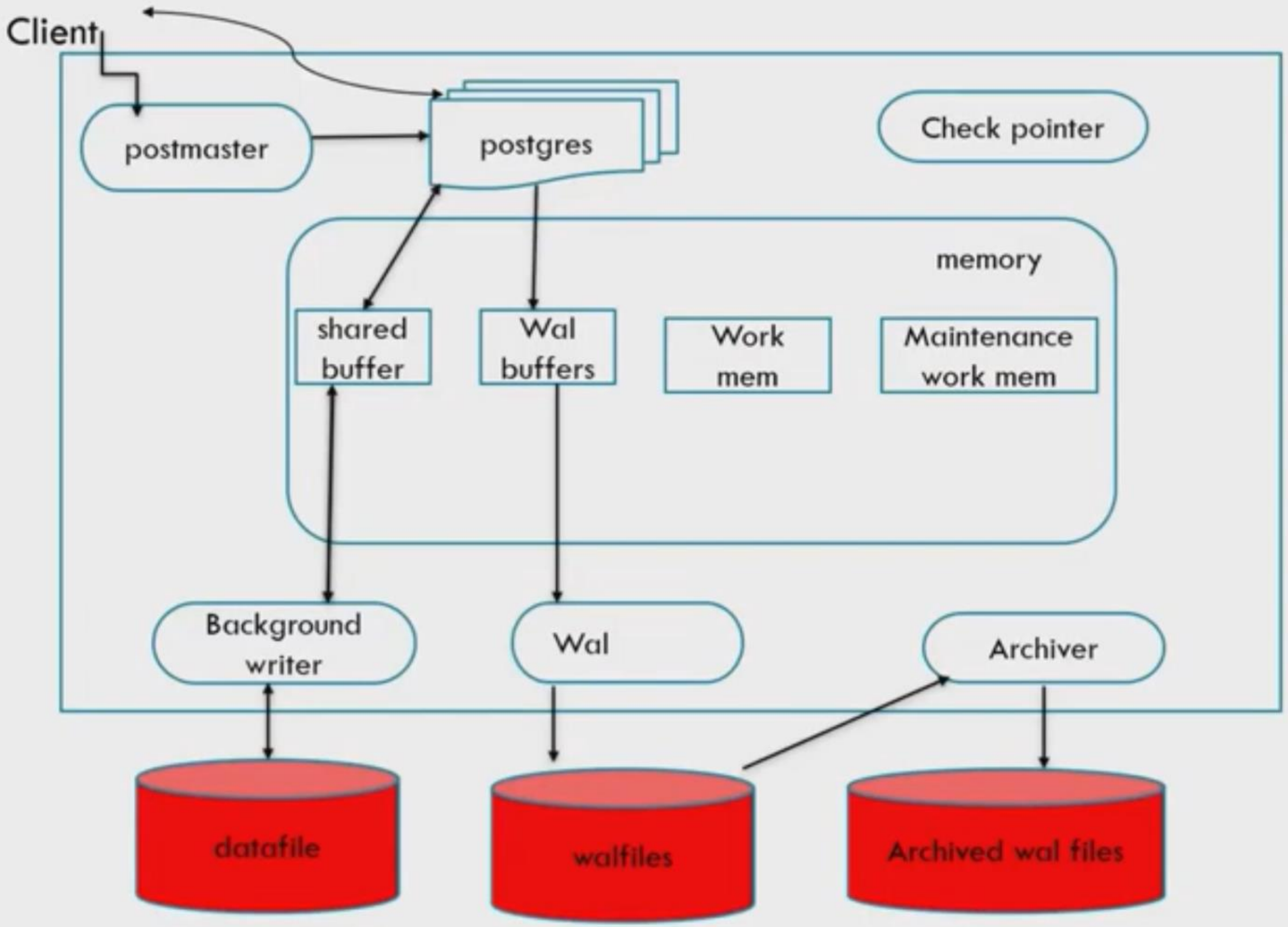
- Evolved from Ingres Project at University of California, Berkeley
- Ingres Team Leader – Michael Stonebraker
- 1982 – Michael left University
- 1985 – Michael returned to University
- Started to work on Post-Ingres project
- 1988 – The first prototype of product
- 1994 – Ingres based Quel Query Language interpreter was replaced with SQL Query language interpreter
- 1996 – PostgreSQL
- 1997 – First PostgreSQL release

What is PostgreSQL?

- Object Relational Database Management System (ORDBMS)
- Free
- Open Source
- Cross Platform
  - Linux
  - FreeBSD
  - Solaris
  - Microsoft Windows
  - Mac OS X (starting with OS X 10.7 Lion)

PostgreSQL Important Features

- SQL: 2011 Standard
- ACID compliant
- Indexes
- Views, Triggers, Procedures, Functions
- Relationships
- MultiVersion Concurrency Control (MVCC)
- SQL:2008 Datatypes
  - INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP
- Native Programming Interface
  - C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC


Client
postmaster
postgres
Check pointer
memory
shared buffer
Wal buffers
Work mem
Maintenance work mem
Background writer
Wal
Archiver
datafile
walfiles
Archived wal files

FLOW OF DATA

application connects to the *postmaster* process.

The *postmaster* checks the application's rights and - if successful - starts a new *postgres* process and connects it with the client application

the instance doesn't write or read the requested data directly to or from disk files. Instead, it buffers them in a shared memory area which is called the *shared buffers*.

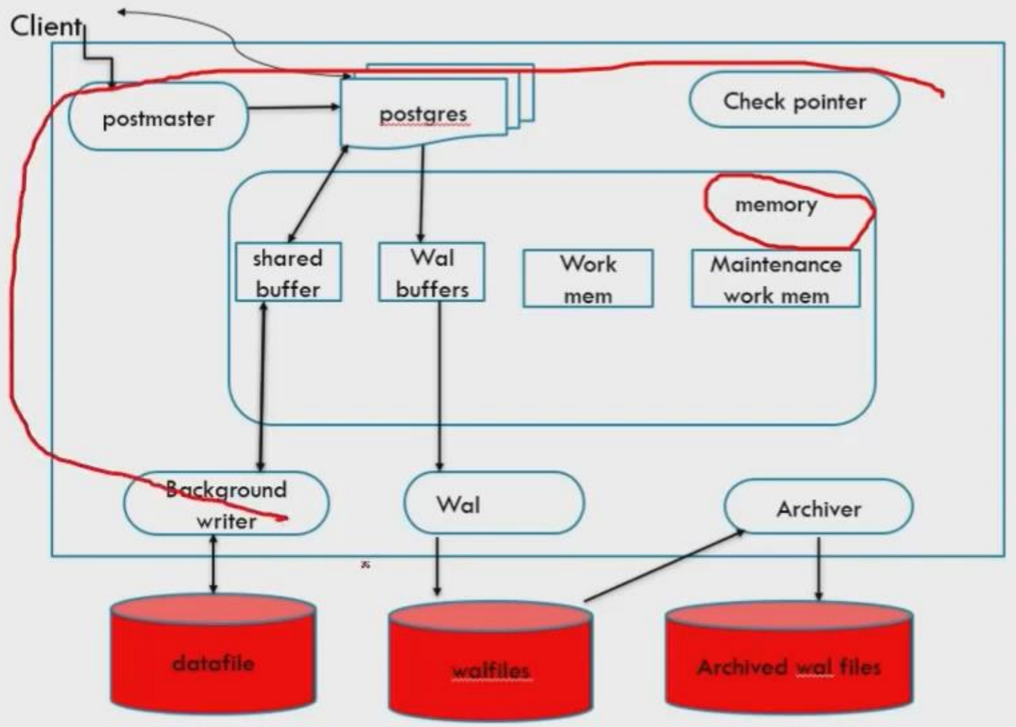
postgres process acts on the *shared buffers* and *WAL buffers* and manipulates their contents

client requests a COMMIT, the WAL writer process writes and flushes all WAL records resulting

from this transaction to the WAL file. As the WAL file - in contrast to the data files - is written strictly sequentially,

Periodically the *background writer* process checks the *shared buffers* for 'dirty' pages and writes them to the appropriate data files.

In essence the instance contains at least the three processes WAL writer, background writer, and checkpointer - and one postgres process per connection. In most cases there are some more processes running.



Optimize PostgreSQL Server Performance Through Configuration

- shared_buffer
  - Inside the postgresql.conf file, there is a parameter called shared_buffers. They are called 'shared' buffers because all of the background servers can access them. This parameter determines the amount of memory allocated to PostgreSQL for caching data.
  - PostgreSQL's design choice is to ensure compatibility on all supported machines and operating systems, this value is set conservatively low by default
  - A value over 25% of the system RAM can be useful if, for example, it is set such that the entire database working set of data can fit in cache, as this would greatly reduce the amount of time reading from disk.

Optimize PostgreSQL Server Performance Through Configuration

- wal_buffer
  - Write-Ahead Logging (WAL) is a standard method for ensuring integrity of data. Much like in the shared_buffers setting, PostgreSQL writes WAL records into buffers and then these buffers are flushed to disk.
  - The default size of the buffer is set by the wal_buffers setting- initially at 16MB. If the system being tuned has a large number of concurrent connections, then a higher value for wal_buffers can provide better performance.

Optimize PostgreSQL Server Performance Through Configuration

- work_mem
  - The value of work_mem is used for complex sort operations, and defines the maximum amount of memory to be used for intermediate results, such as hash tables, and for sorting.
  - It's important to ensure that the work_mem value is not set too high, as it can 'bottleneck' the available memory on the system as the application performs sort operations.
- maintenance_work_mem
  - While work_mem specifies how much memory is used for complex sort operations, maintenance_work_mem specifies how much memory is used for routine maintenance tasks, such as VACUUM, CREATE INDEX, and similar.
  - Unlike work_mem, however, only one of these maintenance operations can be executed at a time by a database session. As a result, most systems do not have many of these processes running concurrently, so it's typically safe to set this value much larger than work_mem, as the larger amounts of available memory could improve the performance of vacuuming and database dump restores. The default value for maintenance_work_mem is 64MB.

More concepts

- Checkpointer
  - Checkpointer creates checkpoints in the Write-Ahead Log. These checkpoints are used for recovery. A checkpoint indicates that all the information prior to the checkpoint has been updated. So at every checkpoint, dirty pages are flushed to disk
  - How often a checkpoint is begun depends on checkpoint_segments and checkpoint_timeout.
  - The integer, checkpoint_segments indicates the maximum number of log segments between two checkpoints. The default value is 3 segments, where each segment is usually 16 MB. This value can be adjusted in the postgresql.conf file.

PostgreSQL Limits

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

Prominent Users of PostgreSQL

- Yahoo
- Skype
- Instagram
- Disqus
- OpenStreetMap
- Reddit

Important Downloads

- **PostgreSQL Download**
  - http://www.postgresql.org/download/
- **Windows Graphical Installer**
  - PostgreSQL Server
  - pgAdmin III – a graphical IDE
  - http://www.postgresql.org/download/windows/
- **Samples Database - *pagilia***
  - http://www.postgresqltutorial.com/postgresql-sample-database/
  - http://bit.ly/pagilia

Connect to PostgreSQL Server

- Connect using psql – Terminal based front end
- Connect using pgAdmin- Web based front end

Install PostgreSQL Server- Ubuntu

- On ubuntu prompt issue the below command
- sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
- wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
- sudo apt-get update
- sudo apt-get install unzip
- sudo apt-get install postgresql

- For specific version you can issue below command
- sudo apt-get install postgresql-12

Install PostgreSQL Server- Ubuntu

- When you installed PostgreSQL, the installation process created a user account called postgres associated with the default postgres role.
- To connect to PostgreSQL using the postgres role, you switch over to the postgres account on your server by typing:
- sudo -i -u postgres

- Then, you can access the PostgreSQL using the psql by typing the following command
- Psql

Install PostgreSQL Server- Ubuntu

- To quit the PostgreSQL prompt, you run the following command:
- \q
- Type exit to go to ubuntu prompt

Load Database - Ubuntu

- sudo -i -u postgres
- Load Sample database
  - curl -O https://sp.postgresqltutorial.com/wp-content/uploads/2019/05/dvdrental.zip
- Unzip the file
  - unzip dvdrental.zip
- Access the PostgreSQL using the psql tool
  - psql
  - create database dvdrental;
- Quit psql by using \q command
- use the pg_restore tool to restore the dvdrental database
  - pg_restore --dbname=dvdrental --verbose dvdrental.tar

Load Database - Ubuntu

- Access the PostgreSQL using the psql tool
  - psql
  - \c dvdrental;
- Test by issuing the below command
  - select count(*) from film;

Connect to PostgreSQL Server

- Using psql – On Linux
  - **Locate**
    - The easiest way to connect is to check whether you already have psql in your environment variables on not. You can try the following command on the terminal:
      - [root@localhost data] # which psql : This gives you the path, and also, since the OS knows the location, you will not need to browse for it. If it gives errors then you can locate psql utility by following command
      - [root@localhost /] # find / -name psql
  - **Connect**
    - [root@localhost data] # psql -h <hostname or ip address> -p <port number of remote machine> -d <database name which you want to connect> -U <username of the database server>

Connect to PostgreSQL Server

- Using psql – On Windows
  - Locate psql
    - You can find psql in the Program Files, and you should be able to launch it in a command prompt simply by clicking on it

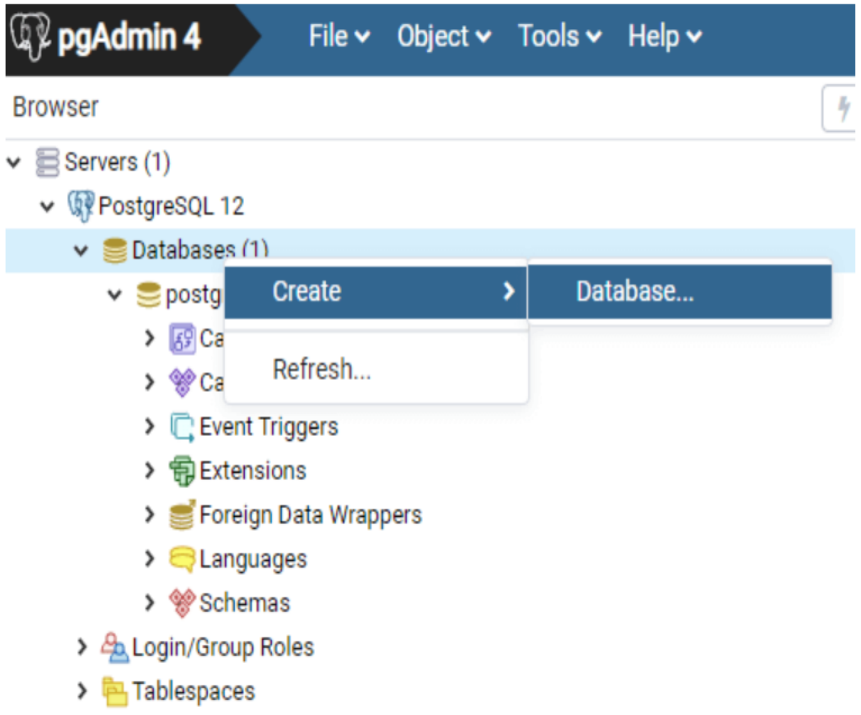
Connect to PostgreSQL Server

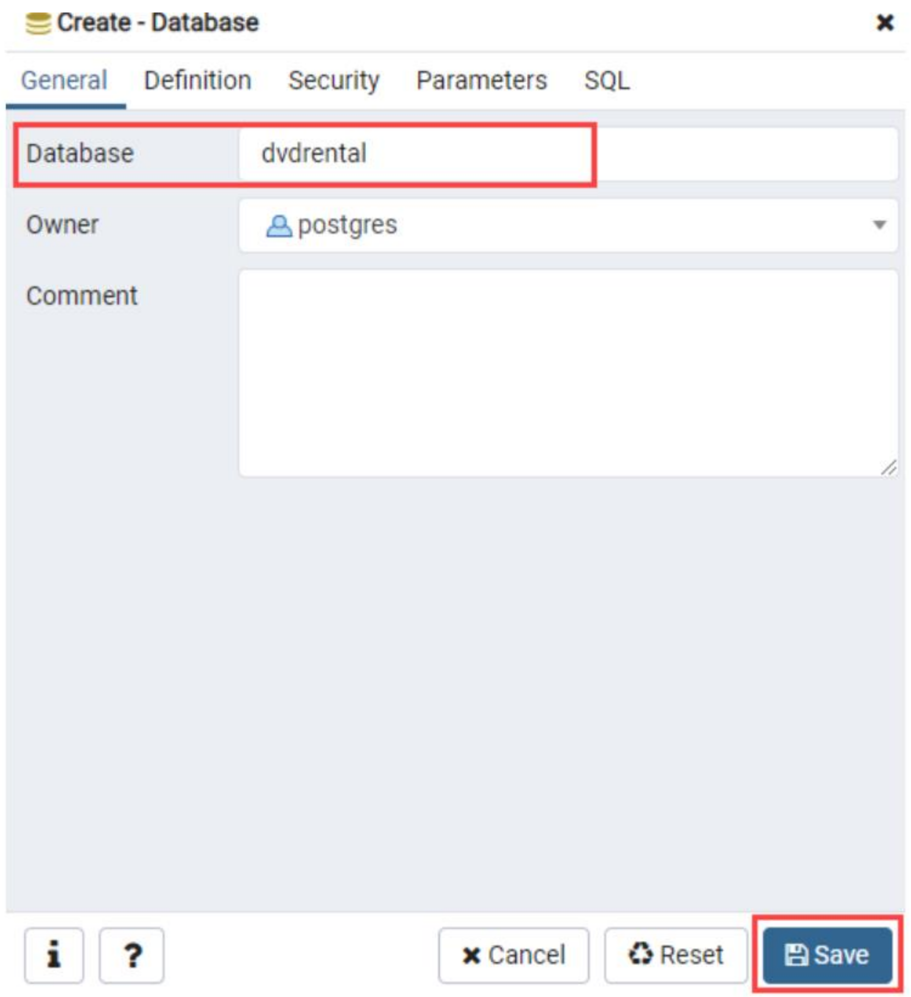
- Connect using pgAdmin4
  - **On Linux pgAdmin is available under Programming in the Applications menu**
  - **On Windows it is available under Program Files**

Create database in PostgreSQL – Using psql

- Open psql terminal prompt and issue the below commands
  - Create database dvdrental
  - pg_restore -U postgres -d dvdrental C:\sampledb\dvdrental.tar

Create database in PostgreSQL – Using pgAdmin

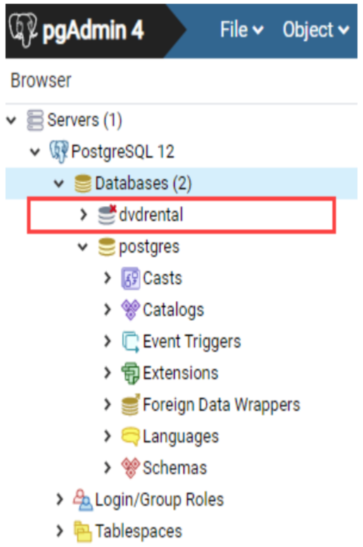




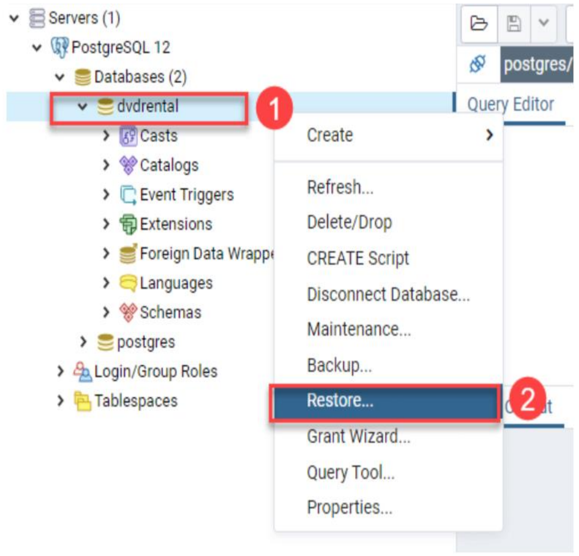
Database: dvdrental
Owner: postgres
Comment:

Create database in PostgreSQL – Using pgAdmin

You'll see the new empty database created under the Databases node:



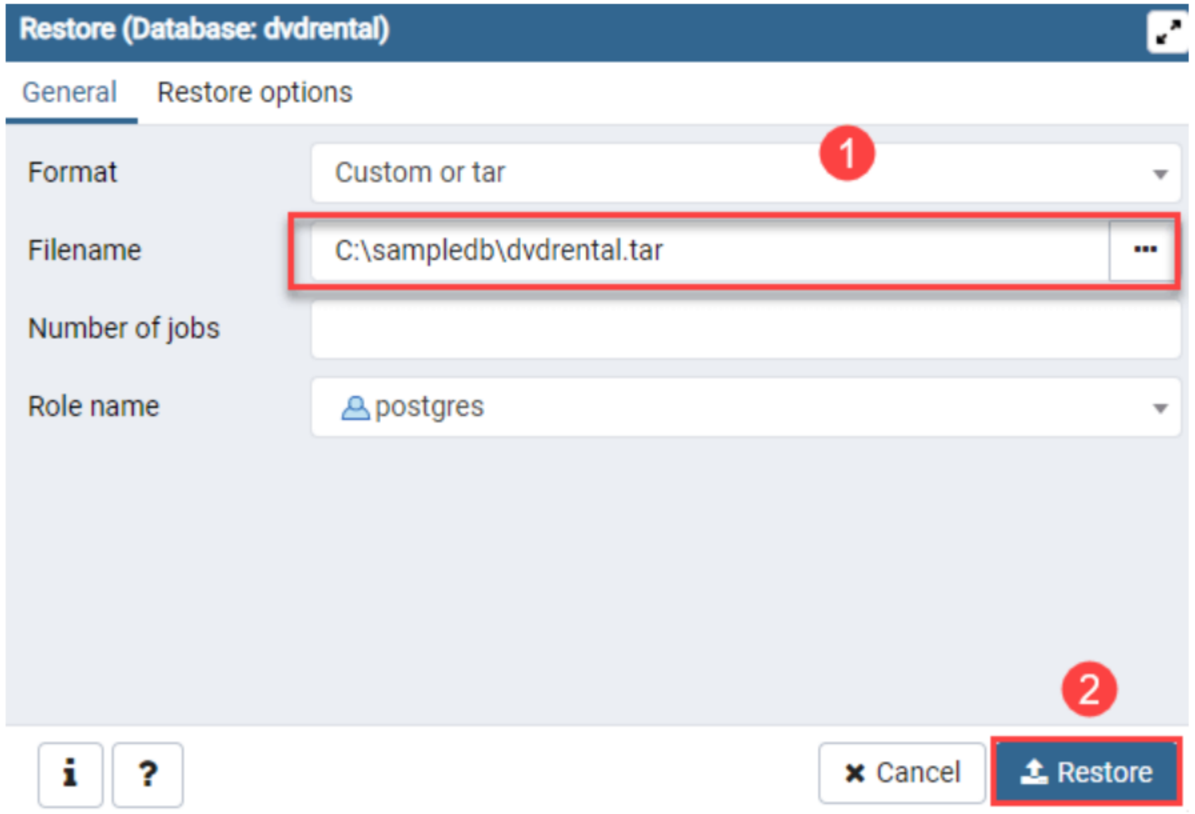
Fourth, right-click on the **dvdrental** database and choose **Restore...** menu item to restore the database from the downloaded database file:



Fifth, enter the path to the sample database file e.g., **c:\sampledb\dvdrental.tar** and click the **Restore** button:

Create database in PostgreSQL – Using pgAdmin

Fifth, enter the path to the sample database file e.g., **c:\sampledb\dvdrental.tar** and click the **Restore** button:



Restore (Database: dvdrental)

General | Restore options

Format	Custom or tar
Filename	C:\sampledb\dvdrental.tar
Number of jobs	
Role name	postgres