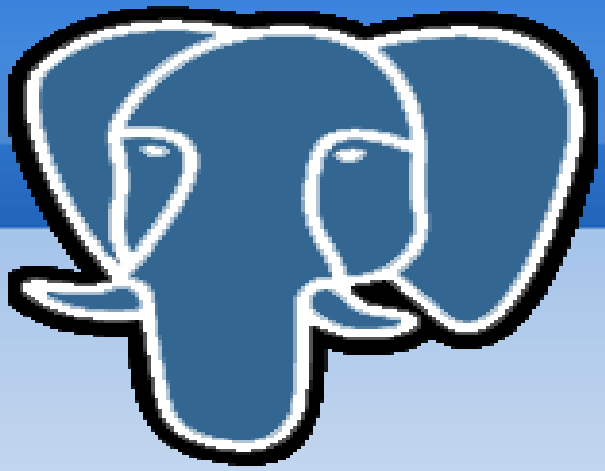


PostgreSQL

Administration under Linux

(PG SYSTEM ARCHITECTURE - INSTALLATION FROM SOURCE - DB CONFIG. AND BACKUPS OVERVIEW)

May 2018

MOUSSA H.

Agenda

0 – Why PostgreSql over MySql

1 – Presentation

2 – Distrituted Database Architecture

3 - Postgresql System Architecture

4 – Postgresql partition system

5 – Postgresql Custom Installation

6 – Post-Installation configuration

7 – Backup & Restore

Why Postgresql ?

Name	MySql	Postgresql
APIs and other access methods	ADO.NET JDBC ODBC	native C library streaming API for large objects ADO.NET JDBC ODBC
OS	FreeBSD Linux OS X Solaris Windows	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows
Developer	Oracle	PostgreSQL Global Development Group

Reasons for using PostgreSQL over MySQL:

- Performance
- Complex database design
- Moving away from Oracle, Sybase, or MSSQL
- Complex rule sets (i.e., business rules)
- Use of procedural languages on the server
- Transactions
- Use of stored proc

Presentation

PostgreSQL is a powerful, open source object-relational database system. It has more than 20 years of active development phase and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness.

PostgreSQL (pronounced as **post-gress-Q-L**) is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge.

Présentation

PostgreSQL is probably the most advanced database in the open source relational database market. It was first released in 1989, and since then, there have been a lot of enhancements. According to db-engines, it is one of the most used database at the time of writing.

A Brief History of PostgreSQL

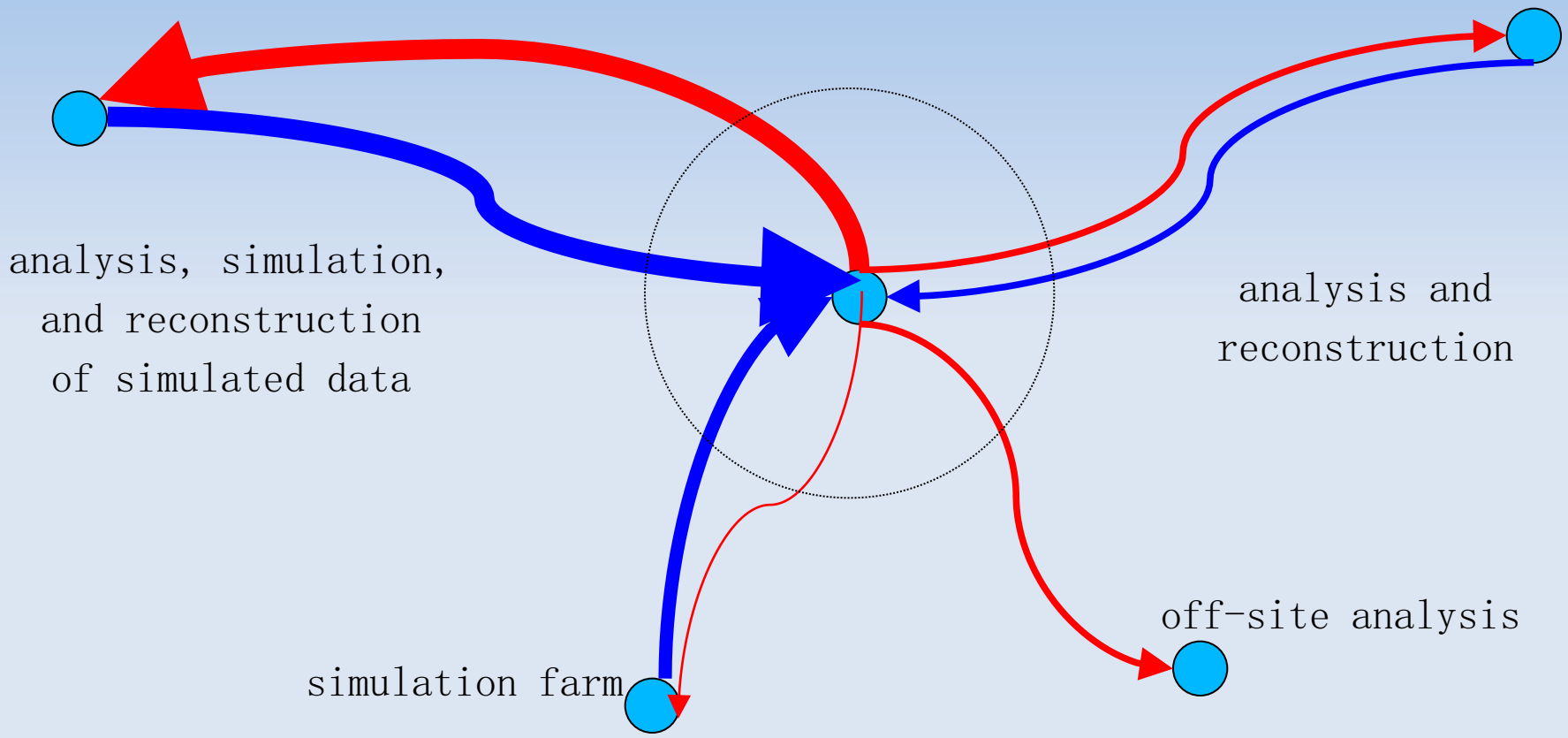
- PostgreSQL, originally called Postgres, created at UCB by a computer science professor named Michael Stonebraker. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres
- **1977-1985** − A project called INGRES was developed.
  - Proof-of-concept for relational databases

A Brief History of PostgreSQL

- **1986-1994** – POSTGRES
  - Development of the concepts in INGRES with a focus on object orientation and the query language – Quel
- **1994-1995** – Postgres95
  - Support for SQL was added in 1994
  - Released as Postgres95 in 1995
  - Re-released as PostgreSQL 6.0 in 1996
  - Establishment of the PostgreSQL Global Development Team

II. Distribured Database Architecture (DDB)

Problematics



Large amount of data is moved between Main and offsite institutions

- Central File Catalog not updated online during production at remote sites
- Remote sites maintained their own catalogs of local files

Hard to keep File Catalogs up-to-date, required a lot of work

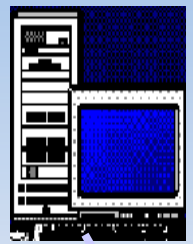
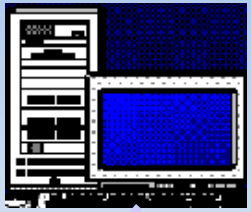
What we were looking for

SOLUTION

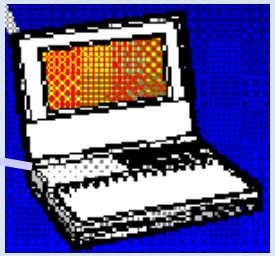
Provide write permissions to the sites that store portions of the data set

Objective: Exposes primary DB to world

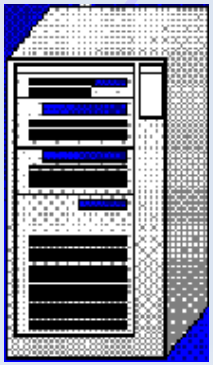
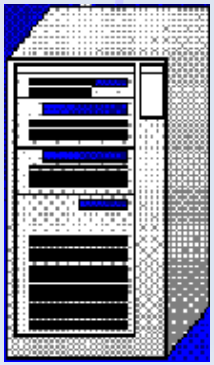
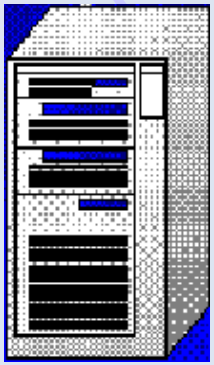
Architecture DBD

Clients

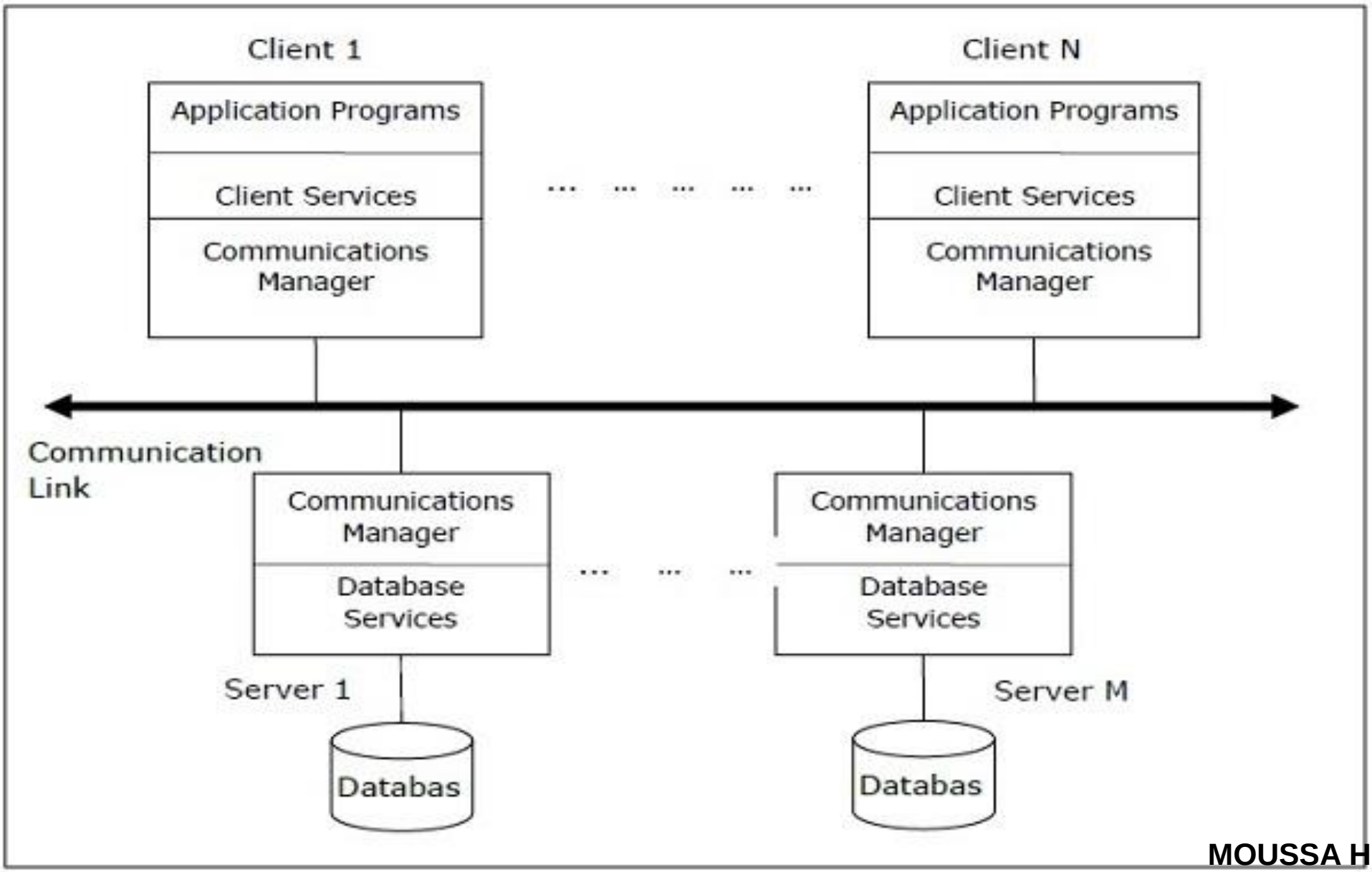


LAN

Serveurs

Architecture DBD



Distributed Database Management System

A distributed database management system (DDBMS) is a software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
- It maintains confidentiality and data integrity of the databases.

Distributed Database

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network.

Postgresql and DDBMS

PostgreSQL is not natively distributed. The feature configuration depends on your needs.

Typically distributed means a database that can work as a group of several nodes (instances, servers, etc.) working together.

Most relational databases are not built for this architecture and instead focus on being a single-instance installation that works on one server.

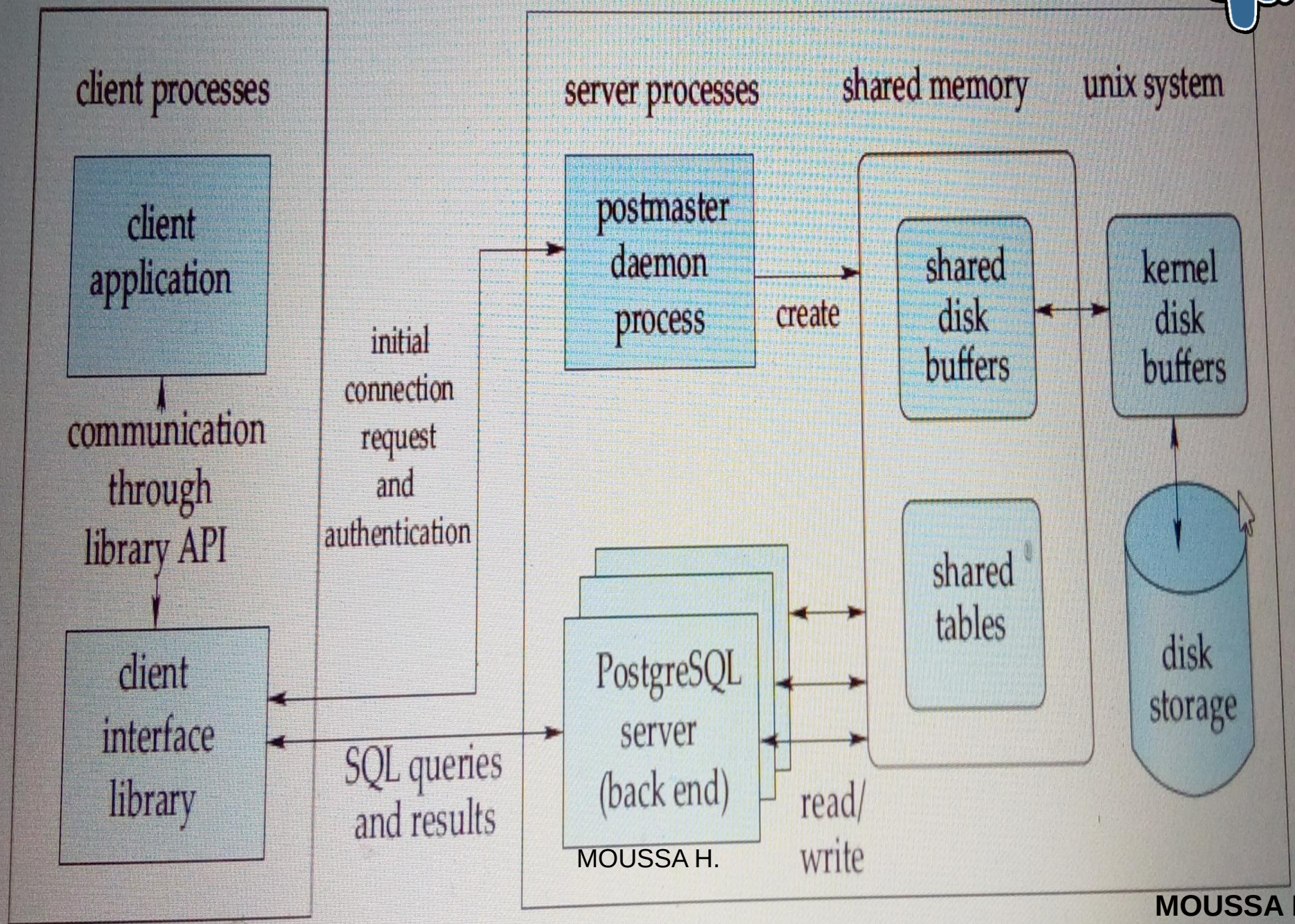
Transforming PostgreSQL into a Distributed

1) The new pg_shard (developed by Citusdata) effectively transforms a single PostgreSQL instance into a distributed parallel database that can run on any number of nodes for the purpose of driving higher scalability, I/O, and availability.
2) Postgres-XL - a forked version of postgres designed to be distributed.
3) Using dblink.

III. Postgresql System architecture

The physical structure of PostgreSQL is very simple. It consists of shared memory and a few background processes and data files

See structure below:


client processes
server processes
shared memory
unix system
client application
communication through library API
client interface library
initial connection request and authentication
postmaster daemon process
create
shared disk buffers
kernel disk buffers
shared tables
disk storage
PostgreSQL server (back end)
SQL queries and results
read/ write

MOUSSA H

* Shared Memory

Shared Memory refers to the memory reserved for database caching and transaction log caching. The most important elements in shared memory are Shared Buffer and WAL buffers

Shared Buffer

The purpose of Shared Buffer is to minimize DISK IO. For this purpose, the following principles must be met

- You need to access very large (tens, hundreds of gigabytes) buffers quickly.
- You should minimize contention when many users access it at the same time.
- Frequently used blocks must be in the buffer for as long as possible

- WAL Buffer

The WAL buffer is a buffer that temporarily stores changes to the database. The contents stored in the WAL buffer are written to the WAL file at a predetermined point in time. From a backup and recovery point of view, WAL buffers and WAL files are very important.

PostgreSQL Process Types

PostgreSQL has four process types.

- Postmaster (Daemon) Process
- Background Process
- Backend Process
- Client Process

Postmaster Process

The Postmaster process is the first process started when you start PostgreSQL. At startup, performs recovery, initialize shared memory, and run background processes. It also creates a backend process when there is a connection request from the client process.

Client Process

Client Process refers to the background process that is assigned for every backend user connection.Usually the postmaster process will fork a child process that is dedicated to serve a user connection.

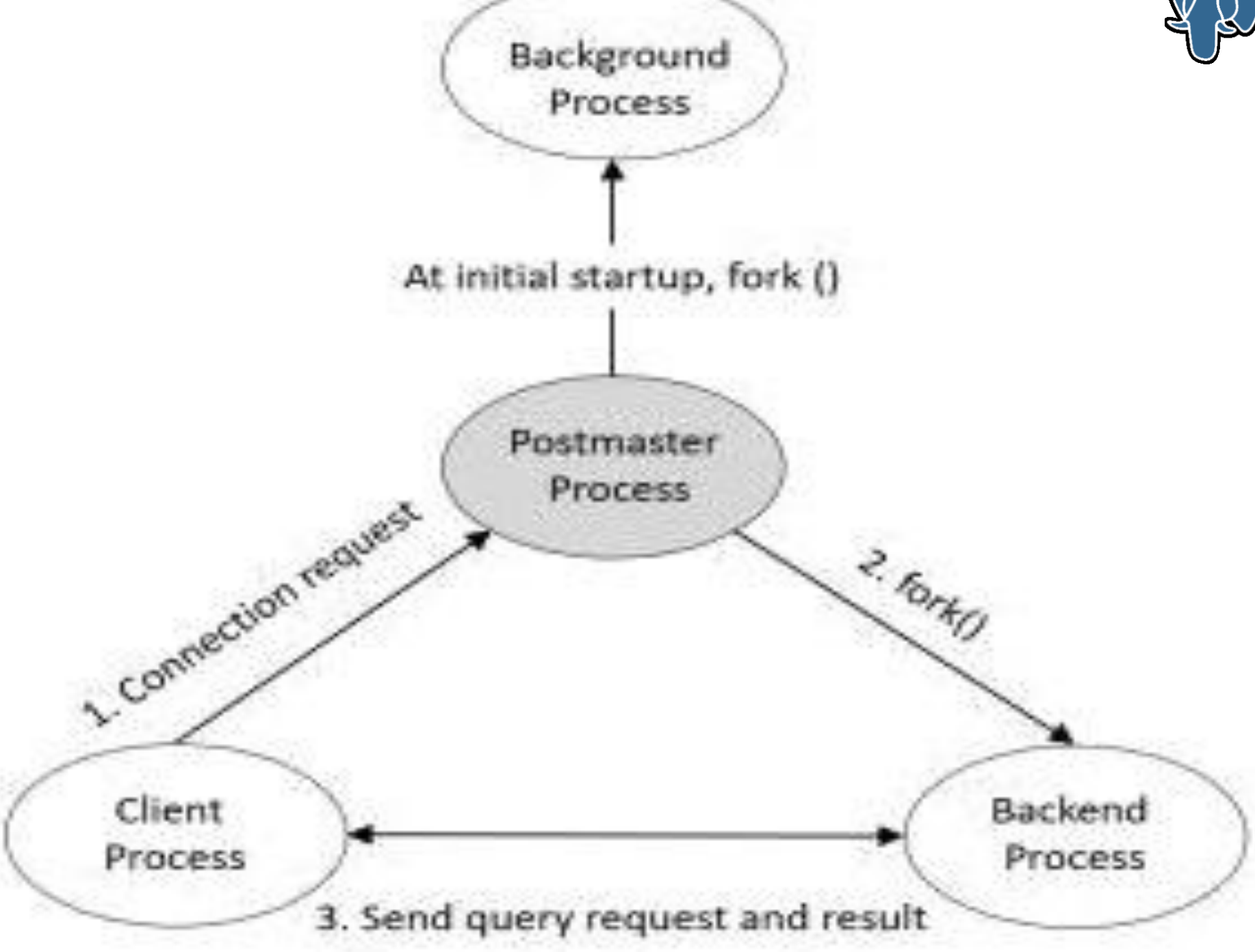
Backend Process

The backend process performs the query request of the user process and then transmits the result. Some memory structures are required for query execution, which is called local memory. The main parameters associated with local memory are:

1) work_mem Space used for sorting, bitmap operations, hash joins, and merge joins. The default setting is 4 MB.

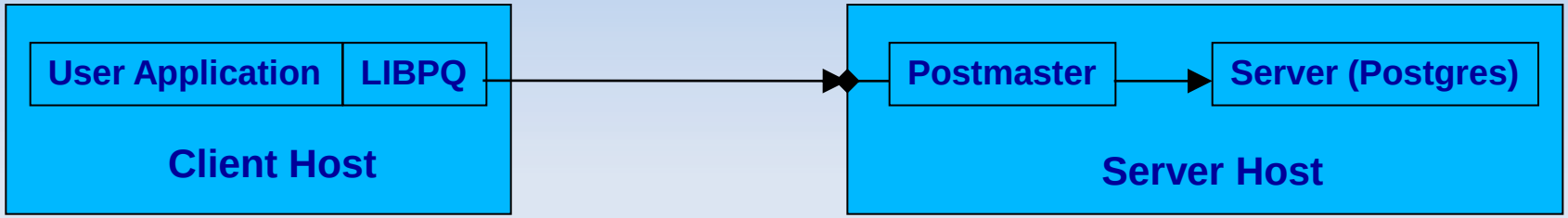
2) Maintenance_work_mem Space used for Vacuum and CREATE INDEX . The default setting is 64 MB.

3) Temp_buffers Space used for temporary tables. The default setting is 8 MB.


Background Process
At initial startup, fork ()
Postmaster Process
1. Connection request
2. fork()
Client Process
Backend Process
3. Send query request and result

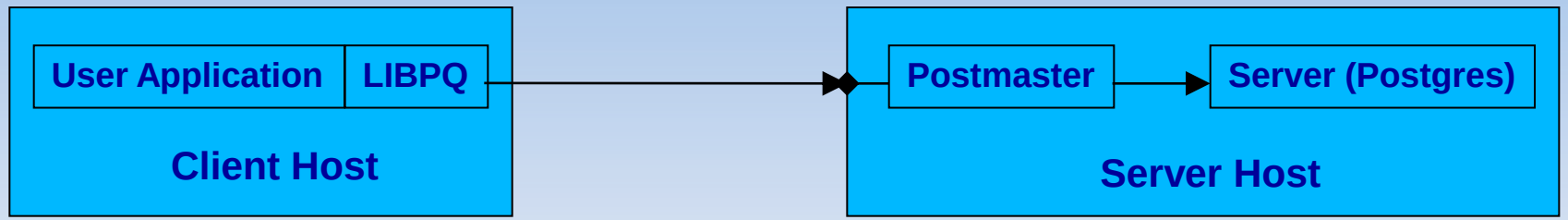
Connection mecanisne

- PostgreSQL uses client-serveur modèle : « one process by user".



- A PostgreSQL session consists of many coorporative processes:
  - a demon process supervisor (postmaster)
  - client application (p.ex. PSQL, PGAdmin3)
  - DB server (processus Postgres)

Connection mecanisne



- The clients that want to access the db call tha library
- This library send the user request throught the network to the Postmaster.
- Postmaster starts a new server process (a process by user) and connect this process to client application process.
- At this point, the client and server processes communicate without intervension of the Postmaster

4.

Postgresql Partition system

Database Structure

Here are some things that are important to know when attempting to understand the database structure of PostgreSQL.

1) **PostgreSQL consists of several databases. This is called a database cluster.**
2) **When initdb () is executed, template0 , template1 , and postgres databases are created.**
3) **The template0 and template1 databases are template databases for user database creation and contain the system catalog tables.**
4) **The list of tables in the template0 and template1 databases is the same immediately after initdb (). However, the template1 database can create objects that the user needs.**
5) **The user database is created by cloning the template1 database.**

If you query the pg_database view after initdb() , you can see that the template0 , template1 , and postgres databases have been created.

> Select * from pg_database;

Tablespaces

- The pg_default and pg_global tablespaces are created immediately after initdb().
- If you do not specify a tablespace at the time of table creation, it is stored in the pg_dafault tablespace.
- Tables managed at the database cluster level are stored in the pg_global tablespace.
- The physical location of the pg_default tablespace is $PGDATA\base.
- The physical location of the pg_global tablespace is $PGDATA\global.
- One tablespace can be used by multiple databases. At this time, a database-specific subdirectory is created in the table space directory.
- Creating a user tablespace creates a symbolic link to the user

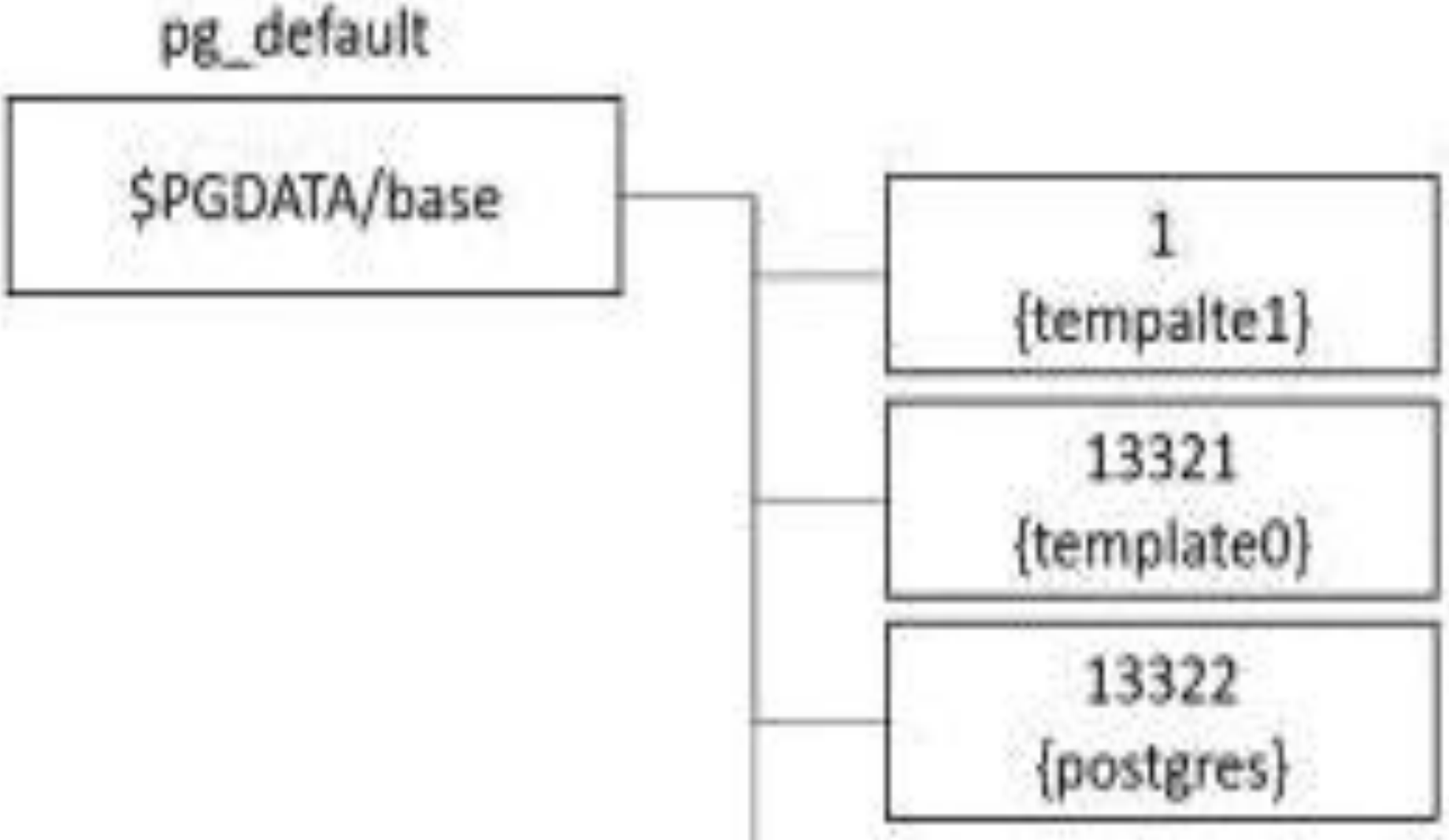
pg_default tablespace

If you query pg_tablespace after initdb (), you can see that the pg_default and pg_global tablespaces have been created.

- ***Select * from pg_tablespace;***

The location of the pg_default tablespace is $PGDATA\base.

Pg_default tablespace and database relationships from a physical configuration perspective

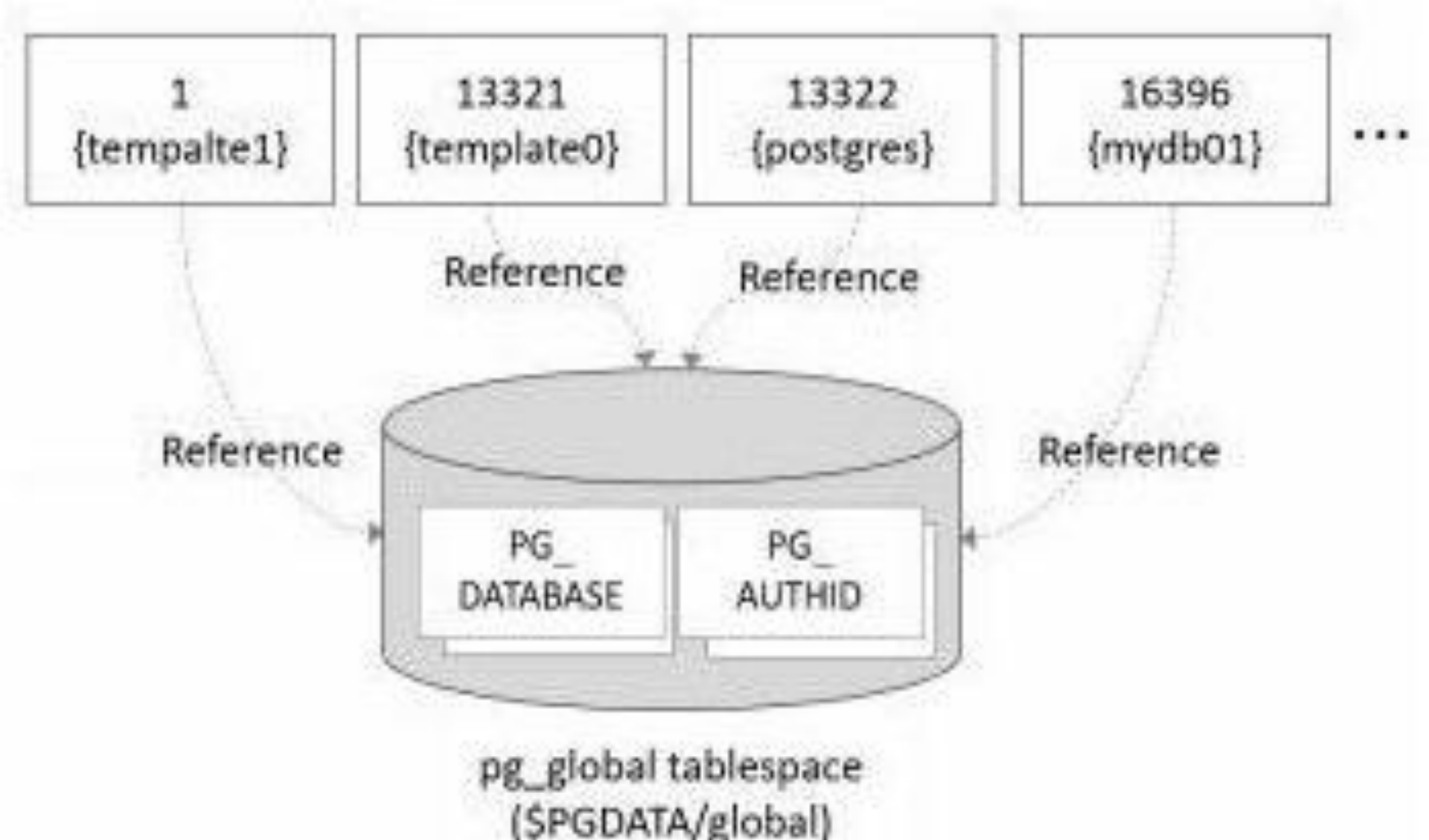


pg_global tablespace

The pg_global tablespace is a tablespace for storing data to be managed at the 'database cluster' level.

- For example, tables of the same type as the pg_database table provide the same information whether they are accessed from any database.
- The location of the pg_global tablespace is $PGDATA\global.

Relationship between pg_global tablespace and database



Create User Tablespace

Basic syntax:

create tablespace btsName
location ‘/dir';

postgres=# create tablespace tbs_test
location ‘/usr/local';

PostgreSQL - Schema

- A **schema** is a named collection of tables. A schema can also contain views, indexes, sequences, data types, operators, and functions. Schemas are analogous to directories at the operating system level, except that schemas cannot be nested. PostgreSQL statement CREATE SCHEMA creates a schema.

The basic syntax of CREATE SCHEMA is as follows −

CREATE SCHEMA name;

Where *name* is the name of the schema.

Syntax to Create Table in Schema

The basic syntax to create table in schema is as follows −

CREATE TABLE myschema.mytable (...);

5. Installation

1 - Pré-requis

- version used is : postgresql-9.5.0
- types of installations (code source, rpm, deb, …)
- the importance of installing from codes sources is that it works on all the differents distributions and to be able to control the installation parameters.
- dir:
  - dir containing the sources : /opt/sources/
  - installation dir: /usr/local/ (par défaut)
  - you can use your desire dir

Installation

- Installation steps :

- Errors may occur durant préparation and compilation, because of libraries dependancies.

- gcc / g++ : GNU C compiler
- libreadline : psql interactif monitor library
- zlib : library useful for archives creation during DB backups in tar or compressed archive .

- Login in as « root » to have rights to install the programs.

Installation

Super user

- Why the super user:
  - For security raisons, it is advice to create « postgres » user, to avoid exposures such as dammages
- Creation of the super user:

```
$> sudo adduser postgres
```

Installation

Preparation of installation

- Creattion of the sources dir in the root dir /opt :

```
$> sudo  mkdir  /opt/sources
```

- Copy the binary file in the sources dir:

```
$> sudo  cp postgresql-9.5.0.tar.gz
/opt/sources/
```

- Unzip the source:

```
$> cd  /opt/sources/
$>sudo  tar -zxvf postgresql-9.5.0.tar.gz
```

Installation

Préparation de l'installation

- **configuration of the environnement : place yoursel in postgresql-9.5.0 dir and type**

```
$> ./configure
Or
$> ./configure –without-readline –without-zlib
```

- For more info, type:

```
$> ./configure  - - help
```

- Download necessary libraries:

```
$> sudo aptitude install g++
$> sudo aptitude install libreadline5-dev
$> sudo aptitude install zlibc
$> sudo aptitude install zlibg-dev
```

- Or:

```
$> sudo aptitude install g++ libreadline5-dev zlibc zlibg-dev
```

Installation

Compilation et installation

- To compile :

```
$> make
```

You will receive msg: All of postgresql successfully made. Ready to install

- To see compilation errors:

```
$> sudo make  - - check
```

- To lunch installation :

```
$> sudo make  install

If Ok, will receive: postgresql installation complete
```

- Postgres will be installed in the following dir:
  - /usr/local/pgsql/
  - /usr/local/pgsql/bin/ : contents binary files(commandes)
  - /usr/local/pgsql/lib/ : Postgres libraries

Installation

DB config

- Create the super user (postgres)

```
$>sudo adduser postgres
```

- Creation of the data dir:

```
$> sudo mkdir /usr/local/pgsql/data
```

- Define postgres ownership:

```
$> sudo chown  postgres  /usr/local/pgsql/data
```

- Become postgres for the DB initialisation

```
$> su – postgres
$>  /usr/local/pgsql/bin/initdb  –D
/usr/local/pgsql/data

NB: this cmd is executed one in a DB life
```

"Success. You can now start the DB server using: "

/usr/local/pgsql/bin/pg_ctl –D /usr/local/pgsql/data –l logfile start

Starting the Server:

$> /usr/local/pgsql/bin/postgres –D /usr/local/pgsql/data –l logfile > logfile 2>&1 &

NB: this starting of the server is required at any boot of the system.

6. Postgresql post - Installation Configuration

Creation of the DB:

$> /usr/local/pgsql/bin/createdb expenditure2018

To connect to the created DB:

$> /usr/local/pgsql/bin/psql expenditure2018

Will prompt:

expenditure2018=#

Waiting for cmds, to exit enter: \q

SOME BASIC CMD

To Display the list of DB (connected user is postgres):

postgres=# \l

To display the list of users

postgres =# \du

To change à user ppasswd

postgres =#ALTER USER postgres WITH PASSWORD 'pwd';

To create new user:

postgres =#CREATE USER u1 WITH PASSWORD 'pwd';

SOME BASIC CMD

Giving privilege to the new DBA

postgres =#ALTER USER u1 WITH SUPERUSER;

Droping a user

postgres =#DROP USER u1;

To know about all the cmd in psql

postgres =# man psql

Configuration of env. Variables:

LD_LIBRARY_PATH and PATH

- Make sure you are postgresl and edit .profile by adding the lines below

```
export LD_LIBRARY_PATH=/usr/local/pgsql/lib
export PATH=/usr/local/pgsql/bin:$PATH
export PGDATA=/usr/local/pgsql/data
```

- You can then start server and stop by

```
$> pg_ctl start | stop
```

Auto start

On Linux systems either add

/usr/local/pgsql/bin/pg_ctl start -l logfile -D /usr/local/pgsql/data

to /etc/rc.d/rc.local

or /etc/rc.local

or look at the file

contrib/start-scripts/linux

in the PostgreSQL source distribution.

importants files

/usr/local/pgsql/data/postgresql.conf

set listen_address=‘*’

/usr/local/pgsql/data/pg_hba.conf

Note: Data_directory = ‘/usr/local/pgsql/data’

hba_file= ‘/usr/local/pgsql/data/pg_hba.conf’

Ident_file= ‘/usr/local/pgsql/data/pg_ident.conf’

Installation client pgadmin

$ sudo apt-get install postgresql-9-5 pgadmin3

After installation, reload config and system files by:

$ pg_ctl reload

Then lunch pgadmin:

$ pgadmin3

Installation client phppgadmin

$ sudo apt-get install phppgadmin

After installation, apache2 and phppgadmin

$ vi /etc/apache2/conf-available/phppgadmin.conf

Comment required local and add allow from all

$ vi /etc/phppgadmin/config.inc.php

Set $config[‘extra_login_security’]= false;

$ pgadmin3

After system files config. Restart apache and postgres service:

$ systemclt restart postgresql

$ systemclt restart apache2

7. BACKUP & RESTORE

Types of Backups

There exist 2 types:

a) SQL and dump
b) File system backups (not discussed here, required advanced knowledge of linux)

a) SQL and dump

SQL Dump

The idea behind this dump method is to generate a file with SQL commands that, when fed back to the server, will recreate the database in the same state as it was at the time of the dump. PostgreSQL provides the utility program pg_dump for this purpose
The syntax is:

$ pg_dump databasename > output | backup_file

In our example,

$ pg_dump expenditure18 > expenditure18_bkp

a) SQL (pg_dump)

$ pg_dump databasename > output | backup_file

$ pg_dump –n schema_name Dbname > output | backup_file

In our example,

$ pg_dump expenditure18 > expenditure18_bkp

a) SQL (pg_dump)

$ pg_dump –n depense18 expenditure2018 > depense1820180525

Where –n is a parameter used to set the schema's name to backup.
To backup a table: Use --table to tell pg_dump what table it has to backup:

$ pg_dump –table schema.tablename dbname > filename

a) SQL (pg_dump)

$ pg_dump –-table depense18.users expenditure2018 > users20180525

Where –-table is a parameter used to set the table's name to backup.
To backup a table: Use --table to tell pg_dump what table it has to backup:

$ pg_dump --host localhost --port 5432 --username postgres --format plain --ignore-version --verbose --file "filename.backup" --table public.tablename dbname

NB: pg_dumpall: dump all db instance

Restauration

The text files created by pg_dump are intended to be read in by the psql program. The general command form to restore a dump is

$ psql dbname < infile

NB: before a restore, make sure to have a db in which to do the restore.

p.exp:

postgres# createbd –T template0 db_restore

postgres# db_restore < depense1820180425.dmp

Praticals with pgadmin during presentation (backup table, schema, database)

THANK YOU !