



# Universidad de la Sierra Sur

## Proyecto del Primer Parcial

### Nombres completo:

- Stephanie Jacqueline Lagunas González  
([lagunasgonzalezstephanie123@gmail.com](mailto:lagunasgonzalezstephanie123@gmail.com)).
- Jazziel Pérez Hernández  
([perezhernandezjazziel183@gmail.com](mailto:perezhernandezjazziel183@gmail.com)).

**Grupo:** 706.

**Materia:** Bases de Datos Distribuidas.

**Fecha de entrega:** 03 de noviembre del 2022.

# Instalación de PostgreSQL

La primera actividad que se realizó fue la instalación de PostgreSQL, cliente y la interfaz gráfica, esto se realizó mediante los pasos que se muestran a continuación.

## Paso 1

Crear la configuración del repositorio de archivos mediante la siguiente instrucción:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

```
labingsw08@labingsw08-All-Series:~$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
[sudo] contraseña para labingsw08:
labingsw08@labingsw08-All-Series:~$
```

## Paso 2

Importar la clave de firma del repositorio.

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

```
labingsw08@labingsw08-All-Series:~$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
labingsw08@labingsw08-All-Series:~$
```

## Paso 3

Actualizar las listas de paquetes.

```
sudo apt-get update
```

```

labingsw08@labingsw08-All-Series:~$ sudo apt-get update
Obj:1 http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease
Obj:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:3 https://dl.google.com/linux/chrome/deb stable InRelease
Obj:4 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Obj:5 http://download.virtualbox.org/virtualbox/debian jammy InRelease
Obj:6 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:7 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease
Obj:8 https://deb.nodesource.com/node_18.x jammy InRelease
Obj:9 https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/jammy pgadmin4 InRelease
Leyendo lista de paquetes... Hecho
W: http://apt.postgresql.org/pub/repos/apt/dists/jammy-pgdg/InRelease: Key is stored in
legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-ke
y(8) for details.
N: Omitiendo el uso del fichero configurado «main/binary-i386/Packages» ya que el reposi
torio «http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease» no admite la arquit
ectura «i386»
labingsw08@labingsw08-All-Series:~$

```

#### Paso 4:

Instalación de PostgreSQL versión 15.

**sudo apt-get -y install postgresql**

```

labingsw08@labingsw08-All-Series:~$ sudo apt-get -y install postgresql
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl pgdg-keyring postgresql-15 postgresql-client-15
  postgresql-client-common postgresql-common sysstat
Paquetes sugeridos:
  postgresql-doc postgresql-doc-15 isag
Se instalarán los siguientes paquetes NUEVOS:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl pgdg-keyring postgresql postgresql-15 postgresql-client-15
  postgresql-client-common postgresql-common sysstat
0 actualizados, 13 nuevos se instalarán, 0 para eliminar y 33 no actualizados.
Se necesita descargar 19.0 MB/43.7 MB de archivos.
Se utilizarán 173 MB de espacio de disco adicional después de esta operación.
Des:1 http://apt.postgresql.org/pub/repos/apt jammy-pgdg/main amd64 postgresql-client-co
mmon all 244.pgdg22.04+1 [92.0 kB]

```

#### Paso 5

Se revisó que se halla instalado correctamente.

**/usr/lib/postgresql/15/bin/postgres -version**

```
labingsw08@labingsw08-All-Series:~$ /usr/lib/postgresql/15/bin/postgres --version
postgres (PostgreSQL) 15.0 (Ubuntu 15.0-1.pgdg22.04+1)
labingsw08@labingsw08-All-Series:~$
```

## Paso 6

En este paso se realizó la instalación de la interfaz gráfica de PostgreSQL (pgAdmin 4).

Instalar la clave pública para el repositorio

```
curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --
dearmor -o /usr/share/keyrings/packages-pgadmin-org.gpg
```

```
labingsw08@labingsw08-All-Series:~$ curl -fsS https://www.pgadmin.org/static/pac
kages_pgadmin_org.pub | sudo gpg --dearmor -o /usr/share/keyrings/packages-pgadm
in-org.gpg
[sudo] contraseña para labingsw08:
El fichero '/usr/share/keyrings/packages-pgadmin-org.gpg' ya existe. ¿Sobreescri
bir? (s/N) s
labingsw08@labingsw08-All-Series:~$
```

## Paso 7

Crear el archivo de configuración del repositorio.

```
sudo sh -c 'echo "deb [signed-by=/usr/share/keyrings/packages-pgadmin-org.gpg]
https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/${lsb_release
```

```
labingsw08@labingsw08-All-Series:~$ sudo sh -c 'echo "deb [signed-by=/usr/share/
keyrings/packages-pgadmin-org.gpg] https://ftp.postgresql.org/pub/pgadmin/pgadmi
n4/apt/${lsb_release -cs} pgadmin4 main" > /etc/apt/sources.list.d/pgadmin4.list
&& apt update'
Des:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Obj:2 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Obj:3 https://dl.google.com/linux/chrome/deb stable InRelease
Des:4 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Des:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadat
a [20.0 kB]
Des:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Met
adata [13.3 kB]
Des:7 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Obj:8 https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/jammy pgadmin4 InRelea
```

## Paso 8

### Instalar pgAdmin.

`sudo apt install pgadmin4-desktop`

```
labingsw08@labingsw08-All-Series:~$ sudo apt install pgadmin4-desktop
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  pgadmin4-server
Se instalarán los siguientes paquetes NUEVOS:
  pgadmin4-desktop pgadmin4-server
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 33 no actualizados.
Se necesita descargar 196 MB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Después de la instalación, se realizó la configuración del usuario Postgres, mediante la siguiente instrucción.

`alter user postgres 'password';`

```
labingsw08@labingsw08-All-Series:~$ sudo -u postgres psql
[sudo] contraseña para labingsw08:
could not change directory to "/home/labingsw08": Permiso denegado
psql (15.0 (Ubuntu 15.0-1.pgdg22.04+1))
Type "help" for help.

postgres=# alter user postgres password 'fany';
ALTER ROLE
postgres=#
```

**Nota:** Estos 5 pasos se realizaron en las tres máquinas.

Al culminar las actividades anteriores, procedimos a realizar la creación de la base de datos llamada **dd\_test**, en el servidor maestro(servidor París) con la instrucción:

```
create database dd_test;
```

```
labso21@labso21-All-Series:~$ psql -h localhost -U postgres
Contraseña para usuario postgres:
psql (15.0 (Ubuntu 15.0-1.pgdg22.04+1))
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: d
esactivado)
Digite «help» para obtener ayuda.

postgres=# create database dd_test;

CREATE DATABASE
postgres=#
```

Dentro de la base **dd\_test** se realizó la creación de la tabla **estudiante(student)** y la tabla **libro(book)**, con las siguientes instrucciones:

```
drop table if exists student cascade;
```

```
create table student(id serial primary key, name varchar);
```

```
drop table if exists book cascade;
```

```
create table book(id serial primary key, title varchar);
```

```
postgres=# \c dd_test
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado)
Ahora está conectado a la base de datos «dd_test» con el usuario «postgres».
dd_test=# drop table if exists student cascade;
create table student(id serial primary key, name varchar);
NOTICE: la tabla «student» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=# drop table if exists book cascade;
create table book(id serial primary key, title varchar);
NOTICE: la tabla «book» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=#
```

También se realizó la creación de las tablas ejemplares(**booksample**) y prestamos(**lend**), con las siguientes instrucciones:

```
drop table if exists booksample cascade;
```

```
create table booksample(id serial primary key, state varchar, lendable varchar, location varchar,  
book_id int references book(id));
```

```
dd_test=# drop table if exists booksample cascade;  
create table booksample(id serial primary key, state varchar, lendable varchar, location va  
rchar, book_id int references book(id));  
NOTICE: la tabla «booksample» no existe, omitiendo  
DROP TABLE  
CREATE TABLE  
dd_test=#
```

```
drop table if exists lend cascade;
```

```
create table lend(student_id int references student(id), booksample_id int references  
booksample(id), at date, returned_at date);
```

```
dd_test=# drop table if exists lend cascade;  
create table lend(student_id int references student(id), booksample_id int references books  
ample(id), at date, returned_at date);  
NOTICE: la tabla «lend» no existe, omitiendo  
DROP TABLE  
CREATE TABLE  
dd_test=#
```

En el servidor de parís se crear las tablas ejemplares\_parís (**booksample\_paris**) y prestamos\_paris (**lend\_paris**) con las instrucciones:

```
drop table if exists booksample_paris cascade;
```

```
create table booksample_paris(check(location='paris')) inherits(booksample);
```

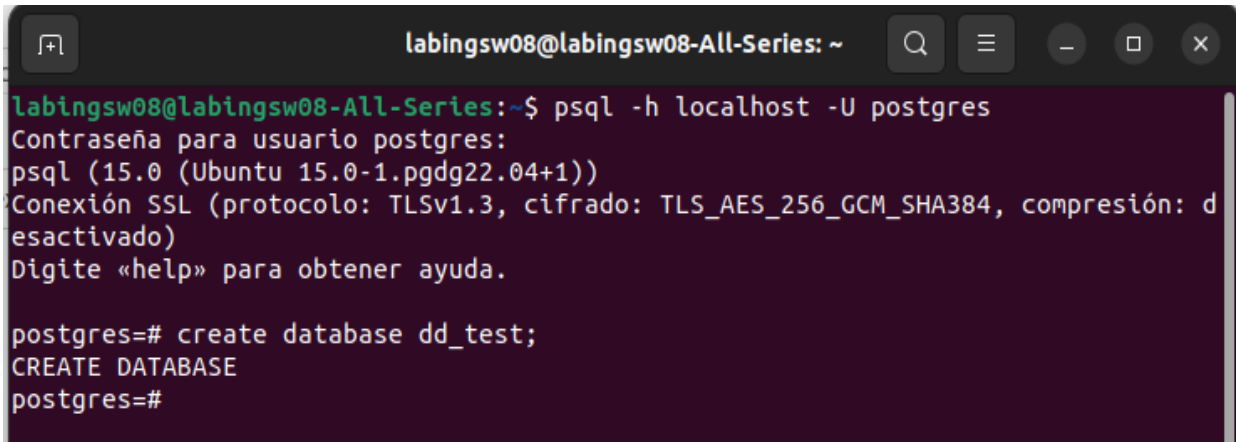
```
drop table if exists lend_paris cascade;
```

```
create table if not exists lend_paris() inherits(lend);
```

```
dd_test=# drop table if exists booksample_paris cascade;  
create table booksample_paris(check(location='paris')) inherits(booksample);  
NOTICE: la tabla «booksample_paris» no existe, omitiendo  
DROP TABLE  
CREATE TABLE  
dd_test=# drop table if exists lend_paris cascade;  
create table if not exists lend_paris() inherits(lend);  
NOTICE: la tabla «lend_paris» no existe, omitiendo  
DROP TABLE  
CREATE TABLE  
dd_test=#
```

Después nos ubicamos en el servidor lagos y realizamos la creación de la base de datos **dd\_test** con el siguiente comando:

```
create database dd_test;
```

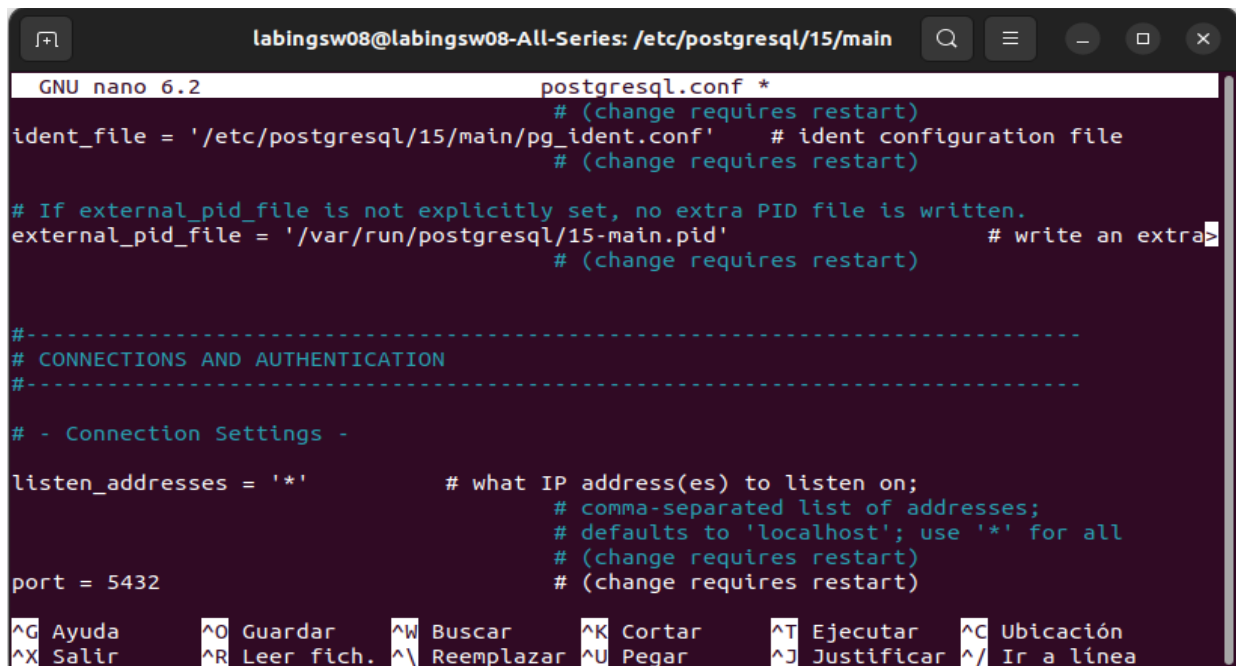


```
labingsw08@labingsw08-All-Series: ~  
labingsw08@labingsw08-All-Series:~$ psql -h localhost -U postgres  
Contraseña para usuario postgres:  
psql (15.0 (Ubuntu 15.0-1.pgdg22.04+1))  
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado)  
Digite «help» para obtener ayuda.  
  
postgres=# create database dd_test;  
CREATE DATABASE  
postgres=#
```

También se realizó la configuración para que el servidor Postgres pueda escuchar las otras interfaces de red, esto se hizo modificando el archivo `postgresql.conf`, el cual se encuentra en la ruta `/etc/postgresql/15/main/postgresql.conf`, en el cual se habilita la siguiente línea:

```
listen_addresses = '*'
```

y de después se guardan los cambios con `Ctrl + O`.



```
labingsw08@labingsw08-All-Series: /etc/postgresql/15/main  
GNU nano 6.2 postgresql.conf *  
# (change requires restart)  
ident_file = '/etc/postgresql/15/main/pg_ident.conf' # ident configuration file  
# (change requires restart)  
  
# If external_pid_file is not explicitly set, no extra PID file is written.  
external_pid_file = '/var/run/postgresql/15-main.pid' # write an extra  
# (change requires restart)  
  
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
  
# - Connection Settings -  
  
listen_addresses = '*' # what IP address(es) to listen on;  
# comma-separated list of addresses;  
# defaults to 'localhost'; use '*' for all  
# (change requires restart)  
port = 5432 # (change requires restart)  
  
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación  
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^/_ Ir a línea
```



De igual manera se configura el archivo `pg_hba.conf`, para darle permisos a los usuarios que se conecten a través de las interfaces de red. Esto se logra modificando la configuración de IPv4 de la siguiente manera:

```
#TYPE DATABASE USER ADDRESS METHOD
```

```
#IPv4 local connections:
```

```
host all postgres all md5
```

De igual manera se guardan los cambios con `Ctrl + O`.

```
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all postgres all md5
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
```

<sup>^G</sup> Ayuda    <sup>^O</sup> Guardar    <sup>^W</sup> Buscar    <sup>^K</sup> Cortar    <sup>^T</sup> Ejecutar    <sup>^C</sup> Ubicación  
<sup>^X</sup> Salir    <sup>^R</sup> Leer fich.    <sup>^\_\</sup> Reemplazar    <sup>^U</sup> Pegar    <sup>^J</sup> Justificar    <sup>^/</sup> Ir a línea

También se crean las tablas de particiones las cuales son `ejemplares_lagos` (**booksample\_lagos**) y `prestamos_lagos` (**lend\_lagos**) con las instrucciones:

```
drop table if exists booksample_lagos cascade;
create table booksample_lagos(id int, state varchar, lendable bool default false, location varchar,
book_id int);
```

```
drop table if exists lend_lagos cascade;
create table lend_lagos(student_id int, booksample_id int, at date, returned_at date);
```

```
postgres=# \c dd_test
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado)
Ahora está conectado a la base de datos «dd_test» con el usuario «postgres».
dd_test=# drop table if exists booksample_lagos cascade;
create table booksample_lagos(id int, state varchar, lendable bool default false, location varchar, book_id int);
NOTICE: la tabla «booksample_lagos» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=# drop table if exists lend_lagos cascade;
create table lend_lagos(student_id int, booksample_id int, at date, returned_at date);
NOTICE: la tabla «lend_lagos» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=#
```

Terminada la configuración en el servidor lagos, nos vamos al servidor mexico, en cual también se creara de la base de datos **dd\_test** con el siguiente comando:

**create database dd\_test;**

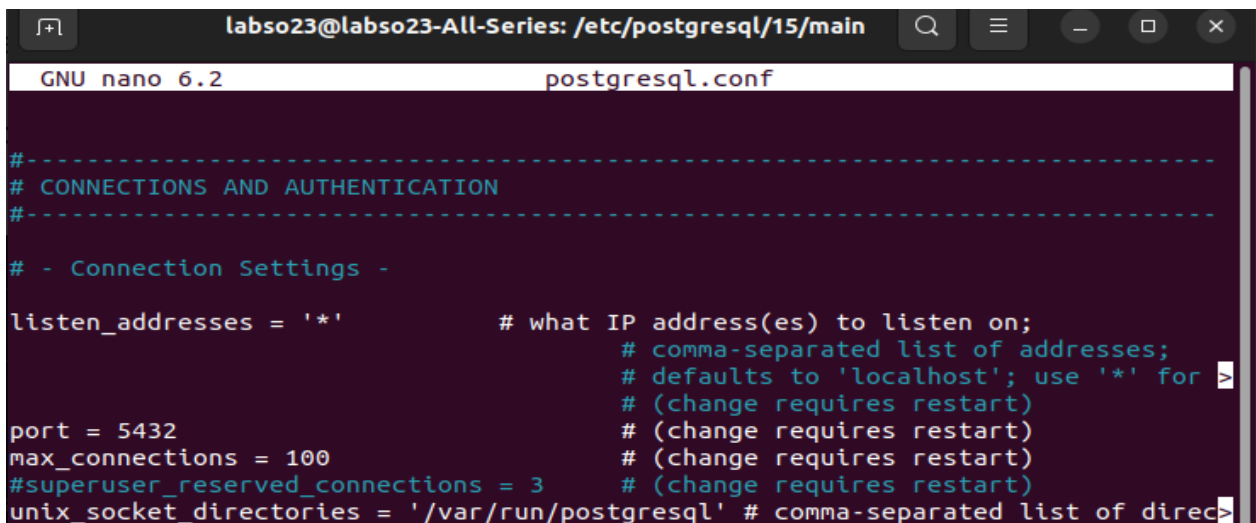
```
labso23@labso23-All-Series:~$ psql -h localhost -U postgres
Contraseña para usuario postgres:
psql (15.0 (Ubuntu 15.0-1.pgdg22.04+1))
Conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, compresión: desactivado)
Digite «help» para obtener ayuda.

postgres=# create database dd_test;
CREATE DATABASE
postgres=#
```

Así como en el servidor lagos se realizo la configuración para que el servidor postgres pueda escuchar las otras interfaces de red, esto se hizo modificando el archivo postgresql.conf, de igual manera se realiza este paso en el servidor México mediante la habilitación de la siguiente línea:

**listen\_addresses = '\*'**

y de después se guardan los cambios con Ctrl + O.



```
labso23@labso23-All-Series: /etc/postgresql/15/main
GNU nano 6.2 postgresql.conf

#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                     # (change requires restart)
max_connections = 100           # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
```

También se configura el archivo `pg_hba.conf`, para darle permisos a los usuarios que se conecten a través de las interfaces de red, modificando la configuración de IPv4:

#TYPE	DATABASE	USER	ADDRESS	METHOD
-------	----------	------	---------	--------

#IPv4 local connections:

host	all	postgres	all	md5
------	-----	----------	-----	-----

y se guardan los cambios con Ctrl + O.

```
# "local" is for Unix domain socket connections only
local    all             all                                     peer
# IPv4 local connections:
host     all             postgres                               md5
# IPv6 local connections:
host     all             all                                     ::1/128         scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                                     peer
host     replication     all                                     127.0.0.1/32    scram-sha-256
host     replication     all                                     ::1/128         scram-sha-256
█

^G Ayuda      ^O Guardar    ^W Buscar     ^K Cortar     ^T Ejecutar   ^C Ubicación
^X Salir      ^R Leer fich.^_ Reemplazar  ^U Pegar      ^J Justificar ^_ Ir a línea
```

Por ultimo se crean las tablas de particiones las cuales son ejemplares\_mexico (**booksample\_mexico**) y prestamos\_mexico (**lend\_mexico**) con las instrucciones:

**drop table if exists booksample\_mexico cascade;**

**create table booksample\_mexico(id int, state varchar, lendable bool default false, location varchar, book\_id int);**

**drop table if exists lend\_mexico cascade;**

**create table lend\_mexico(student\_id int, booksample\_id int, at date, returned\_at date);**

```
dd_test=# drop table if exists booksample_mexico cascade;
create table booksample_mexico(id int, state varchar, lendable bool default false, location varchar, book_id int);
NOTICE: la tabla «booksample_mexico» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=# drop table if exists lend_mexico cascade;
create table lend_mexico(student_id int, booksample_id int, at date, returned_at date);
NOTICE: la tabla «lend_mexico» no existe, omitiendo
DROP TABLE
CREATE TABLE
dd_test=# █
```

Nos ubicamos nuevamente en el servidor máster (servidor París), en donde crearemos una extensión llamada **postgres\_fdw** mediante la siguiente instrucción:

```
create extension postgres_fdw;
```

```
dd_test=# create extension postgres_fdw;  
CREATE EXTENSION  
dd_test=#
```

Después se crea un "**servidor de datos extranjeros**", al igual que se realiza un **mapeo de usuarios** para poder consultar en el servidor de Lagos, al igual que una actualización y esto se realiza mediante las siguientes instrucciones:

```
create server master_server foreign data wrapper postgres_fdw options (host '132.18.53.24', port '5432', dbname 'dd_test');
```

```
dd_test=# create server master_server foreign data wrapper postgres_fdw options  
(host '132.18.53.24', port '5432', dbname 'dd_test');  
CREATE SERVER  
dd_test=#
```

```
create user mapping for postgres server master_server options (user 'postgres', password 'fany');
```

```
dd_test=# create user mapping for postgres server master_server options (user 'p  
ostgres', password 'fany');  
CREATE USER MAPPING  
dd_test=#
```

```
alter server master_server owner to postgres;
```

```
dd_test=# alter server master_server owner to postgres;  
ALTER SERVER  
dd_test=#
```

Después de eso se realiza la creación de las tablas foráneas ejemplares\_lagos (**booksample\_lagos**) y prestamos\_lagos (**lend\_lagos**) de la siguiente manera..

```
drop foreign table if exists booksample_lagos cascade;  
create foreign table booksample_lagos (check(location='lagos')) inherits(booksample) server  
master_server;
```

```
drop foreign table if exists lend_lagos cascade;  
create foreign table lend_lagos () inherits(lend) server master_server;
```

```
dd_test=# drop foreign table if exists booksample_lagos cascade;
create foreign table booksample_lagos (check(location='lagos')) inherits(booksam
ple) server master_server;
NOTICE: la tabla foránea «booksample_lagos» no existe, omitiendo
DROP FOREIGN TABLE
CREATE FOREIGN TABLE
dd_test=# drop foreign table if exists lend_lagos cascade;
create foreign table lend_lagos () inherits(lend) server master_server;
NOTICE: la tabla foránea «lend_lagos» no existe, omitiendo
DROP FOREIGN TABLE
CREATE FOREIGN TABLE
dd_test=#
```

De igual manera se crea un "servidor de datos extranjeros", y se realiza un **mapeo de usuarios** para poder consultar en el servidor de mexico, al igual que una actualización, mediante las siguientes instrucciones:

```
create server master_server1 foreign data wrapper postgres_fdw options (host '132.18.53.23',
port '5432' , dbname 'dd_test');
```

```
dd_test=# create server master_server1 foreign data wrapper postgres_fdw options
(host '132.18.53.23', port '5432' , dbname 'dd_test');
CREATE SERVER
dd_test=#
```

```
create user mapping for postgres server master_server1 options (user 'postgres', password
'mexico');
```

```
dd_test=# create user mapping for postgres server master_server1 options
(user 'postgres', password 'mexico');
CREATE USER MAPPING
dd_test=#
```

```
alter server master_server1 owner to postgres;
```

```
CREATE USER MAPPING
dd_test=# alter server master_server1 owner to postgres;
ALTER SERVER
dd_test=#
```

Después de eso se realiza la creación de las tablas foráneas ejemplares\_mexico (**booksample\_mexico**) y prestamos\_mexico (**lend\_mexico**) de la siguiente manera.

```
drop foreign table if exists booksample_mexico cascade;
create foreign table booksample_mexico (check(location='mexico')) inherits(booksample) server
master_server1;
```

```
drop foreign table if exists lend_mexico cascade;
create foreign table lend_mexico () inherits(lend) server master_server1;
```

```
dd_test=# drop foreign table if exists booksample_mexico cascade;
create foreign table booksample_mexico (check(location='mexico'))
inherits(booksample) server master_server1;
NOTICE: la tabla foránea «booksample_mexico» no existe, omitiendo
DROP FOREIGN TABLE
CREATE FOREIGN TABLE
dd_test=# drop foreign table if exists lend_mexico cascade;
create foreign table lend_mexico () inherits(lend) server master_server1;
NOTICE: la tabla foránea «lend_mexico» no existe, omitiendo
DROP FOREIGN TABLE
CREATE FOREIGN TABLE
dd_test=#
```

Después de eso se realizaron los disparadores(triggers) para la inserción de datos en las tablas ejemplares(**booksample**) y prestamos (**lend**) , dichos disparadores insertaran los datos correctamente según su localización, a continuación se muestran las instrucciones.

**create or replace function booksample\_trigger\_fn() returns trigger as**

```
$$
begin
    if new.location = 'paris' then
        insert into booksample_paris values(new.*);
    elsif new.location = 'lagos' then
        insert into booksample_lagos values(new.*);
    elsif new.location = 'mexico' then
        insert into booksample_mexico values(new.*);
    end if;
    return null;
end
$$
language plpgsql;
```

**drop trigger if exists booksample\_trigger on booksample;**

**create trigger booksample\_trigger before insert on booksample for each row execute procedure booksample\_trigger\_fn();**

```
dd_test=# create or replace function booksample_trigger_fn() returns trigger as
$$
begin
    if new.location = 'paris' then
        insert into booksample_paris values(new.*);
    elsif new.location = 'lagos' then
        insert into booksample_lagos values(new.*);
    elsif new.location = 'mexico' then
        insert into booksample_mexico values(new.*);
    end if;
    return null;
end
$$
language plpgsql;

drop trigger if exists booksample_trigger on booksample;
create trigger booksample_trigger before insert on booksample for each row execute procedure booksample_trigger_fn();
CREATE FUNCTION
DROP TRIGGER
CREATE TRIGGER
dd_test=#
```

```

create or replace function lend_trigger_fn() returns trigger as
$$
declare
    vbooksample booksample%rowtype;
begin
    select * into vbooksample from booksample where id=new.booksample_id;

    if vbooksample.location = 'paris' then
        insert into lend_paris values(new.*);
    elsif vbooksample.location = 'lagos' then
        insert into lend_lagos values(new.*);
    elsif vbooksample.location = 'mexico' then
        insert into lend_mexico values(new.*);
    end if;

    return null;
end
$$
language plpgsql;

drop trigger if exists lend_trigger on lend;
create trigger lend_trigger before insert on lend for each row execute procedure
lend_trigger_fn();

```

```

dd_test=# create or replace function lend_trigger_fn() returns trigger as
$$
declare
    vbooksample booksample%rowtype;
begin
    -- select the booksample referenced by the booksample_id
    select * into vbooksample from booksample where id=new.booksample_id;

    -- get the location to use and save the row
    if vbooksample.location = 'paris' then
        insert into lend_paris values(new.*);
    elsif vbooksample.location = 'lagos' then
        insert into lend_lagos values(new.*);
    elsif vbooksample.location = 'mexico' then
        insert into lend_mexico values(new.*);
    end if;

    return null;
end
$$
language plpgsql;

drop trigger if exists lend_trigger on lend;
create trigger lend_trigger before insert on lend for each row execute procedure lend_trigger_fn();
CREATE FUNCTION
DROP TRIGGER
CREATE TRIGGER
dd_test=# █

```

La siguiente actividad que se realizo fue la inserción de datos a las tablas libros(book), estudiantes(students), ejemplares(**booksample**) y prestamos (**lend**), a continuación se muestra el scrip de inserción:

```
insert into student values(default,'Juan'),(default, 'Maria'),(default,'Pedro'),(default,'Marta'),
(default,'Jaime'),(default,'Priscila'),(default,'Jorge'),(default,'Elena'),(default,'Luis'),
(default,'Laura');
```

```
dd_test=# insert into student values(default,'Juan'),(default, 'Maria'),(default,'Pedro'),(default,'Marta'),(
default,'Jaime'),(default,'Priscila'),(default,'Jorge'),(default,'Elena'),(default,'Luis'),(default,'Laura');
INSERT 0 10
dd_test=#
```

```
insert into book values(default,' Romeo y Julieta'),(default, 'Don Quijote de la Mancha'),
(default,'Cien años de soledad'),(default,'El laberinto de la soledad'),(default,'De animales a
dioses'),(default,'Vida inteligente en el universo'),(default,'Las trampas de la fe'),(default,'La
evolución de las especies'),(default,'Principia Mathematica'), (default,'Crítica de la razón pura');
```

```
dd_test=# insert into book values(default,' Romeo y Julieta'),(default, 'Don Quijote de la Mancha'),
(default,'Cien años de soledad'),(default,'El laberinto de la soledad'),(default,'De animales a dios
es'),(default,'Vida inteligente en el universo'),(default,'Las trampas de la fe'),(default,'La evolu
ción de las especies'),(default,'Principia Mathematica'), (default,'Crítica de la razón pura');

INSERT 0 10
dd_test=#
```

```
insert into booksample values (1,'new','1','paris',3),(2,'used','1','lagos',4),(3,'used','0','paris',1),
(4,'used','0','lagos',2),(5,'new','1','paris',5),(6,'new','1','lagos',1),(7,'used','0','paris',5),
(8,'used','1','lagos',7), (9,'used','1','paris',8),(10,'used','0','lagos',10), (11,'new','1','mexico',4),
(12,'used','0','mexico',7), (13,'new','0','mexico',5),(14,'used','1','mexico',4),
(15,'new','0','mexico',6);
```

```
dd_test=# insert into booksample values (1,'new','1','paris',3),(2,'used','1','lagos',4),(3,'u
sed','0','paris',1),(4,'used','0','lagos',2),(5,'new','1','paris',5),(6,'new','1','lagos',1),(
7,'used','0','paris',5),(8,'used','1','lagos',7),(9,'used','1','paris',8),(10,'used','0','lago
s',10),(11,'new','1','mexico',4),(12,'used','0','mexico',7),(13,'new','0','mexico',5),(14,'use
d','1','mexico',4),(15,'new','0','mexico',6);
INSERT 0 0
```



insert into lend values

```
(1,1,'14/02/20','02/03/20'),(2,1,'17/05/21','27/05/21'),(3,1,'12/05/20','22/05/20'),  
(4,2,'27/10/22','13/11/22'),(5,2,'17/06/19','03/07/19'),(6,3,'23/08/22','06/09/22'),  
(7,3,'11/11/21','24/11/21'),(8,4,'27/09/21','08/10/21'),(9,4,'02/06/21','17/06/21'),  
(10,4,'31/03/21','07/04/21'),(1,4,'25/12/20','05/01/21'),(2,5,'20/09/22','01/10/22'),  
(3,5,'11/01/19','30/01/19'),(4,6,'06/06/21','26/06/21'),(5,6,'12/10/20','02/11/20'),  
(6,7,'18/05/20','08/06/20'),(7,7,'03/09/20','11/09/20'),(8,8,'08/12/20','20/12/20'),  
(9,8,'27/11/22','14/12/22'),(10,9,'02/09/21','11/09/21'),(9,9,'07/10/20','18/10/20'),  
(4,10,'02/08/19','24/08/19'),(2,10,'07/11/20','30/11/20'),(7,10,'11/03/22','31/03/22'),  
(9,10,'16/04/22','07/05/22'),(3,10,'09/05/22','19/05/22'),(2,10,'19/10/20','06/11/20'),  
(2,11,'01/06/19','09/06/19'),(9,11,'30/09/20','08/10/20'),(2,12,'09/04/22','11/04/22'),  
(3,12,'18/04/22','28/04/22'),(7,13,'16/06/19','24/06/19'),(4,13,'05/12/22','17/12/22'),  
(6,13,'27/04/20','28/04/20'),(6,13,'22/03/19','04/04/19'),(4,14,'22/02/19','23/02/19'),  
(8,14,'01/11/19','09/11/19'),(7,15,'07/02/19','13/02/19'),(9,15,'10/06/22','15/06/22');
```

```
dd_test=# insert into lend values  
(1,1,'14/02/20','02/03/20'),  
(2,1,'17/05/21','27/05/21'),  
(3,1,'12/05/20','22/05/20'),  
(4,2,'27/10/22','13/11/22'),  
(5,2,'17/06/19','03/07/19'),  
(6,3,'23/08/22','06/09/22'),  
(7,3,'11/11/21','24/11/21'),  
(8,4,'27/09/21','08/10/21'),  
(9,4,'02/06/21','17/06/21'),  
(10,4,'31/03/21','07/04/21'),  
(1,4,'25/12/20','05/01/21'),  
(2,5,'20/09/22','01/10/22'),  
(3,5,'11/01/19','30/01/19'),  
(4,6,'06/06/21','26/06/21'),  
(5,6,'12/10/20','02/11/20'),  
(6,7,'18/05/20','08/06/20'),  
(7,7,'03/09/20','11/09/20'),  
(8,8,'08/12/20','20/12/20'),  
(9,8,'27/11/22','14/12/22'),  
(10,9,'02/09/21','11/09/21'),  
(9,9,'07/10/20','18/10/20'),  
(4,10,'02/08/19','24/08/19'),  
(2,10,'07/11/20','30/11/20'),  
(7,10,'11/03/22','31/03/22'),  
(9,10,'16/04/22','07/05/22'),  
(3,10,'09/05/22','19/05/22'),  
(2,10,'19/10/20','06/11/20'),  
(1,11,'26/06/20','08/07/20'),  
(2,11,'01/06/19','09/06/19'),  
(9,11,'30/09/20','08/10/20'),  
(2,12,'09/04/22','11/04/22'),  
(3,12,'18/04/22','28/04/22'),  
(7,13,'16/06/19','24/06/19'),  
(4,13,'05/12/22','17/12/22'),  
(6,13,'27/04/20','28/04/20'),  
(6,13,'22/03/19','04/04/19'),  
(4,14,'22/02/19','23/02/19'),  
(8,14,'01/11/19','09/11/19'),  
(7,15,'07/02/19','13/02/19'),  
(9,15,'10/06/22','15/06/22');  
INSERT 0 0  
dd_test=#
```

Por último se hizo la verificación de que los datos se hallan insertado en los nodos correspondientes.

```
dd_test=# select * from booksample_paris;
 id | state | lendable | location | book_id
-----+-----+-----+-----+-----
  1 | new   | 1        | paris    |      3
  3 | used  | 0        | paris    |      1
  5 | new   | 1        | paris    |      5
  7 | used  | 0        | paris    |      5
  9 | used  | 1        | paris    |      8
(5 filas)
```

```
dd_test=# select * from lend_paris;
 student_id | booksample_id |      at      | returned_at
-----+-----+-----+-----
          1 |              1 | 2020-02-14 | 2020-03-02
          2 |              1 | 2021-05-17 | 2021-05-27
          3 |              1 | 2020-05-12 | 2020-05-22
          6 |              3 | 2022-08-23 | 2022-09-06
          7 |              3 | 2021-11-11 | 2021-11-24
          2 |              5 | 2022-09-20 | 2022-10-01
          3 |              5 | 2019-01-11 | 2019-01-30
          6 |              7 | 2020-05-18 | 2020-06-08
          7 |              7 | 2020-09-03 | 2020-09-11
         10 |              9 | 2021-09-02 | 2021-09-11
          8 |              9 | 2020-10-07 | 2020-10-18
(11 filas)
```

```
dd_test=# select * from booksample_lagos;
 id | state | lendable | location | book_id
-----+-----+-----+-----+-----
   1 | used  | f        | lagos    |      2
   2 | used  | t        | lagos    |      4
   4 | used  | f        | lagos    |      2
   6 | new   | t        | lagos    |      1
   8 | used  | t        | lagos    |      7
  10 | used  | f        | lagos    |     10
(6 filas)
```

```
dd_test=#
```

```
dd_test=# select * from lend_lagos;
 student_id | booksample_id |      at      | returned_at
-----+-----+-----+-----
          4 |              2 | 2022-10-27 | 2022-11-13
          5 |              2 | 2019-06-17 | 2019-07-03
          8 |              4 | 2021-09-27 | 2021-10-08
          9 |              4 | 2021-06-02 | 2021-06-17
         10 |              4 | 2021-03-31 | 2021-04-07
          1 |              4 | 2020-12-25 | 2021-01-05
          4 |              6 | 2021-06-06 | 2021-06-26
          5 |              6 | 2020-10-12 | 2020-11-02
          8 |              8 | 2020-12-08 | 2020-12-20
          9 |              8 | 2022-11-27 | 2022-12-14
          4 |             10 | 2019-08-02 | 2019-08-24
          2 |             10 | 2020-11-07 | 2020-11-30
          7 |             10 | 2022-03-11 | 2022-03-31
          9 |             10 | 2022-04-16 | 2022-05-07
          3 |             10 | 2022-05-09 | 2022-05-19
          2 |             10 | 2020-10-19 | 2020-11-06
(16 filas)
```

```
dd_test=# select *from booksample_mexico;
 id | state | lendable | location | book_id
-----+-----+-----+-----+-----
   | used  | f        | lagos    |      2
  11 | new   | t        | mexico   |      4
  12 | used  | f        | mexico   |      7
  13 | new   | f        | mexico   |      5
  14 | used  | t        | mexico   |      4
  15 | new   | f        | mexico   |      6
(6 filas)
```

```
dd_test=# select *from lend_mexico;
 student_id | booksample_id |      at      | returned_at
-----+-----+-----+-----
          2 |             11 | 2019-06-01 | 2019-06-09
          9 |             11 | 2020-09-30 | 2020-10-08
          2 |             12 | 2022-04-09 | 2022-04-11
          3 |             12 | 2022-04-18 | 2022-04-28
          7 |             13 | 2019-06-16 | 2019-06-24
          4 |             13 | 2022-12-05 | 2022-12-17
          6 |             13 | 2020-04-27 | 2020-04-28
          6 |             13 | 2019-03-22 | 2019-04-04
          4 |             14 | 2019-02-22 | 2019-02-23
          8 |             14 | 2019-11-01 | 2019-11-09
          7 |             15 | 2019-02-07 | 2019-02-13
          9 |             15 | 2022-06-10 | 2022-06-15
(12 filas)
```