

Primero instalaremos postgres en Linux

```
ilse@ilse-HP-Laptop-15-bs0xx:~$ sudo apt-get install postgresql postgresql-client postgresql-contrib
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
libflashrom1 libftdi1-2 libllvm13
```

Verificamos versión y entramos a postgres

```
ilse@ilse-HP-Laptop-15-bs0xx:~$ SELECT version();
bash: error sintáctico cerca del elemento inesperado `('
ilse@ilse-HP-Laptop-15-bs0xx:~$ sudo su postgres
su: user postgres does not exist or the user entry does not contain all the required fields
ilse@ilse-HP-Laptop-15-bs0xx:~$ sudo su postgres
postgres@ilse-HP-Laptop-15-bs0xx:/home/ilse$ cd
postgres@ilse-HP-Laptop-15-bs0xx:~$ psql
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# SELECT version();
               version
-----
PostgreSQL 14.8 (Ubuntu 14.8-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.3.0-1ubuntu1) 11.3.0
(1 row)

postgres=#
```

Partición lógica

- ❖ Creamos una base de datos desde cero
- ❖ Posteriormente entramos a la base de datos con \c y el nombre de la base de datos (\c partition_logic)
- ❖ Creamos una tabla la cual será nuestra tabla primaria o principal, dentro de esta misma tabla le diremos de que modo será nuestra partición en este caso será lógica y de un rango específico, de este modo se le coloca PARTITION BY RANGE
- ❖ Creamos nuestra tabla secundaria numero 1 la cual le diremos que haga la partición de la tabla primaria la cual se llama my_partitioned_table y como le digimos que iba a ser particionada por rangos ahí mismo le asignamos de que rango a que rango será la partición (FOR VALUES FROM ('2022-01-01') TO ('2022-06-30')) le estamos diciendo que la partición la ara apartir del rango del mes de enero al mes de junio
- ❖ Posteriormente creamos nuestra segunda tabla secundaria y se repite el mismo procedimiento solo que aquí cambiamos los valores del rango de decimos que será del mes de julio al mes de diciembre
- ❖ Y \dt nos muestra las tablas que creamos y que están dentro de la base de datos
- ❖

```

postgres=# create database partition_logic;
CREATE DATABASE
postgres=# \c partition_logic
You are now connected to database "partition_logic" as user "postgres".
partition_logic=# CREATE TABLE my_partitioned_table (
id INT,
name TEXT,
created_at TIMESTAMP
) PARTITION BY RANGE (created_at);
CREATE TABLE
partition_logic=# CREATE TABLE my_partitioned_table_partition_1 PARTITION OF my_partitioned_table FOR
VALUES FROM ('2022-01-01') TO ('2022-06-30');
ERROR:  relation "my_partitioned_table" does not exist
partition_logic=# CREATE TABLE my_partitioned_table_partition_1 PARTITION OF
my_partitioned_table
FOR VALUES FROM ('2022-01-01') TO ('2022-06-30');
CREATE TABLE
partition_logic=# CREATE TABLE my_partitioned_table_partition_2 PARTITION OF
my_partitioned_table
FOR VALUES FROM ('2022-07-01') TO ('2022-12-31');
CREATE TABLE
partition_logic=# \dt

```

Schema	Name	Type	Owner
public	my_partitioned_table	partitioned table	postgres
public	my_partitioned_table_partition_1	table	postgres
public	my_partitioned_table_partition_2	table	postgres

```

(3 rows)

partition_logic=#

```

- ❖ Creamos los Alter para comprobar que si se hizo correctamente la partición de acuerdo a los rangos que le pusimos.
- ❖ Posteriormente insertamos dos registros a la tabla primaria
- ❖ Automáticamente el manejador redirecciona las inserciones a la tabla secundaria a la que pertenece por el rango
- ❖ Comprobamos que sea correcto y que nuestras tablas tengan los datos

```

partition_logic=# ALTER TABLE my_partitioned_table_partition_1 ADD CONSTRAINT partition_1_check CHECK (created_at >= '2022-01-01' AND created_at <= '2022-06-30');
ALTER TABLE
partition_logic=# ALTER TABLE my_partitioned_table_partition_2 ADD CONSTRAINT partition_2_check CHECK (created_at >= '2022-07-01' AND created_at <= '2022-12-31');
ALTER TABLE
partition_logic=# INSERT INTO my_partitioned_table (id, name, created_at)
VALUES (1, 'John', '2022-03-15');
INSERT 0 1
partition_logic=# INSERT INTO my_partitioned_table (id, name, created_at)
VALUES (2, 'Jane', '2022-09-01');
ERROR:  syntax error at or near "INSERT"
LINE 1: INSERT INTO my_partitioned_table (id, name, created_at)
        ^
partition_logic=# INSERT INTO my_partitioned_table (id, name, created_at)
VALUES (2, 'Jane', '2022-09-01');
INSERT 0 1
partition_logic=# select * from my_partitioned_table;
 id | name |      created_at
-----+-----+-----
  1 | John | 2022-03-15 00:00:00
  2 | Jane | 2022-09-01 00:00:00
(2 rows)

partition_logic=# select * from my_partitioned_table_partition_1;
 id | name |      created_at
-----+-----+-----
  1 | John | 2022-03-15 00:00:00
(1 row)

partition_logic=# select * from my_partitioned_table_partition_2;
 id | name |      created_at
-----+-----+-----
  2 | Jane | 2022-09-01 00:00:00
(1 row)

partition_logic=#

```

Partición física

- ❖ Creamos una nueva base de datos
- ❖ Creamos nuestra tabla primaria que se llamara sales
- ❖ Creamos nuestras tablas secundarias lo cual la palabra INHERITS significa heredar lo cual va a heredar de la tabla principal que en este caso es sales de acuerdo al campo org.
- ❖ Visualizamos que nuestras tablas se hallan creado correctamente

```
postgres=# create database particion_fisica;
CREATE DATABASE
postgres=# \c particion_fisica
You are now connected to database "particion_fisica" as user "postgres".
particion_fisica=# create table sales(org int,name varchar(10));
CREATE TABLE
particion_fisica=# create table sales_part1 (CHECK (org < 6)) INHERITS (sales);
CREATE TABLE
particion_fisica=# create table sales_part2 (CHECK (org >=6 and org <=10)) INHERITS (sales);
CREATE TABLE
particion_fisica=# \dt
      List of relations
Schema |      Name      | Type | Owner
-----+-----+-----+-----
public | sales           | table | postgres
public | sales_part1     | table | postgres
public | sales_part2     | table | postgres
(3 rows)

particion_fisica=#
```

- ❖ Creamos un trigger para la tabla primaria para que redireccione
- ❖ Insertamos los registros en la tabla primaria y ella se encargara de redireccionarlos a la tabla secundaria
- ❖ Y visualizamos que se haya asignado a la tabla correspondiente

```

partition_fisica=# create or replace rule insert_seles_p1 AS ON INSERT TO sales where (org <6) do instead INSERT INTO sales_part1 values (new.
org,new.name);
CREATE RULE
partition_fisica=# create or replace rule insert_seles_p2 AS ON INSERT TO sales where (org >=6 AND org <=10) do instead INSERT INTO sales_part
2 values (new.org,new.name);
CREATE RULE
partition_fisica=# insert into sales values (1,'Craig');INSERT 0 0
partition_fisica=# insert into sales values (2,'Mike');
INSERT 0 0
partition_fisica=# insert into sales values (3,'Michelle');
INSERT 0 0
partition_fisica=# insert into sales values (4,'Joe');
INSERT 0 0
partition_fisica=# insert into sales values (5,'Scott');
INSERT 0 0
partition_fisica=# insert into sales values (6,'Roger');
INSERT 0 0
partition_fisica=# insert into sales values (7,'Fred');
INSERT 0 0
partition_fisica=# insert into sales values (8,'Sam');
INSERT 0 0
partition_fisica=# select * from sales;
 org | name
-----+-----
  1  | Craig
  2  | Mike
  3  | Michelle
  4  | Joe
  5  | Scott
  6  | Roger
  7  | Fred
  8  | Sam
(8 rows)

partition_fisica=# 

```

```

particion_fisica=# insert into sales values (7,'Fred');
INSERT 0 0
particion_fisica=# insert into sales values (8,'Sam');
INSERT 0 0
particion_fisica=# select * from sales;
 org |  name
-----+-----
   1 | Craig
   2 | Mike
   3 | Michelle
   4 | Joe
   5 | Scott
   6 | Roger
   7 | Fred
   8 | Sam
(8 rows)

particion_fisica=# select * from sales_part1;
 org |  name
-----+-----
   1 | Craig
   2 | Mike
   3 | Michelle
   4 | Joe
   5 | Scott
(5 rows)

particion_fisica=# select * from sales_part2;
 org | name
-----+-----
   6 | Roger
   7 | Fred
   8 | Sam
(3 rows)

particion_fisica=# 

```

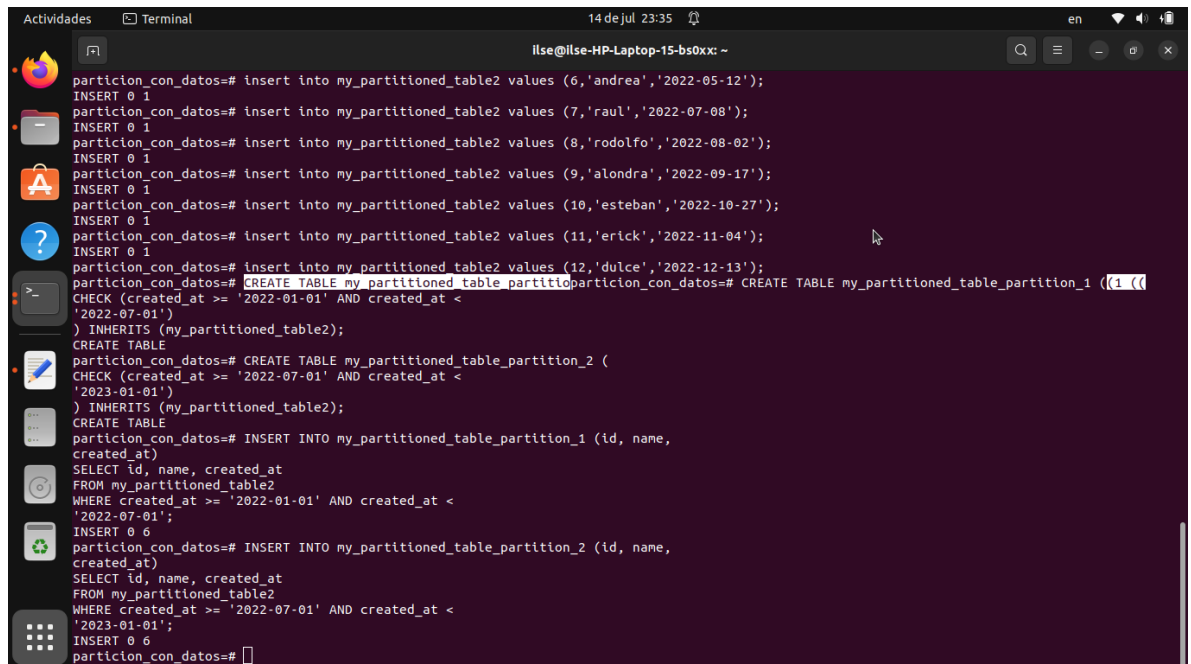
- ❖ Volvemos a crear una base de datos nueva
- ❖ Junto con sus tablas primaria y secundarias
- ❖ Hacemos registros
- ❖

```

partition_fisica=# select count (*) from sales;
count
-----
      8
(1 row)

partition_fisica=# \c postgres
You are now connected to database "postgres" as user "postgres".
postgres=# create database partition_con_datos;
CREATE DATABASE
postgres=# \c partition_con_datos
You are now connected to database "partition_con_datos" as user "postgres".
partition_con_datos=# CREATE TABLE my_partitioned_table2 (
id INT,
name TEXT,
created_at TIMESTAMP
);
CREATE TABLE
partition_con_datos=# insert into my_partitioned_table2 values (1,'luis','2022-06-30');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (2,'anahi','2022-01-05');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (3,'teresa','2022-02-07');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (4,'alexis','2022-03-14');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (5,'ximena','2022-04-22');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (6,'andrea','2022-05-12');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (7,'raul','2022-07-08');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (8,'rodolfo','2022-08-02');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (9,'alondra','2022-09-17');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (10,'esteban','2022-10-27');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (11,'erick','2022-11-04');
INSERT 0 1

```



```

Actividades Terminal 14 de jul 23:35 en
ilse@ilse-HP-Laptop-15-bs0xx: ~
partition_con_datos=# insert into my_partitioned_table2 values (6,'andrea','2022-05-12');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (7,'raul','2022-07-08');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (8,'rodolfo','2022-08-02');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (9,'alondra','2022-09-17');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (10,'esteban','2022-10-27');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (11,'erick','2022-11-04');
INSERT 0 1
partition_con_datos=# insert into my_partitioned_table2 values (12,'dulce','2022-12-13');
INSERT 0 1
partition_con_datos=# CREATE TABLE my_partitioned_table_partition_1 (
CHECK (created_at >= '2022-01-01' AND created_at <
'2022-07-01')
) INHERITS (my_partitioned_table2);
CREATE TABLE
partition_con_datos=# CREATE TABLE my_partitioned_table_partition_2 (
CHECK (created_at >= '2022-07-01' AND created_at <
'2023-01-01')
) INHERITS (my_partitioned_table2);
CREATE TABLE
partition_con_datos=# INSERT INTO my_partitioned_table_partition_1 (id, name,
created_at)
SELECT id, name, created_at
FROM my_partitioned_table2
WHERE created_at >= '2022-01-01' AND created_at <
'2022-07-01';
INSERT 0 6
partition_con_datos=# INSERT INTO my_partitioned_table_partition_2 (id, name,
created_at)
SELECT id, name, created_at
FROM my_partitioned_table2
WHERE created_at >= '2022-07-01' AND created_at <
'2023-01-01';
INSERT 0 6
partition_con_datos=#

```

```

FROM my_partitioned_table2
WHERE created_at >= '2022-07-01' AND created_at <
'2023-01-01';
INSERT 0 6
partition_con_datos=# \dt

```

```

List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | my_partitioned_table2 | table | postgres
public | my_partitioned_table_partition_1 | table | postgres
public | my_partitioned_table_partition_2 | table | postgres
(3 rows)

```

```

partition_con_datos=# select * from my_partitioned_table2;

```

```

id | name | created_at
---+-----+-----
1 | luis | 2022-06-30 00:00:00
2 | anahi | 2022-01-05 00:00:00
3 | teresa | 2022-02-07 00:00:00
4 | alexis | 2022-03-14 00:00:00
5 | ximena | 2022-04-22 00:00:00
6 | andrea | 2022-05-12 00:00:00
7 | raul | 2022-07-08 00:00:00
8 | rodolfo | 2022-08-02 00:00:00
9 | alondra | 2022-09-17 00:00:00
10 | esteban | 2022-10-27 00:00:00
11 | erick | 2022-11-04 00:00:00
12 | dulce | 2022-12-13 00:00:00
1 | luis | 2022-06-30 00:00:00
2 | anahi | 2022-01-05 00:00:00
3 | teresa | 2022-02-07 00:00:00
4 | alexis | 2022-03-14 00:00:00
5 | ximena | 2022-04-22 00:00:00
6 | andrea | 2022-05-12 00:00:00
7 | raul | 2022-07-08 00:00:00
8 | rodolfo | 2022-08-02 00:00:00
9 | alondra | 2022-09-17 00:00:00
10 | esteban | 2022-10-27 00:00:00
11 | erick | 2022-11-04 00:00:00
12 | dulce | 2022-12-13 00:00:00
(24 rows)

```

```

partition_con_datos=# select * from my_partitioned_table_partition_1;

```

```

id | name | created_at
---+-----+-----
1 | luis | 2022-06-30 00:00:00
2 | anahi | 2022-01-05 00:00:00
3 | teresa | 2022-02-07 00:00:00
4 | alexis | 2022-03-14 00:00:00
5 | ximena | 2022-04-22 00:00:00
6 | andrea | 2022-05-12 00:00:00
(6 rows)

```

```

partition_con_datos=# select * from my_partitioned_table_partition_2;

```

```

id | name | created_at
---+-----+-----
7 | raul | 2022-07-08 00:00:00
8 | rodolfo | 2022-08-02 00:00:00
9 | alondra | 2022-09-17 00:00:00
10 | esteban | 2022-10-27 00:00:00
11 | erick | 2022-11-04 00:00:00
12 | dulce | 2022-12-13 00:00:00
(6 rows)

```

```

partition_con_datos=# 

```

```

9 | alondra | 2022-09-17 00:00:00
10 | esteban | 2022-10-27 00:00:00
11 | erick | 2022-11-04 00:00:00
12 | dulce | 2022-12-13 00:00:00
(6 rows)

partition_con_datos=# delete from my_partitioned_table2;
DELETE 24
partition_con_datos=# \dt
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | my_partitioned_table2 | table | postgres
public | my_partitioned_table_partition_1 | table | postgres
public | my_partitioned_table_partition_2 | table | postgres
(3 rows)

partition_con_datos=# select count(*) from my_partitioned_table2;
count
-----
0
(1 row)

partition_con_datos=# select count(*) from my_partitioned_table_partition_1;
count
-----
0
(1 row)

partition_con_datos=# select count(*) from my_partitioned_table_partition_2;
count
-----
0
(1 row)

partition_con_datos=# CREATE OR REPLACE FUNCTION partition_insert_trigger()
RETURNS TRIGGER AS $$
0
(1 row)

partition_con_datos=# CREATE OR REPLACE FUNCTION partition_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
IF (NEW.created_at >= '2022-01-01' AND NEW.created_at < '2022-07-
01') THEN
INSERT INTO my_partitioned_table_partition_1 VALUES (NEW.*);
ELSIF (NEW.created_at >= '2022-07-01' AND NEW.created_at < '2023-
01-01') THEN
INSERT INTO my_partitioned_table_partition_2 VALUES (NEW.*);
ELSE
RAISE EXCEPTION 'Date out of range';
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
partition_con_datos=# select count(*) from my_partitioned_table_partition_1;
count
-----
0
(1 row)

partition_con_datos=# CREATE TRIGGER insert_trigger
BEFORE INSERT ON my_partitioned_table2
FOR EACH ROW EXECUTE FUNCTION
partition_insert_trigger();
CREATE TRIGGER
partition_con_datos=# select count(*) from my_partitioned_table_partition_2;
count
-----
0
(1 row)

partition_con_datos=# 
```