

Instituto Tecnológico de Costa Rica

Escuela de Computación



Bases de Datos II

Grupo 20

Proyecto 2 - Replicación e Inteligencia de Negocios

Profesor (a):

Alberto Shum Chan

Estudiante (s):

José Adrián Amador Ávila - 2016101574

Pablo Jesús Mora Barrantes - 2019205110

Jose Andrés Vargas Serrano - 2019211290

Alajuela, I Semestre 2023

Introducción

El presente trabajo es la documentación para el segundo proyecto del curso Bases de Datos II de la carrera de Ingeniería en Computación del Instituto Tecnológico de Costa Rica.

En este documento se puede encontrar toda la información referente al proceso de creación de funciones, procedimientos almacenados, usuarios, tablas, modelo multidimensional, replicación, entre otros.

Descripción del proyecto

El proyecto tiene como objetivo el desarrollar un proyecto básico de Inteligencia de Negocios el cual incluya la configuración de elementos de la infraestructura tecnológica requerida, así como, la visualización de datos al servicio de los usuarios finales, permitiendo así poder experimentar con ambientes de apoyo a toma de decisiones.

El proyecto consiste en la creación de una base de datos a partir del ejemplo brindado por PostgreSQL (dvrental). Agregar varias funciones y procedimientos almacenados para poder modificar y consultar los datos. Implementación de un cierto nivel de seguridad con roles y usuarios. Creación de un sistema de replicación entre la base principal y una base réplica/esclava. Desarrollo de un modelo multidimensional de inteligencia de negocios. Y visualización de los datos por medio del software Tableau.

1. Creación de las funciones o procedimientos almacenados

- Insertar un nuevo cliente

The screenshot shows two code editors side-by-side, both titled "Query".

Top Query:

```
15 --esta función permite la inserción de un nuevo registro en la tabla "customer"
16
17 --DROP FUNCTION IF EXISTS insertCustomer( SMALLINT,VARCHAR,VARCHAR,VARCHAR,SMALLINT,BOOLEAN,DATE,INT);
18
19 CREATE OR REPLACE FUNCTION insertCustomer(s_id SMALLINT,f_name VARCHAR,
20                                         l_name VARCHAR,email VARCHAR,add_id SMALLINT,
21                                         a_bool BOOLEAN,
22                                         act INT)
23 RETURNS VARCHAR
24
25 LANGUAGE plpgsql
26
27 AS
28
29 $insertCustomer$
30
31 DECLARE c_result PUBLIC.customer%rowtype;
32
33 BEGIN
34
35     SELECT * FROM PUBLIC.customer c
36     INTO c_result
37     WHERE c.first_name = f_name AND c.last_name = l_name;
38
```

Bottom Query:

```
30      INTO c_result
31      WHERE c.first_name = f_name AND c.last_name = l_name;
32
33      IF c_result IS NOT NULL THEN
34
35          RETURN 'EL CLIENTE YA EXISTE';
36
37      ELSE
38
39          INSERT INTO PUBLIC.customer(customer_id,store_id,first_name,last_name,email,
40                                         address_id,activebool,create_date,active)
41
42              VALUES(NEXTVAL('customer_customer_id_seq'),s_id,f_name,l_name,email,add_id,a_bool,('now'::text)::date)
43
44          RETURN 'INSERTADO CORRECTAMENTE';
45
46      END IF;
47
48  END;
49
50 $insertCustomer$
51
52
53
54
55
56
57
58 SELECT PUBLIC.insertCustomer(1::SMALLINT,'Andrea','Mora','amora@gmail.com',22::SMALLINT,true,123::INT);
59
60
```

- Registrar un alquiler

```
Query  Query History
107 --drop procedure if exists registrar_alquiler(varchar, varchar, varchar, varchar);
108
109 -- Este procedimiento almacenado registra el alquiler de una película
110 create or replace procedure registrar_alquiler(
111     customer_first_name varchar,
112     customer_last_name varchar,
113     staff_username varchar,
114     movie_name varchar
115 )
116 language plpgsql
117 as $$
118 declare
119     customerid int;
120     staffid int;
121     rent_duration int;
122     today_time timestamp(0) without time zone := now();
123     var_inventory_id int := -1;
124     store_id int;
125     film_id int;
126     customer_results public.customer%rowtype;
127     film_results public.film%rowtype;
128     inventories_rec record;
129     -- variables para el manejo de excepciones
130     v_state TEXT;
131     v_msg TEXT;
```

```
Query  Query History
130     v_state TEXT;
131     v_msg TEXT;
132     v_detail TEXT;
133     v_hint TEXT;
134     v_context TEXT;
135 begin
136     -- obtenemos el id del cliente y el id de la tienda que está realizando la renta
137     select * from public.customer c
138     into customer_results
139     where c.first_name = customer_first_name and c.last_name = customer_last_name;
140
141     -- asignamos los valores de los ids a las variables a usar
142     if customer_results is not null then
143         customerid := customer_results.customer_id;
144         store_id := customer_results.store_id;
145         raise notice 'Id del cliente y de la tienda obtenidos';
146     end if;
147
148     -- obtenemos el id del empleado
149     select s.staff_id into staffid from public.staff s where s.username = 'Mike';
150
151     -- obtenemos la duración del préstamo de la película
152     select *
153     from public.film f
154     into film_results
155     where f.title = movie_name;
```

```
Query  Query History
154     into film_results
155     where f.title = movie_name;
156
157     -- asignamos los valores a las variables a usar
158     if film_results is not null then
159         film_id := film_results.film_id;
160         rent_duration := film_results.rental_duration;
161         raise notice 'Id de la película y su duración obtenidos';
162     end if;
163
164     -- obtenemos el id del primer inventario disponible
165     for inventories_rec in select public.film_in_stock(film_id, store_id)
166     loop
167         var_inventory_id := inventories_rec.film_in_stock;
168         raise notice 'Encontrado inventario disponible con la película a alquilar';
169         exit;
170     end loop;
171
172     if var_inventory_id != -1 then
173         -- insertamos los datos en un nuevo registro dentro de la tabla rental
174         insert into public.rental (rental_date, inventory_id, customer_id, return_date, staff_id)
175             values (today_time, var_inventory_id, customerid,
176                     today_time + cast('1 day' as interval) * rent_duration, staffid);
177         raise notice 'Transacción de préstamo exitosa';
178     else
179         raise notice 'Transacción de préstamo fallida';
180     end if;
181
182 exception when others then
183
184     get stacked diagnostics
185         v_state    = returned_sqlstate,
186         v_msg      = message_text,
187         v_detail   = pg_exception_detail,
188         v_hint     = pg_exception_hint,
189         v_context  = pg_exception_context;
190
191     raise notice E'Got exception:
192         state : %
193         message: %
194         detail : %
195         hint   : %
196         context: %', v_state, v_msg, v_detail, v_hint, v_context;
197
198     raise notice E'Got exception:
199         SQLSTATE: %
200         SQLERRM: %', SQLSTATE, SQLERRM;
201 end;$$
```

```
Query  Query History
178     else
179         raise notice 'No existen películas disponibles en el inventario para realizar la transacción';
180     end if;
181
182 exception when others then
183
184     get stacked diagnostics
185         v_state    = returned_sqlstate,
186         v_msg      = message_text,
187         v_detail   = pg_exception_detail,
188         v_hint     = pg_exception_hint,
189         v_context  = pg_exception_context;
190
191     raise notice E'Got exception:
192         state : %
193         message: %
194         detail : %
195         hint   : %
196         context: %', v_state, v_msg, v_detail, v_hint, v_context;
197
198     raise notice E'Got exception:
199         SQLSTATE: %
200         SQLERRM: %', SQLSTATE, SQLERRM;
201 end;$$
```

- Registrar una devolución

```

Query  Query History
61  /*
62  Procedimiento encargado de registrar las devoluciones de una película, esto anota los datos asociados en la tabla
63  -rental_id = número de id de la renta por la cuál se está realizando la devolución
64  -v_username = nombre del usuario del empleado que está llevando a cabo el proceso de devolución
65  */
66 CREATE OR REPLACE PROCEDURE devolucion(v_rental_id integer, v_username varchar)
67   LANGUAGE plpgsql
68   AS $$
69   DECLARE ---definimos variables que nos ayudarán más adelante
70   v_customer_id INTEGER;
71   v_staff_id INTEGER;
72   v_rental_date DATE;
73
74
75   --- seleccionamos el id del cliente perteneciente a la renta y se lo pasamos a la variable v_customer_id
76   select customer_id into v_customer_id
77   from rental
78   where v_rental_id = rental_id;
79
80   --- seleccionamos el id del empleado encargado de la devolución y se lo pasamos a la variable v_staff_id
81   select staff_id into v_staff_id
82   from staff
83   where v_username = username;
84
85   --- seleccionamos la fecha en la que se realizó la renta y se lo pasamos a la variable v_rental_date
86   select rental_date into v_rental_date
87   from rental
88   where v_rental_id = rental_id;
89
90   --- comenzamos con el proceso de inserción en la tabla payment
91   insert into payment VALUES
92   (nextval('payment_payment_id_seq'), v_customer_id, v_staff_id, v_rental_id,
93   (select extract(day from localtimestamp - v_rental_date))*500, localtimestamp);
94   COMMIT;
95   /*
96   Datos que tal vez no se entiendan en el insert:
97   -nextval('payment_payment_id_seq') : valor que sigue en la secuencia hecha de la tabla payment.
98   -(select extract(day from localtimestamp - v_rental_date))*500 : fórmula que calcula lo que se tiene que
99   pagar al devolver la película, el localtimestamp - v_rental_date devolvía un intervalo de tiempo, por lo
100  que se extraen los días del intervalo y el número de días se multiplica por 500.
101  -localtimestamp: timestamp without timezone, se usó esta para que fuera igual al tipo de datos de las fechas
102  que ya venían ingresados en la base de datos.
103  */
104 END;
105 $$;
106
107 --drop procedure if exists registrar_alquiler(varchar, varchar, varchar, varchar);
108
Data Output  Messages  Notifications

```

- Buscar una película

```

Query  Query History
204 -- Esta función busca el lenguaje por medio del ID que es
205 -- pasado como único parámetro y retorna el nombre como un varchar
206 create or replace function get_film_language(
207     film_language_id int
208 )
209 returns varchar
210 language plpgsql
211 as $$
212 declare
213     language_name varchar;
214 begin
215     select l.name into language_name from public.language l where l.language_id = film_language_id;
216     return language_name;
217 end; $$
218
219 -- select public.get_film_language(1);
220 -- DROP FUNCTION get_film_language(integer);
221
222
223 -- Esta función busca las películas que coincide con el parámetro de entrada
224 -- y retorna una tabla con todas las películas, además, cambia el campo
225 -- language_id por el nombre del lenguaje
226 create or replace function buscar_pelicula(
227     film_title varchar
228
Data Output  Messages  Notifications
Query  Query History
226 create or replace function buscar_pelicula(
227     film_title varchar
228 )
229 returns table (film_id int, title varchar, description varchar, release_year int, film_lan varchar,
230                 rental_duration int, rental_rate int, duration int, replacement_cost numeric(5,2), rating varchar
231 language plpgsql
232 as $$
233 declare
234     var_record record;
235     -- variables para el manejo de excepciones
236     v_state TEXT;
237     v_msg TEXT;
238     v_detail TEXT;
239     v_hint TEXT;
240     v_context TEXT;
241 begin
242     for var_record in (
243         select f.film_id, f.title, f.description, f.release_year, f.language_id, f.rental_duration,
244               f.rental_rate, f.length, f.replacement_cost, f.rating
245         from public.film f
246         where f.title like '%' || film_title || '%'
247     ) loop
248         film_id := var_record.film_id;
249         title := upper(var_record.title);
250         description := var_record.description;
251     end loop;
252 end; $$

Data Output  Messages  Notifications

```

```
Query Query History
248     film_id := var_record.film_id;
249     title := upper(var_record.title);
250     description := var_record.description;
251     release_year := var_record.release_year;
252     film_lan := public.get_film_language(var_record.language_id);
253     rental_duration := var_record.rental_duration;
254     rental_rate := var_record.rental_rate;
255     duration := var_record.length;
256     replacement_cost := var_record.replacement_cost;
257     rating := var_record.rating;
258     return next;
259 end loop;
260
261 exception when others then
262
263     get stacked diagnostics
264         v_state = returned_sqlstate,
265         v_msg = message_text,
266         v_detail = pg_exception_detail,
267         v_hint = pg_exception_hint,
268         v_context = pg_exception_context;
269
270     raise notice E'Got exception:
271         state : %
272         message: %
273
274         context: %', v_state, v_msg, v_detail, v_hint, v_context;
275
276     raise notice E'Got exception:
277         SQLSTATE: %
278         SQLERRM: %', SQLSTATE, SQLERRM;
279
280 end; $$
```

1.2 Seguridad

- ## - Rol EMP

```
281 -- creamos el role EMP sin ninguna opción (todas las opciones están por defecto, privilegios más básicos)
282 create role EMP;
283
284
285 -- se le conceden permisos para ejecutar las funciones y procedimientos almacenados de:
286 -- buscar_pelicula(varchar)
287 grant execute on function public.buscar_pelicula(varchar) to EMP;
288 -- registrar_alquiler(varchar, varchar, varchar, integer)
289 grant execute on procedure public.registrar_alquiler(varchar, varchar, varchar, varchar) to EMP;
290 -- registrar_devolucion() - W.I.P
291 grant execute on procedure public.devolucion(integer, varchar) to EMP;
```

- Rol ADMIN

```
293 CREATE ROLE ADMIN INHERIT;
294
295 GRANT EMP TO ADMIN;
296
297 GRANT EXECUTE ON PROCEDURE devolucion(integer, varchar) TO ADMIN;
298
```

- Usuario video

```
299 -- se crean los usuarios y se les asignan los roles correspondientes
300 CREATE USER video;
301
302 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO video;
303
304 GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO video;
305
306 GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO video;
```

- Usuario empleado1 y administrador1

```
308 CREATE USER empleado1 WITH PASSWORD 'empleado1';
309
310 CREATE USER administrador1 WITH PASSWORD 'admin1';
311
312 GRANT EMP TO empleado1;
313
314 GRANT ADMIN TO administrador1;
315
316 CREATE USER repuser REPLICATION WITH PASSWORD '1234';
317
318
```

2. Réplicas

Para el proceso de replicación se usó el Streaming Replication de PostgreSQL. Todos los pasos seguidos se encuentran en el archivo *replication_process.pdf* dentro de la carpeta *Replicacion*. Dentro de la carpeta también se encuentra la versión en Markdown.

Acá se dejan las capturas del archivo PDF:

Proceso de replicación

A continuación se mostrará la lista de comandos a ejecutar para realizar streaming replication del proyecto 02 del Curso de Base de Datos (exclusivo para [Ubuntu con PostgreSQL 14](#)):

- Primero nos movemos al usuario postgres con el siguiente comando

```
$ sudo -i -u postgres
```

- De no tener el comando `initdb` en el path, estos se encuentran en el directorio `/usr/lib/postgresql/14/bin`. Para agregarlos como instrucción ejecutable se puede usar el siguiente comando:

```
$ sudo ln -s /usr/lib/postgresql/9.1/bin/initdb /usr/local/bin/
```

Nota: la versión varía dependiendo de la que se tenga instalada, además, este es el directorio para el sistema operativo Ubuntu, este puede variar dependiendo del OS.

- Creamos el cluster que va actuar como `instancia master o primary database`

```
$ cd 14/  
$ initdb -D rep_primary_db/data
```

En este caso especificamos el directorio donde el cluster va a ser almacenado. Como yo tengo instalado PostgreSQL 14, el directorio al que quiero ingresar con `cd 14/` es la versión del mismo.

En este caso especificamos el directorio donde el cluster va a ser almacenado. Como yo tengo instalado PostgreSQL 14, el directorio al que quiero ingresar con `cd 14/` es la versión del mismo.

- Editamos el archivo `postgresql.conf` del cluster creado (este archivo debería de estar dentro del directorio especificado en el paso anterior)

```
$ nano rep_primary_db/data/postgresql.conf
```

Dentro del editor con `Ctrl + W` ingresamos `listen_address` para encontrar la linea. Le quitamos `#` para descomentarlo y le colocamos la dirección IP a la cual deseamos que escuche, en este caso usamos `*` para indicar que escuche para cualquier cliente

```
listen_address = '*'      # what IP address(es) to listen on;
```

Luego pasamos a la línea `port` y asignamos algún puerto libre, en mi caso sería `5434`

```
port = 5433
```

1 / 5

replication_process.md

30/4/2023

Nota: con `netstat` se puede verificar si el puerto está siendo usado. En Ubuntu se puede instalar con el comando `sudo apt install net-tools`. Y con el comando `sudo netstat -lntp | grep -w '5434'` se realiza la verificación.

- Iniciamos la instancia `master/primary`

- Iniciamos la instancia `master/primary`

```
$ pg_ctl -D rep_primary_db/data start
```

- Ahora nos conectamos a `psql` usando el puerto especificado y el nombre de la base de datos

```
$ psql postgres --port=5434
```

Creamos la base de datos `dvdrental` la cual va a ser generada por medio del comando `pg_restore`

```
postgres=# CREATE DATABASE dvdrental;
CREATE DATABASE
```

Salimos de `psql`

```
postgres=# \q
```

Ahora estando en el directorio donde tenemos guardado el archivo `dvdrental.tar`, ejecutamos

```
$ pg_restore -U postgres -p 5434 -d dvdrental dvdrental.tar
```

Con `-U` especificamos el usuario (el cual es `postgres`), con `-p` especificamos el puerto y con `-d` el nombre de la base de datos.

- Volvemos a ingresar dentro de `psql` en el puerto de la base principal pero en este caso en la base de

- Volvemos a ingresar dentro de `psql` en el puerto de la base principal pero en este caso en la base de datos `dvdrental`

```
$ psql dvdrental --port=5434
```

Y creamos un usuario dueño de la replicación con privilegios de replicación

```
dvdrental=# CREATE USER repuser REPLICATION;
```

2 / 5

replication_process.md

30/4/2023

Salimos de `psql` con

```
dvdrental=# \q
```

- Ahora editamos el archivo `pg_hba.conf`

```
$ nano rep_primary_db/data/pg_hba.conf
```

Y agregamos

host	all	repuser	127.0.0.1/32	trust
------	-----	---------	--------------	-------

Y agregamos

host	all	repuser	127.0.0.1/32	trust
------	-----	---------	--------------	-------

El archivo debería de quedar algo parecido a lo siguiente

#	TYPE	DATABASE	USER	ADDRESS	METHOD
#	"local"	is for Unix domain socket connections only			
local	all	all			trust
#	IPv4	local connections:			
host	all	all		127.0.0.1/32	trust
host	all	repuser		127.0.0.1/32	trust

Guardamos con **Ctrl + O** y cerramos el archivo con **Ctrl + X**.

- Reiniciamos la instancia

\$ pg_ctl -D rep_primary_db/data restart
--

- Ahora pasamos a crear la base de datos réplica de la primary database con el comando **pg_basebackup**, el cual simplemente se va a conectar a la base de datos principal y va a copiar todos los archivos de datos a la réplica

\$ pg_basebackup -h localhost -U repuser --checkpoint=fast -D rep_replica_db/data -R --slot=replica_dvdrental -C --port=5434
--

- Ahora pasamos a crear la base de datos réplica de la primary database con el comando `pg_basebackup`, el cual simplemente se va a conectar a la base de datos principal y va a copiar todos los archivos de datos a la réplica

```
$ pg_basebackup -h localhost -U repuser --checkpoint=fast -D rep_replica_db/data -R --slot=replica_dvdrental -C --port=5434
```

Se especifica la dirección, en este caso `localhost`, el usuario `repuser` dueño de la replicación, con `checkpoint=fast` se asegura que el proceso de copia se inicie instantáneamente, el directorio donde se va a guardar la base de datos réplica `rep_replica_db/data`, `-R` indica replicación, se le

3 / 5

replication_process.md

30/4/2023

da un nombre significativo con `slot`, con `-C` la base principal pueda reciclar el archivo `wal` solo después que la réplica lo haya consumido por completo, y por último el puerto `5434`.

- Ahora, abrimos una nueva terminal e iniciamos sesión como el usuario `postgres`

```
$ sudo -i -u postgres
```

- Abrimos el archivo `postgresql.conf` y cambiamos el puerto de la réplica para que no haya conflictos

```
nano 14/rep_replica_db/data/postgresql.conf
```

```
port = 5435
```

Guardamos y salimos del archivo con **Ctrl + O** y luego **Ctrl + X**.

- Iniciamos la base de datos réplica con el comando **pg_ctl**

```
$ pg_ctl -D 14/rep_replica_db/data start
```

- En la terminal original entramos en **psql** en el puerto que indicamos para la base de datos principal

```
$ psql dvdrental --port=5434
```

Realizamos un **SELECT** de los registros de la tabla **staff**, por ejemplo,

```
dvrental=# SELECT * FROM staff;
```

Y en la otra terminal ingresamos también a **psql** pero en esta ocasión en el puerto de la base de datos réplica

```
$ psql dvdrental --port=5435
```

Y realizamos el mismo **SELECT** en tabla **staff**

Y realizamos el mismo `SELECT` en tabla `staff`

```
dvdrental=# SELECT * FROM staff;
```

4 / 5

replication_process.md

30/4/2023

Si ambas transacciones devuelven los mismos datos esto quiere decir que la réplica está funcionando. Cualquier dato que agregemos a la base principal será reflejada en la base réplica.

Verificar estado de las base de datos

Con

```
dvdrental=# \x
Extended display is on
dvdrental=# SELECT * FROM PG_STAT_REPLICATION;
```

Podemos ver el estado de la base de datos principal con respecto a su réplica (este comando se ejecuta dentro de la base de datos principal).

Y con

```
dvdrental=# \x
Extended display is on
dvdrental=# SELECT * FROM PG_STAT_WAL_RECEIVER;
```

Podemos ver el estado de la base de datos principal con respecto a su réplica (este comando se ejecuta dentro de la base de datos principal).

Y con

```
dvdrental=# \x
Extended display is on
dvdrental=# SELECT * FROM PG_STAT_WAL_RECEIVER;
```

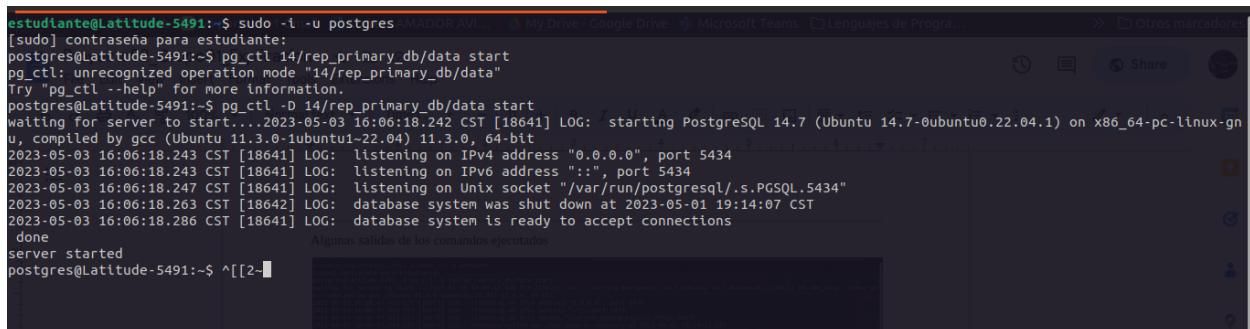
Podemos ver el estado de la base de datos réplica (este comando se ejecuta dentro de la base de datos réplica).

Referencias:

- <https://www.postgresql.org/docs/current/app-pgrestore.html>
- <https://www.postgresql.org/docs/current/creating-cluster.html>
- <https://www.postgresqltutorial.com/postgresql-getting-started/load-postgresql-sample-database/>
- <https://www.youtube.com/watch?v=Yy0GJjRQcRQ>
- <https://www.postgresql.org/docs/current/app-initdb.html>
- <https://www.postgresql.org/docs/current/app-pgbasebackup.html>

Algunas salidas de los comandos ejecutados

Esto representa la forma de iniciar las bases de datos, primera la principal y luego la réplica



The screenshot shows a terminal window with the following text output:

```
estudiante@Latitude-5491:~$ sudo -i -u postgres
[sudo] contraseña para estudiante:
postgres@Latitude-5491:~$ pg_ctl 14/rep_primary_db/data start
pg_ctl: unrecognized operation mode "14/rep_primary_db/data"
Try "pg_ctl --help" for more information.
postgres@Latitude-5491:~$ pg_ctl -D 14/rep_primary_db/data start
waiting for server to start...2023-05-03 16:06:18.242 CST [18641] LOG:  starting PostgreSQL 14.7 (Ubuntu 14.7-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.3.0-1ubuntu1-22.04) 11.3.0, 64-bit
2023-05-03 16:06:18.243 CST [18641] LOG:  listening on IPv4 address "0.0.0.0", port 5434
2023-05-03 16:06:18.243 CST [18641] LOG:  listening on IPv6 address "::", port 5434
2023-05-03 16:06:18.247 CST [18641] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5434"
2023-05-03 16:06:18.263 CST [18642] LOG:  database system was shut down at 2023-05-01 19:14:07 CST
2023-05-03 16:06:18.286 CST [18641] LOG:  database system is ready to accept connections
done
server started
postgres@Latitude-5491:~$ ^[[2~
```

Algunas salidas de los comandos ejecutados

```
estudiante@Latitude-5491:~$ sudo -i -u postgres
[sudo] contraseña para estudiante:
postgres@Latitude-5491:~$ pg_ctl -D /rep_replica_db/data start
waiting for server to start....2023-05-03 16:06:47.420 CST [18673] LOG:  starting PostgreSQL 14.7 (Ubuntu 14.7-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.3.0-1ubuntu1-22.04) 11.3.0, 64-bit
2023-05-03 16:06:47.420 CST [18673] LOG:  listening on IPv4 address "0.0.0.0", port 5435
2023-05-03 16:06:47.420 CST [18673] LOG:  listening on IPv6 address "::", port 5435
2023-05-03 16:06:47.423 CST [18673] LOG:  listening on Unix socket "/var/run/postgresql/.PGSQL.5435"
2023-05-03 16:06:47.434 CST [18674] LOG:  database system was shut down in recovery at 2023-05-01 19:14:15 CST
2023-05-03 16:06:47.434 CST [18674] LOG:  entering standby mode
2023-05-03 16:06:47.442 CST [18674] LOG:  consistent recovery state reached at 0/30D2E20
2023-05-03 16:06:47.442 CST [18674] LOG:  invalid record length at 0/30D2E20: wanted 24, got 0
2023-05-03 16:06:47.443 CST [18673] LOG:  database system is ready to accept read-only connections
2023-05-03 16:06:47.454 CST [18678] LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
2023-05-03 16:06:47.460 CST [18674] LOG:  redo starts at 0/30D2E20
done
server started
postgres@Latitude-5491:~$
```

Edición del archivo postgresql.conf con el comando nano

```
GNU nano 6.2                                     14/rep_primary_db/data/postgresql.conf

#hba_file = 'ConfigDir/pg_hba.conf'      # host-based authentication file
# Projecto02 Grupo01 Explicación # (change requires restart)
#ident_file = 'ConfigDir/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 = ''                   # write an extra PID file
#                                         # (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) para que la réplica esté
                                                # conectada a la base principal y los cambios a la base principal serán reflejados en la base réplica.

port = 5434                                     # (change requires restart) datos
max_connections = 100                          # (change requires restart)
#superuser_reserved_connections = 3           # (change requires restart)
#unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                                # (change requires restart)
#unix_socket_group = ''                        # (change requires restart)
#unix_socket_permissions = 0777                # begin with 0 to use octal notation
                                                # (change requires restart)
#bonjour = off                                # advertise server via Bonjour
                                                # (change requires restart) igual con respecto a su réplica (este comando se ejecuta
                                                # defaults to the computer name
                                                # (change requires restart)

#-----#
# Ayuda   ^O Guardar   ^W Buscar   ^K Cortar   ^T Ejecutar   ^C Ubicación   M-U Deshacer   M-A Poner Marca   M-J A llave
# Salir   ^O Leer fich   ^A Reemplazar   ^L Pegar   ^T Justificar   ^U Ir a línea   M-F Rehacer   M-6 Copiar   ^O Buscar atrás

```

Edición del archivo pg_hba.conf con el usuario repuser

Salida de ejecutar el comando initdb para crear un cluster de base de datos

```
postgres@Latitude-5491:~$ initdb -D /tmp/primarydb
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "es_CR.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "spanish".

Data page checksums are disabled.

creating directory /tmp/primarydb ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB   En este caso especificamos el directorio donde el clúster va a ser almacenado. Como yo tengo PostgreSQL 14, el directorio al que quiero ingresar con pg_ctl es la versión del mismo.
selecting default time zone ... America/Costa_Rica
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

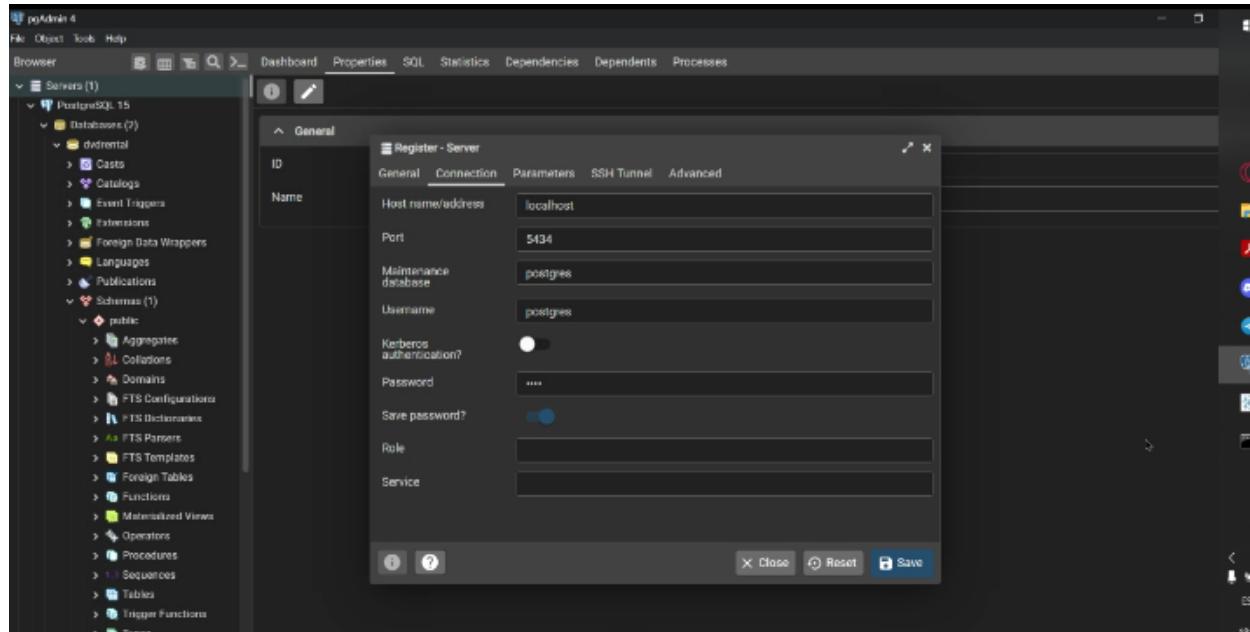
initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

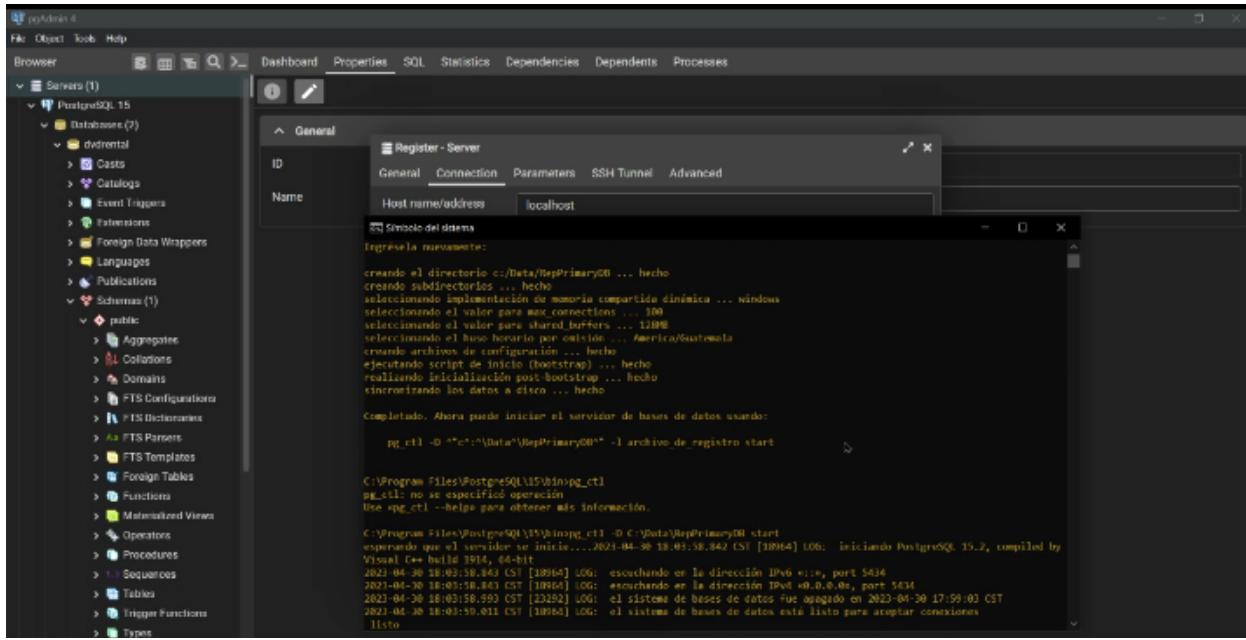
Success. You can now start the database server using:

    pg_ctl -D /tmp/primarydb -l logfile start

postgres@Latitude-5491:~$
```

Capturas de pantalla de la evidencia de replicación en windows:





Símbolo del sistema

```

C:\Program Files\PostgreSQL\15\bin>initdb -D c:/Data/RepPrimaryDB -W -A md5
Los archivos de este cluster serán de propiedad del usuario «Pablo MB».
Este usuario también debe ser quien ejecute el proceso servidor.

El cluster será inicializado con configuración regional «Spanish_Costa Rica.1252».
La codificación por omisión ha sido por lo tanto definida a «WIN1252».
La configuración de búsqueda en texto ha sido definida a «spanish».

Las sumas de verificación en páginas de datos han sido desactivadas.

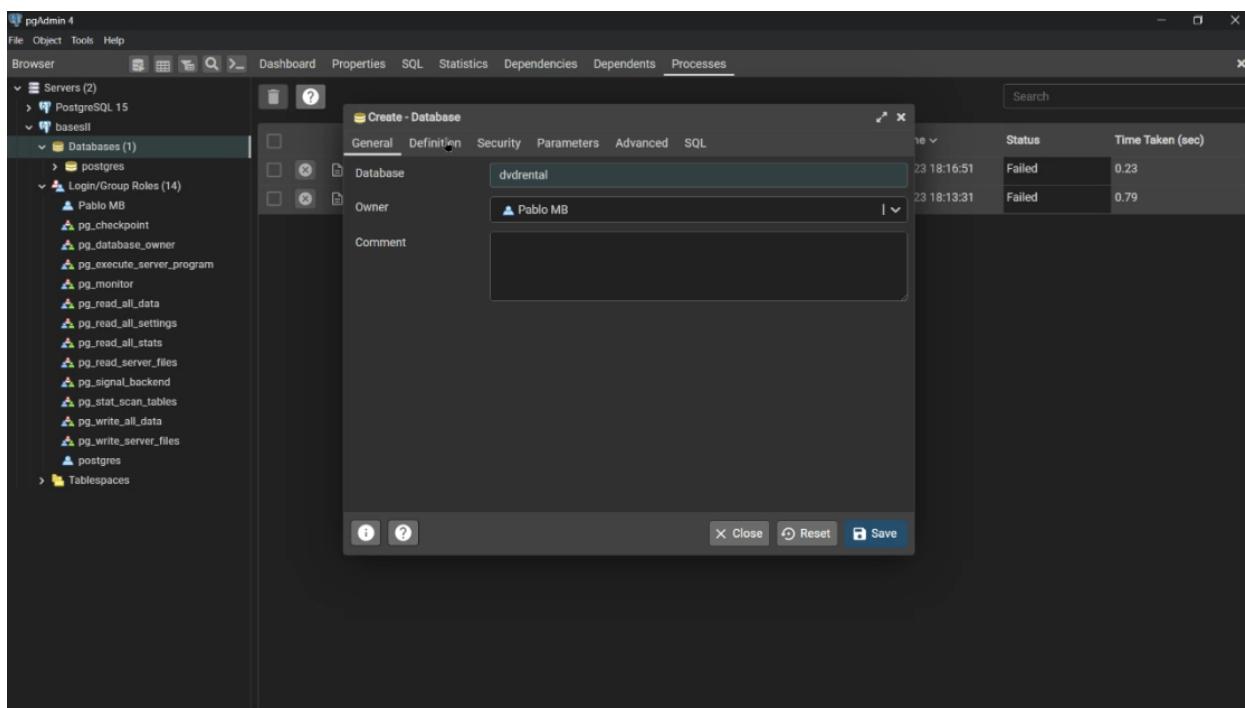
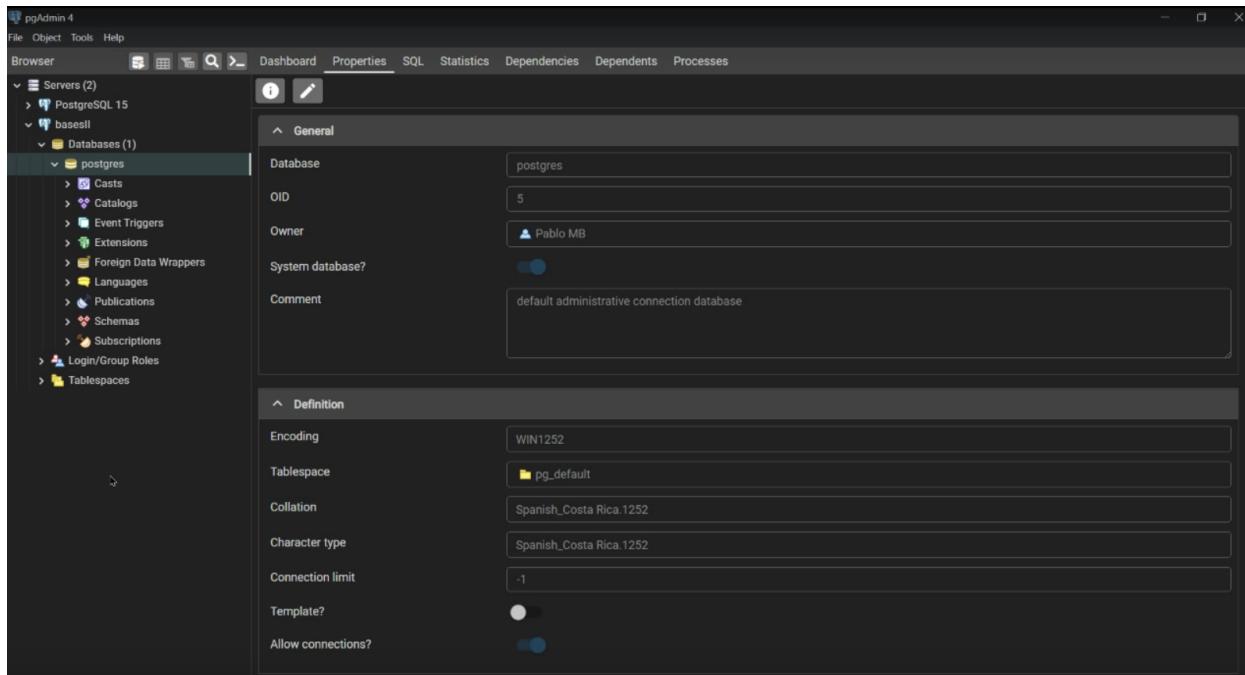
Ingrese la nueva contraseña del superusuario:
Ingrésela nuevamente:

creando el directorio c:/Data/RepPrimaryDB ... hecho
creando subdirectorios ... hecho
seleccionando implementación de memoria compartida dinámica ... windows
seleccionando el valor para max_connections ... 100
seleccionando el valor para shared_buffers ... 128MB
seleccionando el huso horario por omisión ... América/Guatemala
creando archivos de configuración ... hecho
ejecutando script de inicio (bootstrap) ... hecho
realizando inicialización post-bootstrap ... hecho
sincronizando los datos a disco ... hecho

Completado. Ahora puede iniciar el servidor de bases de datos usando:

pg_ctl -D ^\"c:\Data\RepPrimaryDB\" -l archivo_de_registro start

```



	PID	Type	Server	Object	Start Time	Status	Time Taken (sec)
	19080	Restore	basesll (localhost:5434)	dvdrental	30/4/2023 18:18:56	Finished	0.55

```

Query Query History
1 CREATE USER repuser REPLICATION;
2

pg_hba.conf •
C:\Data\RepPrimaryDB> pg_hba.conf
66 = warning: ignoring file pg_hba.conf in a running system; you have to
69 # SIGHUP the server for the changes to take effect, run "pg_ctl reload",
70 # or execute "SELECT pg_reload_conf()".
71 #
72 # Put your actual configuration here
73 # -----
74 #
75 # If you want to allow non-local connections, you need to add more
76 # "host" records. In that case you will also need to make PostgreSQL
77 # listen on a non-local interface via the listen_addresses
78 # configuration parameter, or via the -i or -h command line switches,
79
80
81
82 # TYPE DATABASE USER ADDRESS METHOD
83
84 # "Local" is for Unix domain socket connections only
85 local all all md5
86 # IPv4 Local connections:
87 host all all 127.0.0.1/32 md5
88 host all repuser| 127.0.0.1/32 md5
89 # IPv6 local connections:
90 host all all ::1/128 md5
91 # Allow replication connections from localhost, by a user with the
92 # replication privilege.
93 local replication all md5
94 host replication all 127.0.0.1/32 md5
95 host replication all ::1/128 md5
96

C:\Program Files\PostgreSQL\15\bin>pg_ctl -D C:\Data\RepPrimaryDB restart
2023-04-30 18:25:41.109 CST [18964] LOG: se recibió petición de apagado rápido
esperando que el servidor se detenga....2023-04-30 18:25:41.116 CST [18964] LOG: abortando transacciones activas
2023-04-30 18:25:41.121 CST [18964] LOG: proceso ayudante «logical replication launcher» (PID 19492) terminó con código
de salida 1
2023-04-30 18:25:41.143 CST [16836] LOG: apagando
2023-04-30 18:25:41.146 CST [16836] LOG: empezando checkpoint: shutdown immediate
2023-04-30 18:25:41.163 CST [16836] LOG: checkpoint completado: se escribió 1 buffers (0.0%); 0 archivo(s) de WAL añadi
do(s), 0 eliminado(s), 0 reciclado(s); escritura=0.001 s, sincronización=0.003 s, total=0.020 s; archivos sincronizados=
1, más largo=0.003 s, promedio=0.003 s; distancia=0 kB, estimado=8542 kB
2023-04-30 18:25:41.193 CST [18964] LOG: el sistema de bases de datos está apagado
    listo
servidor detenido
esperando que el servidor se inicie....2023-04-30 18:25:41.293 CST [1236] LOG: iniciando PostgreSQL 15.2, compiled by V
isual C++ build 1914, 64-bit
2023-04-30 18:25:41.295 CST [1236] LOG: escuchando en la dirección IPv6 «::», port 5434
2023-04-30 18:25:41.295 CST [1236] LOG: escuchando en la dirección IPv4 «0.0.0.0», port 5434
2023-04-30 18:25:41.336 CST [2892] LOG: el sistema de bases de datos fue apagado en 2023-04-30 18:25:41 CST
2023-04-30 18:25:41.354 CST [1236] LOG: el sistema de bases de datos está listo para aceptar conexiones
    listo
servidor iniciado

```

```
C:\Program Files\PostgreSQL\15\bin>pg_basebackup -h localhost -U repuser --checkpoint=fast -D C:\Data\RepReplicaDB -R --slot=replica_dydrental -C --port=5434
2023-04-30 18:29:11.259 CST [16512] FATAL: la autenticación password falló para el usuario «repuser»
2023-04-30 18:29:11.259 CST [16512] DETALLE: El usuario «repuser» no tiene una contraseña asignada.
    La conexión coincidió con la línea 95 de pg_hba.conf: «host      replication      all      ::1/128
        md5»
Contraseña:
2023-04-30 18:29:20.433 CST [13948] FATAL: la autenticación password falló para el usuario «repuser»
2023-04-30 18:29:20.433 CST [13948] DETALLE: El usuario «repuser» no tiene una contraseña asignada.
    La conexión coincidió con la línea 95 de pg_hba.conf: «host      replication      all      ::1/128
        md5»
pg_basebackup: error: falló la conexión al servidor en «localhost» (::1), puerto 5434: FATAL: la autenticación password falló para el usuario «repuser»
```

```
1 CREATE USER repuser REPLICATION;
2 alter user repuser password '1234';
```

Contraseña:

```
2023-04-30 18:31:39.773 CST [18560] LOG: empezando checkpoint: immediate force wait
2023-04-30 18:31:39.816 CST [18560] LOG: checkpoint completado: se escribió 13 buffers (0.1%); 0 archivo(s) de WAL añadido(s), 0 eliminado(s), 1 reciclado(s); escritura=0.001 s, sincronización=0.019 s, total=0.044 s; archivos sincronizados =7, más largo=0.003 s, promedio=0.003 s; distancia=7688 kB, estimado=7688 kB
```

C:\Program Files\PostgreSQL\15\bin>

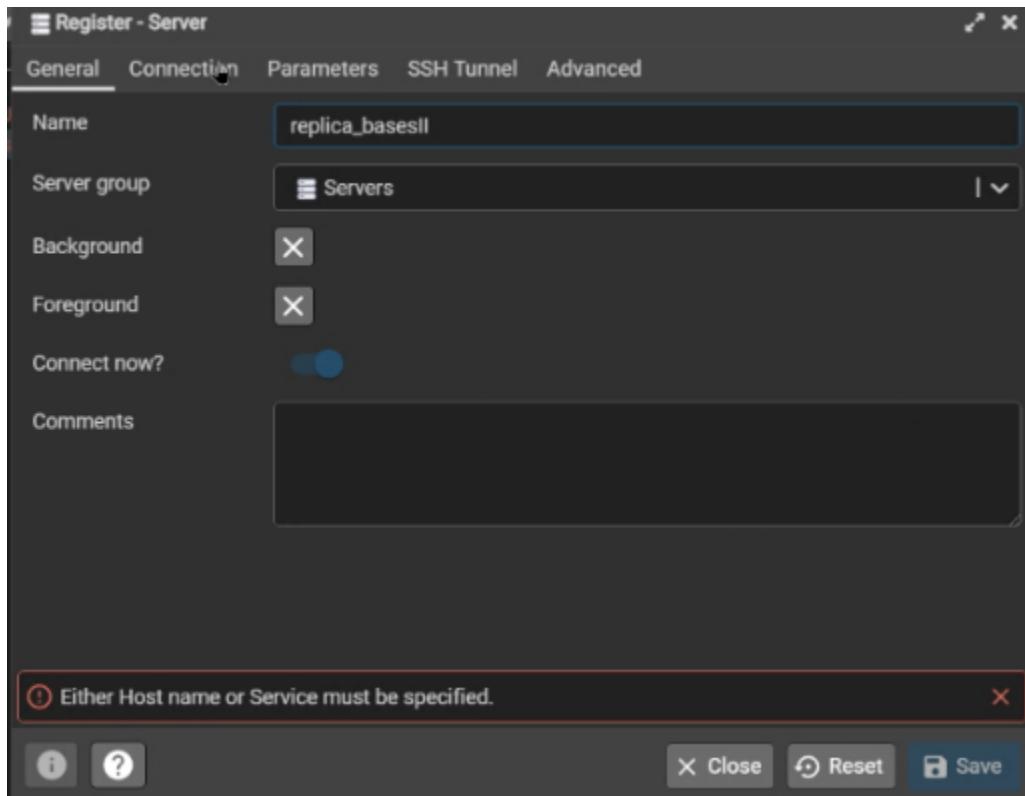
Este equipo > SSD M.2 // 512 GB (C) > Data > RepReplicaDB >				
	Nombre	Fecha de modificación	Tipo	Tamaño
do	base	30/04/2023 18:31	Carpeta de archivos	
os	global	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_commit_ts	30/04/2023 18:31	Carpeta de archivos	
os	pg_dynshmem	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_logical	30/04/2023 18:31	Carpeta de archivos	
os	pg_multixact	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_notify	30/04/2023 18:31	Carpeta de archivos	
os	pg_replslot	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_serial	30/04/2023 18:31	Carpeta de archivos	
os	pg_snapshots	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_stat	30/04/2023 18:31	Carpeta de archivos	
os	pg_stat_tmp	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_subtrans	30/04/2023 18:31	Carpeta de archivos	
os	pg_tblspc	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_twophase	30/04/2023 18:31	Carpeta de archivos	
os	pg_wal	30/04/2023 18:31	Carpeta de archivos	
le pantalla	pg_xact	30/04/2023 18:31	Carpeta de archivos	
os	backup_label	30/04/2023 18:31	Archivo	1 KB
512 GB (C)	backup_manifest	30/04/2023 18:31	Archivo	188 KB
le pantalla	pg_hba	30/04/2023 18:31	Archivo CONF	5 KB
os	pg_ident	30/04/2023 18:31	Archivo CONF	2 KB
le pantalla	PG_VERSION	30/04/2023 18:31	Archivo	1 KB
os	postgresqlauto	30/04/2023 18:31	Archivo CONF	1 KB
le pantalla	postgresql	30/04/2023 18:31	Archivo CONF	30 KB
os	standby.signal	30/04/2023 18:31	Archivo SIGNAL	0 KB

```
C:\Data>RepReplicaDB> postgresql.conf
41
42 #data_directory = 'ConfigDir'      # use data in another directory
43 |          # (change requires restart)
44 #hba_file = 'ConfigDir/pg_hba.conf' # host-based authentication file
45 |          # (change requires restart)
46 #ident_file = 'ConfigDir/pg_ident.conf' # ident configuration file
47 |          # (change requires restart)
48
49 # If external_pid_file is not explicitly set, no extra PID file is written.
50 #external_pid_file = ''           # write an extra PID file
51 |          # (change requires restart)
52
53 #
54 #-----
55 # CONNECTIONS AND AUTHENTICATION
56 #
57
58 # - Connection Settings -
59
60 listen_addresses = '*'      # what IP address(es) to listen on;
61 |          # comma-separated list of addresses;
62 |          # defaults to 'localhost'; use '*' for all
63 |          # (change requires restart)
64 port = 5435      # (change requires restart)
65 max_connections = 100      # (change requires restart)
66 #superuser_reserved_connections = 3 # (change requires restart)
67 #unix_socket_directories = ''    # comma-separated list of directories
68 |          # (change requires restart)
69 #unix_socket_group = ''        # (change requires restart)
70 #unix_socket_permissions = 0777  # begin with 0 to use octal notation
71 |          # (change requires restart)
72 #bonjour = off                # advertise server via Bonjour
73 |          # (change requires restart)
74 #bonjour_name = ''            # defaults to the computer name
75 |          # (change requires restart)
76
77 # - TCP settings -
78 # see "man tcp" for details
79
80 #tcp_keepalives_idle = 0       # TCP_KEEPIDLE, in seconds;
```

```
Símbolo del sistema
2023-04-30 18:30:06.785 CST [15588] FATAL: la autenticación password falló para el usuario «repuser»
2023-04-30 18:30:06.785 CST [15588] DETALLE: El usuario «repuser» no tiene una contraseña asignada.
    La conexión coincidió con la línea 95 de pg_hba.conf: «host      replication      all              ::/128
                                            md5»
Contraseña: 2023-04-30 18:30:41.440 CST [18560] LOG: empezando checkpoint: time
2023-04-30 18:30:41.573 CST [18560] LOG: checkpoint completado: se escribió 4 buffers (0.0%); 0 archivo(s) de WAL añadido(s), 0 eliminado(s), 0 reciclado(s); escritura=0.110 s, sincronización=0.009 s, total=0.135 s; archivos sincronizados=3, más largo=0.004 s, promedio=0.003 s; distancia=0 kB, estimado=0 kB

Contraseña:
2023-04-30 18:31:39.773 CST [18560] LOG: empezando checkpoint: immediate force wait
2023-04-30 18:31:39.816 CST [18560] LOG: checkpoint completado: se escribió 13 buffers (0.1%); 0 archivo(s) de WAL añadido(s), 0 eliminado(s), 1 reciclado(s); escritura=0.001 s, sincronización=0.019 s, total=0.044 s; archivos sincronizados=7, más largo=0.003 s, promedio=0.003 s; distancia=7688 kB, estimado=7688 kB

C:\Program Files\PostgreSQL\15\bin>pg_ctl -D C:\Data\RepReplicaDB start
esperando que el servidor se inicie....2023-04-30 18:34:00.327 CST [22904] LOG: iniciando PostgreSQL 15.2, compilado por
Visual C++ build 1914, 64-bit
2023-04-30 18:34:00.328 CST [22904] LOG: escuchando en la dirección IPv6 «::», port 5435
2023-04-30 18:34:00.328 CST [22904] LOG: escuchando en la dirección IPv4 «0.0.0.0», port 5435
2023-04-30 18:34:00.371 CST [20948] LOG: el sistema de bases de datos fue interrumpido; última vez en funcionamiento en
2023-04-30 18:31:39 CST
2023-04-30 18:34:01.175 CST [20948] LOG: entrando al modo standby
2023-04-30 18:34:01.189 CST [20948] LOG: redo comienza en 0/3000028
2023-04-30 18:34:01.192 CST [20948] LOG: el estado de recuperación consistente fue alcanzado en 0/3000138
2023-04-30 18:34:01.192 CST [22904] LOG: el sistema de bases de datos está listo para aceptar conexiones de sólo lectura
a
listo
servidor iniciado
```

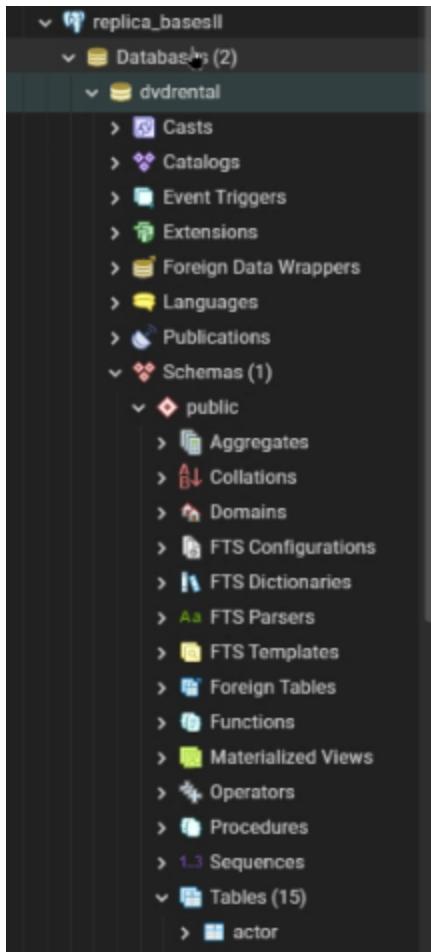


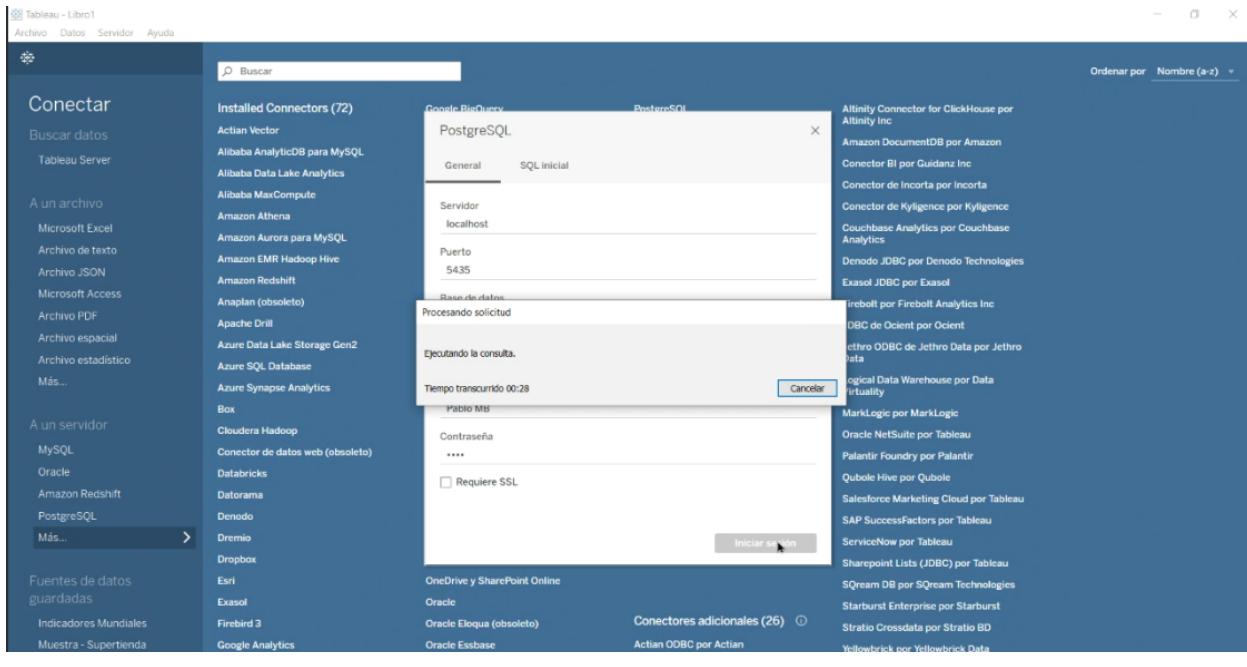
Register - Server

General Connection Parameters SSH Tunnel Advanced

Host name/address	localhost
Port	5435
Maintenance database	postgres
Username	Pablo MB
Kerberos authentication?	<input checked="" type="checkbox"/>
Password
Save password?	<input checked="" type="checkbox"/>
Role	
Service	

i ? X Close Reset Save





This screenshot shows the 'Conecciones' (Connections) pane in Tableau. It lists a single connection named 'localhost - PostgreSQL'. Below the connections, the 'Base de datos' (Database) pane is expanded, showing the 'dvrental' database. Under 'Tablas' (Tables), the 'customer' table is selected. To the right, a large grid icon with the text 'Arrastrar tablas aquí' (Drag tables here) is displayed, indicating where to drop the selected table for analysis.

3. Modelo Multidimensional

3.1. Dimensiones y sus atributos

Dimensión Lugar (Address):

- address_id
- city

- country

Dimensión Película (Film):

- film_id
- title
- category
- actors

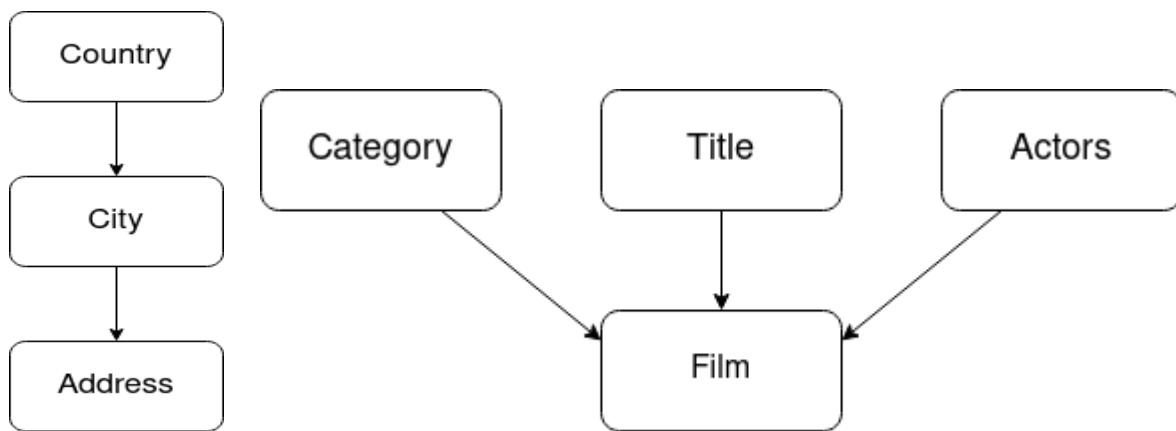
Dimensión Fecha (Date):

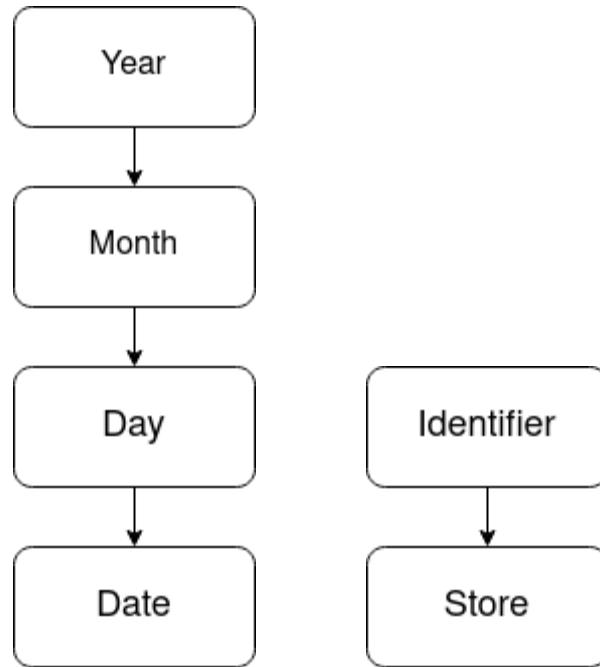
- date_key (formato)
- day
- month
- year

Dimensión Sucursal (Store):

- store_id

Jerarquías de las dimensiones





3.2. Definición de los atributos de cada dimensión

Dimensión	Atributo	Definición	Muestra
Address	address_id	Identificador del lugar	9
	city	Nombre de la ciudad asociada a la dirección	Barcelona
	country	Nombre del país asociado a la dirección	Angola
Film	film_id	Identificador de la película	4
	title	Título de la película	Ace Goldfinger
	category	Nombre de la categoría asociada a la película	Horror
	actors	Nombres de los actores separados por comas asociados a la película	Minnie Zellweger, Chris Depp, Bob Fawcett, Sean Guiness
Date	date_key	Llave que es usada como identificador para la dimensión fecha	2005-05-24
	day	Número que representa	24

		el día que es parte de la fecha almacenada en date_key	
	month	Número que representa el mes que es parte de la fecha almacenada en date_key	05
	year	Número que representa el año que es parte de la fecha almacenada en date_key	2005
Store	store_id	Identificador de la tienda	123

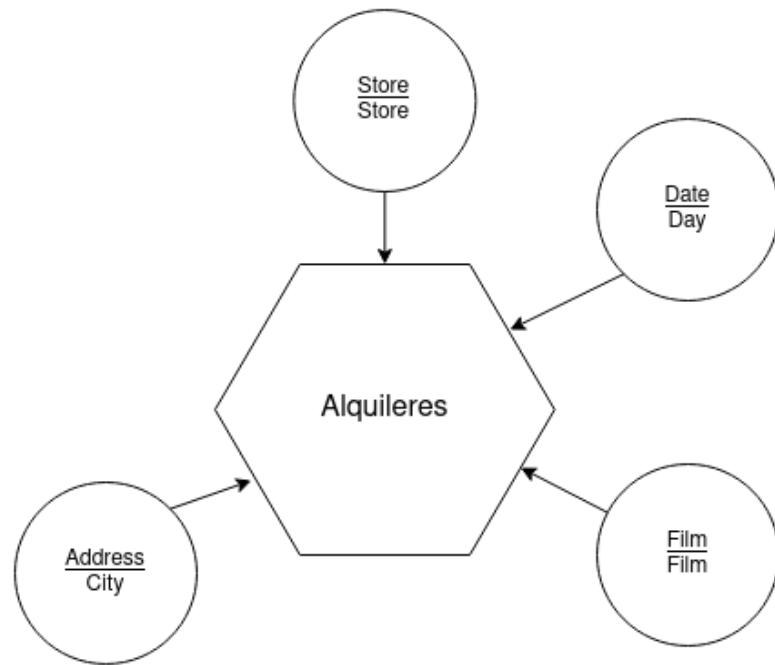
3.3. Medidas y su forma de cálculo

Medida	Forma de cálculo
Número de alquileres	COUNT
Monto total cobrado por alquileres	SUM(cantidad)

3.4. Granularidad de las dimensiones

Dimensión	Granularidad (nivel de detalle)
Date	Day
Film	Film
Store	Store
Address	City

3.5. Esquema estrella



3.6. Fuente de datos para cada atributo de las dimensiones (tabla y atributo origen)

Dimensión	Atributo	Origen
Address	city	city.city
	country	country.country
Film	title	film.title
	category	category.name
	actors	actor.first_name + actor.last_name

Date	date_key	rental.rental_date
Store	store_id	store.store_id

3.7 Creación del modelo en PostgreSQL

- Tabla de hechos

```

6  DROP TABLE rental_facts;
7
8  CREATE TABLE rental_facts(
9      rental_id INTEGER CONSTRAINT id_rental_rf NOT NULL,
10     address_id NUMERIC(10) CONSTRAINT id_address_rf NOT NULL,
11     film_id NUMERIC(10) CONSTRAINT id_film_rf NOT NULL,
12     date_key DATE CONSTRAINT date_key_rf NOT NULL,
13     store_id NUMERIC(10) CONSTRAINT id_store_rf NOT NULL,
14     payment_amount NUMERIC(5,2),
15
16     -- constraints de llaves foráneas de la tabla
17     CONSTRAINT fk_rf_rental_id FOREIGN KEY(rental_id) REFERENCES rental(rental_id),
18     CONSTRAINT fk_rf_address_id FOREIGN KEY(address_id) REFERENCES address_dim(address_id),
19     CONSTRAINT fk_rf_film_id FOREIGN KEY(film_id) REFERENCES film_dim(film_id),
20     CONSTRAINT fk_rf_store_id FOREIGN KEY(store_id) REFERENCES store_dim(store_id),
21     CONSTRAINT fk_rf_date_key FOREIGN KEY(date_key) REFERENCES date_dim(date_key)
22 );
23
24 /* Dimensión Película (film_dim)
```

Data Output Messages Notifications

- Dimensión Film

```

20
21 /* Dimensión Película (film_dim)
22     - film_id numeric
23     - title varchar
24     - category varchar
25     - actors text
26 */
27
28 CREATE TABLE film_dim(
29     film_id NUMERIC(10) NOT NULL,
30     title VARCHAR(100) NOT NULL,
31     category VARCHAR(50) NOT NULL,
32     actors TEXT NOT NULL,
33     -- Constraint de la llave principal de la tabla
34     CONSTRAINT pk_film_dim_id PRIMARY KEY(film_id)
35 );
36
```

- Dimensión Address

```
36
37 /* Dimensión Lugar (address_dim)
38   - address_id numeric
39   - city varchar
40   - country varchar
41 */
42 CREATE TABLE address_dim(
43     address_id NUMERIC(10) NOT NULL,
44     city VARCHAR(50) NOT NULL,
45     country VARCHAR(50) NOT NULL,
46     -- Constraint de la llave principal de la tabla
47     CONSTRAINT pk_address_dim_id PRIMARY KEY(address_id)
48 );
49
```

- Dimensión Date

```
50 /* Dimensión Fecha (date_dim)
51   - date_key date
52   - year numeric
53   - month numeric
54   - day numeric
55 */
56 CREATE TABLE date_dim(
57     date_key DATE NOT NULL,
58     year NUMERIC(5) NOT NULL
59     GENERATED ALWAYS AS (EXTRACT(YEAR FROM date_key)) STORED,
60     month NUMERIC(2) NOT NULL
61     GENERATED ALWAYS AS (EXTRACT(MONTH FROM date_key)) STORED,
62     day NUMERIC(2) NOT NULL
63     GENERATED ALWAYS AS (EXTRACT(DAY FROM date_key)) STORED,
64     -- Constraint de la llave principal de la tabla
65     CONSTRAINT pk_date_dim_id PRIMARY KEY(date_key)
66 );
```

- Dimensión Store

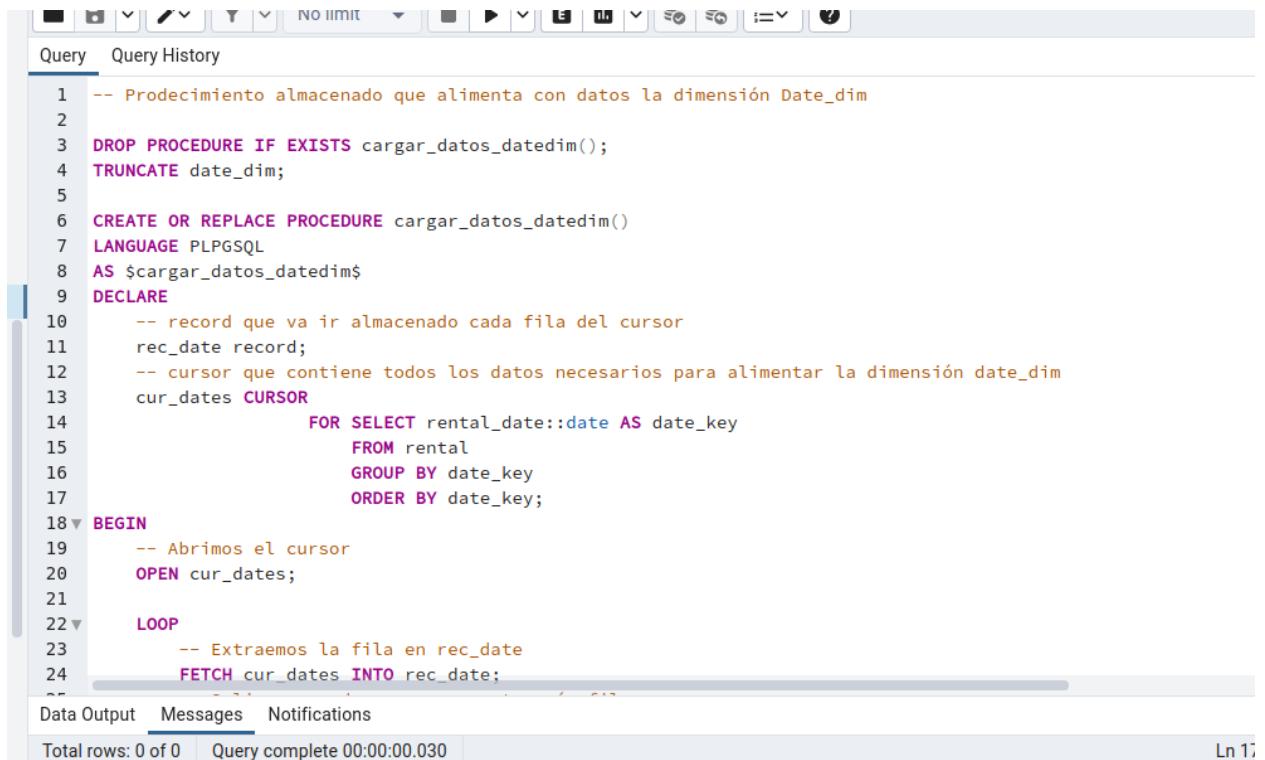
```

73 /* Dimensión Sucursal (store_dim)
74   - store_id numeric
75 */
76 CREATE TABLE store_dim(
77   store_id NUMERIC(10) NOT NULL,
78   -- Constraint de la llave principal de la tabla
79   CONSTRAINT pk_store_dim_id PRIMARY KEY(store_id)
80 );

```

Data Output Messages Notifications

- Procedimientos almacenados para alimentar el modelo multidimensional



The screenshot shows a PostgreSQL query editor interface with a toolbar at the top and a main code area below. The code area contains a stored procedure named 'cargar_datos_datedim'. The procedure starts by dropping the existing procedure if it exists, truncating the 'date_dim' table, and then creating a new one. It defines a cursor 'cur_dates' that selects rental dates from the 'rental' table, grouped by date key and ordered by date key. A loop then opens this cursor and fetches rows into a record 'rec_date' until there are no more rows.

```

1 -- Prodecimiento almacenado que alimenta con datos la dimensión Date_dim
2
3 DROP PROCEDURE IF EXISTS cargar_datos_datedim();
4 TRUNCATE date_dim;
5
6 CREATE OR REPLACE PROCEDURE cargar_datos_datedim()
7 LANGUAGE PLPGSQL
8 AS $cargar_datos_datedim$
9 DECLARE
10    -- record que va ir almacenado cada fila del cursor
11    rec_date record;
12    -- cursor que contiene todos los datos necesarios para alimentar la dimensión date_dim
13    cur_dates CURSOR
14        FOR SELECT rental_date::date AS date_key
15                      FROM rental
16                      GROUP BY date_key
17                      ORDER BY date_key;
18 BEGIN
19    -- Abrimos el cursor
20    OPEN cur_dates;
21
22 LOOP
23    -- Extraemos la fila en rec_date
24    FETCH cur_dates INTO rec_date;
25
26 END LOOP;
27
28 END;
$
```

Total rows: 0 of 0 Query complete 00:00:00.030 Ln 17

Query History

```

23      -- Extraemos la fila en rec_date
24      FETCH cur_dates INTO rec_date;
25      -- Salimos cuando no se encuentre más filas
26      EXIT WHEN NOT FOUND;
27      -- Insertamos cada fila dentro de la dimensión date_dim
28      INSERT INTO date_dim(date_key)
29          VALUES(rec_date.date_key);
30  END LOOP;
31      -- Cerramos el cursor
32      CLOSE cur_dates;
33      -- Commit cambios
34      COMMIT;
35
36 END; $cargar_datos_datedim$
```

-- CALL public.cargar_datos_datedim();
-- select * from date_dim;
-- Prodecimiento almacenado que alimenta con datos la dimensión Film_dim

```

41      DROP PROCEDURE IF EXISTS cargar_datos_filmdim();
42      TRUNCATE TABLE film_dim CASCADE;
43
44 CREATE OR REPLACE PROCEDURE cargar_datos_filmdim()
45 LANGUAGE PLPGSQL
Data Output Messages Notifications
Total rows: 0 of 0 | Query complete 00:00:00.030 | Ln 17

```

Query History

```

43      CREATE OR REPLACE PROCEDURE cargar_datos_filmdim()
44      TRUNCATE TABLE film_dim CASCADE;
45
46 CREATE OR REPLACE PROCEDURE cargar_datos_filmdim()
47 LANGUAGE PLPGSQL
48 AS $cargar_datos_filmdim$
49
50     DECLARE
51         -- record que va ir almacenado cada fila del cursor
52         rec_film record;
53         -- cursor que contiene todos los datos necesarios para alimentar la dimensión film_dim
54         cur_films CURSOR
55             FOR SELECT f.film_id, f.title, c.name AS category, string_agg(CONCAT (a.first_name, ' ', a.last_name), ',') AS actors
56                 FROM actor a
57                 RIGHT JOIN film_actor fa
58                     ON fa.actor_id = a.actor_id
59                 RIGHT JOIN film f
60                     ON f.film_id = fa.film_id
61                 RIGHT JOIN film_category fc
62                     ON fc.film_id = f.film_id
63                 RIGHT JOIN category c
64                     ON c.category_id = fc.category_id
65                 GROUP BY f.film_id, c.name
66                 Order by f.title;
```

BEGIN

-- Abrimos el cursor

```

67      OPEN cur_films;
68
Data Output Messages Notifications
Total rows: 1000 of 16044 | Query complete 00:00:00.048 | Ln 23

```

Query History

```

55
56
57
58
59
60
61
62
63
64
65
66 BEGIN
67     -- Abrimos el cursor
68     OPEN cur_films;
69
70 LOOP
71     -- Extraemos la fila en rec_film
72     FETCH cur_films INTO rec_film;
73     -- Salimos cuando no se encuentre más filas
74     EXIT WHEN NOT FOUND;
75     -- Insertamos cada fila dentro de la dimensión film_dim
76     INSERT INTO film_dim(film_id, title, category, actors)
77     VALUES(rec_film.film_id, rec_film.title, rec_film.category, rec_film.actors);
78 END LOOP;
79     -- Cerramos el cursor
80     CLOSE cur_films;
81     -- Commit cambios
82     COMMIT;
83
84 END; $cargar_datos_filmdim$
85
86 -- CALL public.cargar_datos_filmdim();
87 -- select * from film_dim
88
89 -- Procedimiento almacenado que alimenta con datos dimensión Address_dim

```

Data Output Messages Notifications

Total rows: 0 of 0 | Query complete 00:00:00.030 | Ln 17

Query History

```

88
89 -- Procedimiento almacenado que alimenta con datos dimensión Address_dim
90
91 DROP PROCEDURE IF EXISTS cargar_datos_addressdim();
92 TRUNCATE address_dim;
93
94 CREATE OR REPLACE PROCEDURE cargar_datos_addressdim()
95 LANGUAGE PLPGSQL
96 AS $cargar_datos_addressdim$
97 DECLARE
98     -- record que va ir almacenado cada fila del cursor
99     rec_address record;
100    -- cursor que contiene todos los datos necesarios para alimentar la dimensión address_dim
101    cur_addresses CURSOR
102        FOR SELECT a.address_id, c.city, k.country
103            FROM address a
104                INNER JOIN city c
105                    ON a.city_id = c.city_id
106                    INNER JOIN country k
107                        ON c.country_id = k.country_id;
108 BEGIN
109     -- Abrimos el cursor
110     OPEN cur_addresses;
111
112 LOOP

```

Data Output Messages Notifications

Total rows: 0 of 0 | Query complete 00:00:00.030 | Ln 17

Query History

```

109      -- Abrimos el cursor
110      OPEN cur_addresses;
111
112      LOOP
113          -- Extraemos la fila en rec_address
114          FETCH cur_addresses INTO rec_address;
115          -- Salimos cuando no se encuentre más filas
116          EXIT WHEN NOT FOUND;
117          -- Insertamos cada fila dentro de la dimensión address_dim
118          INSERT INTO address_dim(address_id, city, country)
119              VALUES(rec_address.address_id, rec_address.city, rec_address.country);
120      END LOOP;
121      -- Cerramos el cursor
122      CLOSE cur_addresses;
123      -- Commit cambios
124      COMMIT;
125
126  END; $cargar_datos_addressdim$
127
128  -- CALL public.cargar_datos_addressdim();
129  -- select * from address_dim;
130
131  -- Procedimiento almacenado para alimentar la Dimensión Store_dim
132  DROP PROCEDURE IF EXISTS load_store_dim_date();
133  TRUNCATE store_dim;

```

Data Output Messages Notifications

Total rows: 0 of 0 Query complete 00:00:00.030 Ln 17

Query History

```

132  DROP PROCEDURE IF EXISTS load_store_dim_date();
133  TRUNCATE store_dim;
134
135  CREATE OR REPLACE PROCEDURE load_store_dim_data()
136      LANGUAGE PLPGSQL
137      AS
138      $store_dim_data$
139      DECLARE
140          --record para almacenar los datos extraidos
141          rec_store RECORD;
142          --cursor que extrae los datos de la tabla store
143          cur_store CURSOR
144              FOR SELECT store_id::NUMERIC AS store_id
145                  FROM store
146                  GROUP BY store_id
147                  ORDER BY store_id;
148      BEGIN
149          --abrimos el cursor
150          OPEN cur_store;
151      LOOP
152          --guardar los registros en el record
153          FETCH cur_store INTO rec_store;
154          --salimos cuando no encuentre más datos
155          EXIT WHEN NOT FOUND;

```

Data Output Messages Notifications

Total rows: 0 of 0 Query complete 00:00:00.030 Ln 17

Query History

```

154      --sacamos cuando no encuentre mas datos
155      EXIT WHEN NOT FOUND;
156      --insertamos los datos dentro de la tabla de la dimensión
157      INSERT INTO store_dim(store_id)
158      VALUES(rec_store.store_id);
159  END LOOP;
160  --cerramos el cursor
161  CLOSE cur_store;
162  --guardamos los cambios
163  COMMIT;
164 END;
$store_dim_data$
```

-- CALL public.load_store_dim_data();
-- SELECT * FROM store_dim;

-- Procedimiento almacenado que alimenta la tabla de hechos
DROP PROCEDURE IF EXISTS cargar_datos_tabla_hechos();
TRUNCATE TABLE rental_facts;

CREATE OR REPLACE PROCEDURE cargar_datos_tabla_hechos()
LANGUAGE PLPGSQL
AS \$cargar_datos_tablahechos\$
DECLARE
 -- Declaración de variables locales

Data Output Messages Notifications

Total rows: 0 of 0 | Query complete 00:00:00.030 | Ln 17

Query History

```

175 LANGUAGE PLPGSQL
176 AS $cargar_datos_tablahechos$
177 DECLARE
178     -- Declaración de variables locales
179     v_film_id NUMERIC(10);
180     v_store_id NUMERIC(10);
181     v_address_id NUMERIC(10);
182     v_amount NUMERIC(5,2);
183     rec_inventory RECORD;
184     rec_rent RECORD;
185     -- Cursor que consulta todos los alquileres
186     cur_rents CURSOR
187         FOR SELECT rental_id, rental_date, inventory_id, customer_id
188             FROM rental;
189 BEGIN
190     -- Abrimos el cursor de rental
191     OPEN cur_rents;
192
193     -- Extraemos los registros obtenidos
194 LOOP
195     FETCH cur_rents INTO rec_rent;
196     EXIT WHEN NOT FOUND;
197
198     -- Obtenemos el film_id y store_id por medio del inventory_id
```

Data Output Messages Notifications

Total rows: 0 of 0 | Query complete 00:00:00.030 | Ln 17

Query History

```

197      -- Obtenemos el film_id y store_id por medio del inventory_id
198      SELECT film_id, store_id
199      FROM inventory
200      INTO rec_inventory
201      WHERE inventory_id = rec_rent.inventory_id;
202
203      -- Guardamos los ids en las variables correspondientes
204      v_film_id := rec_inventory.film_id;
205      v_store_id := rec_inventory.store_id;
206
207      -- Obtenemos el address_id por medio del customer_id
208      SELECT address_id
209      FROM customer
210      INTO v_address_id
211      WHERE customer_id = rec_rent.customer_id;
212
213      -- Obtenemos el payment amount por medio del rental_id
214      SELECT amount
215      FROM payment
216      INTO v_amount
217      WHERE rental_id = rec_rent.rental_id;
218
219      -- Insertamos los datos en la tabla de hechos
220      INSERT INTO rental_facts(rental_id, address_id, film_id, date_key, store_id, payment_amount)
221
Data Output Messages Notifications
Total rows: 0 of 0 Query complete 00:00:00.030 Ln 17

```

Query History

```

212      WHERE customer_id = rec_rent.customer_id,
213
214      -- Obtenemos el payment amount por medio del rental_id
215      SELECT amount
216      FROM payment
217      INTO v_amount
218      WHERE rental_id = rec_rent.rental_id;
219
220      -- Insertamos los datos en la tabla de hechos
221      INSERT INTO rental_facts(rental_id, address_id, film_id, date_key, store_id, payment_amount)
222      VALUES(rec_rent.rental_id, v_address_id, v_film_id, rec_rent.rental_date::date, v_store_id, v_amount
223      END LOOP;
224      -- Cerramos el cursor
225      CLOSE cur_rents;
226      -- Guardamos los cambios
227      COMMIT;
228  END; $cargar_datos_tablahechos$
229
230  CALL cargar_datos_tabla_hechos();
231  TRUNCATE TABLE rental_facts;
232
233  select * from rental_facts;
234
235
Data Output Messages Notifications
Total rows: 0 of 0 Query complete 00:00:00.030 Ln 17

```

4. Visualización del modelo multidimensional por medio de Tableau

Tableau - Libro1

Archivo Datos Servidor Ayuda

Conectar

Buscar datos

Tableau Server

A un archivo

Microsoft Excel

Archivo de texto

Archivo JSON

Microsoft Access

Archivo PDF

Archivo espacial

Archivo estadístico

Más...

A un servidor

MySQL

Oracle

Amazon Redshift

PostgreSQL

Más...

Fuentes de datos guardadas

Indicadores Mundiales

Muestra - Supertienda

Sample - Superstore

PostgreSQL

General SQL inicial

Servidor localhost

Puerto 5435

Base de datos dvrental

Autenticación Nombre de usuario y contraseña

Nombre de usuario Pablo MB

Contraseña ****

Requiere SSL

Iniciar sesión

Aceleradores

Supertienda Regional Indicadores Mundiales

Descubrir

Capacitación

Introducción

Conectarse a los datos

Análisis visual

Funcionamiento de Tableau

Más videos de capacitación...

Recursos

Obtener Tableau Prep

Evaluación del Blueprint de Tableau

Foros de la comunidad de Tableau

Aceleradores de Tableau

Blog Leer la publicación más reciente

Más aceleradores

Tableau 2023.1 disponible ahora Acelere su análisis sin problemas Explorar ahora →

Tableau - Libro1

Archivo Datos Servidor Ventana Ayuda

Conexiones localhost PostgreSQL

Base de datos dvrental

Tabla

- actor
- actor_info
- address
- address_dim
- category
- city
- country
- customer
- customer_list
- date_dim
- film
- film_actor
- film_category
- film_dim
- film_list
- inventory
- language
- Nueva SQL personalizada
- Nueva unión de filas
- Nueva extensión de tabla

rental_facts+ (dvrental)

rental_facts — film_dim

En que se diferencian las relaciones de la unión de columnas? Más información

#	film_dim	film_dim	film_dim	film_dim
	Film Id (Film Dim)	Title	Category	Actors
rental_facts	Operator	film_dim		
# Film Id	=	# Film Id (Film Dir		

Añadir más campos

Opciones de rendimiento

Filtros 0 | Añadir

Conexión En tiempo real Extraer

Filas

Actualizar ahora

Actualizar automáticamente

Fuente de datos Hoja 1

