

Instituto Tecnológico de Costa Rica
Centro Académico de Alajuela
Bases de Datos II

Proyecto II
Replicación y análisis de datos

Grupo 8:
Marco Rodríguez Vargas – Carné: 2022149445
Kevin Carranza Jiménez – Carné: 2015100260
Jorge Esteban Benavides Castro – Carné: 2022230697

Profesor:
Alberto Shum Chan

Fecha de entrega:
19 de octubre. 2023

II Semestre, 2023

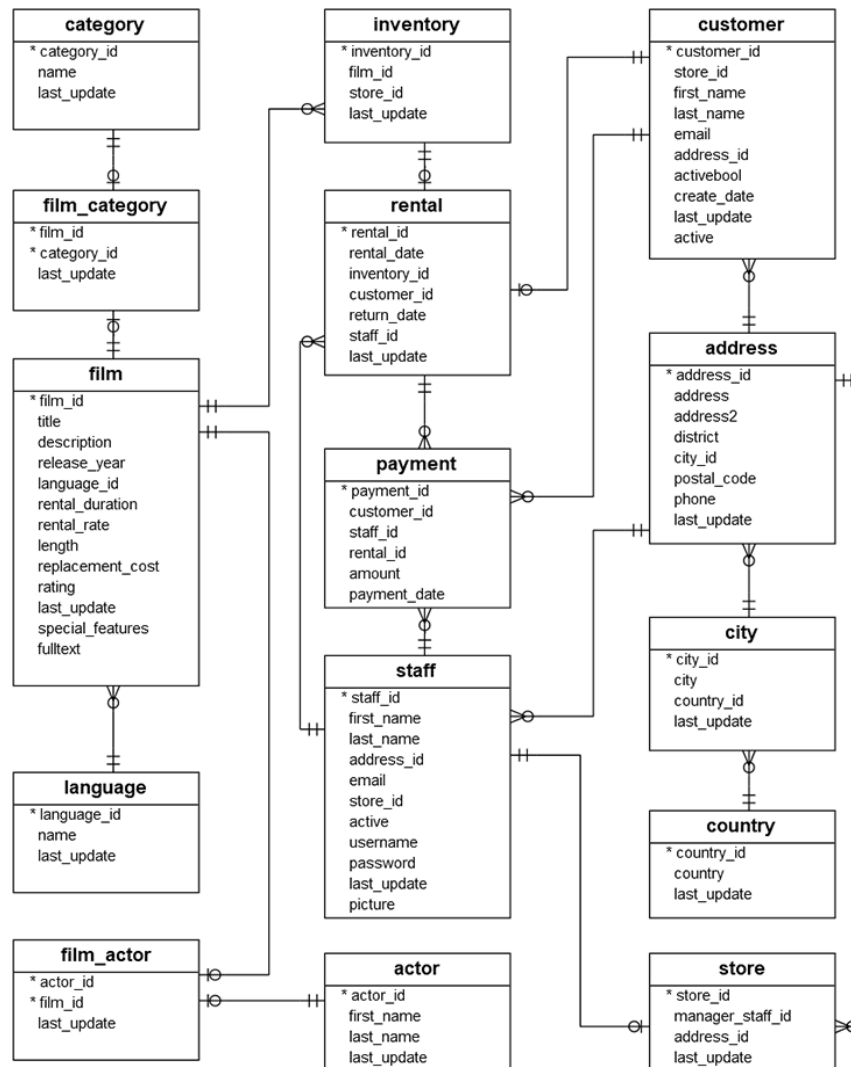
Introducción

El proyecto de inteligencia de negocios se compone de varios elementos fundamentales. Comienza con la restauración de una base de datos para el alquiler de películas ya existente, seguido de la instanciación de una base de datos esclava con una réplica de la base de datos que se actualiza con la base de datos maestra. Por otra parte, el proyecto requiere de procedimientos almacenados, usuarios y roles. Los procedimientos almacenados deben poder insertar un nuevo cliente, registrar un alquiler, registrar una devolución y buscar una película. Además, requiere implementar un datamart, alimentado por los datos de la instancia esclava a través del proceso de replicación. Finalmente, se emplea el software Tableau para diseñar un dashboard interactivo que resume información importante a partir de los datos almacenados en el datamart.

Descripción del proyecto

1 - Base de datos para alquiler de películas

Se usará la base de datos descrita en la página PostgreSQL DVD rental Sample Database.



Se deben crear las siguientes funciones o procedimientos almacenados que afecten el sistema transaccional, es decir las tablas originales en la instancia maestra:

- insertar un nuevo cliente
- registrar un alquiler
- registrar una devolución
- buscar una película

Los procedimientos deben de estar debidamente documentados. Debe incluir una descripción, descripción de parámetros, descripción de salida y descripción de bloques relevantes.

En cuanto a Seguridad se deben crear los siguientes roles:

- EMP: solo tiene el derecho de ejecutar los siguientes procedimientos almacenados; no puede leer ni actualizar ningún objeto de la base de datos

- registrar un alquiler
- registrar una devolución
- buscar una película
- ADMIN: tiene el derecho de un empleado más el derecho de ejecutar los siguientes procedimientos almacenados; no puede leer ni actualizar ningún objeto de la base de datos
 - insertar un nuevo cliente

Se deben crear los siguientes usuarios:

- video: no login, dueño de todas las tablas y de todos los procedimientos
- empleado1: un usuario con rol EMP
- administrador1: un usuario con rol ADMIN

Seguridad en procedimientos almacenados: los procedimientos almacenados deben correr usando las credenciales de su dueño, video.

2 - Replicación

Se debe establecer una réplica o instancia esclava de la base de datos. No es necesario que la réplica esté en una máquina distinta. Para efectos del proyecto, la réplica puede estar en otra instancia en la misma estación de trabajo.

3 - Modelo multidimensional

Con el fin de analizar la información acumulada de los alquileres se debe implementar un modelo multidimensional e implementar un dashboard para consultar dicho modelo.

Las medidas de interés son:

- número de alquileres
- monto total cobrado por alquileres

Las dimensiones de interés son:

- Película (Film): jerarquía de categoría, filme incluyendo actores.
- Lugar (Address): jerarquía de país y ciudad
- Fecha (Rental): jerarquía de año, mes y día
- Sucursal (Store)

Se deben desarrollar procedimientos almacenados para alimentar con datos el modelo multidimensional a partir de la réplica de la base de datos de alquileres. Todos los procedimientos y funciones almacenadas deben contener documentación completa.

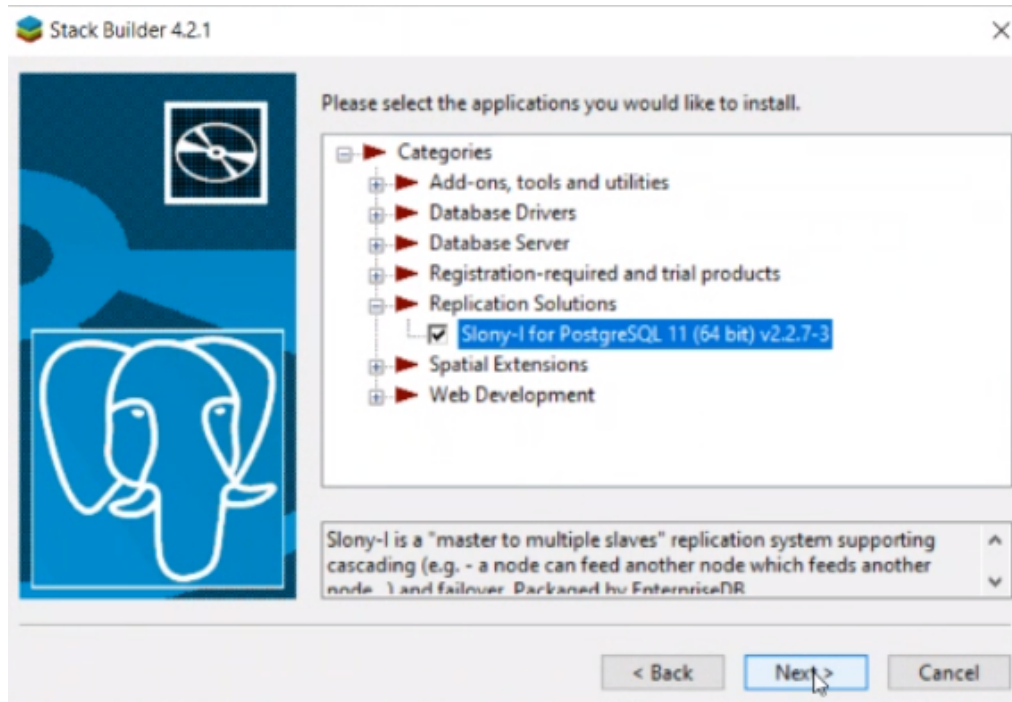
4 - Visualización y acceso por medio de interfaz gráfica al modelo estrella

Se debe diseñar e implementar un dashboard que resuma información importante sobre los datos utilizando el software Tableau. Las consultas que debe solventar el dashboard se listan a continuación:

- Para una sucursal (a seleccionar por el usuario), grafique el número de alquileres realizados y el monto cobrado por mes, sin importar el año-
- Graficar para un año (a seleccionar por el usuario) los montos cobrados por alquileres por mes.
- Para una categoría de película (a seleccionar por el usuario), graficar el número de alquileres y el monto cobrado por año.
- Para los 10 actores con más alquileres, graficar los montos totales de alquileres por año (a seleccionar por el usuario). Incluya la opción de todos los años.
- Despliegue un mapa de ciudades que presente por año el monto de alquiler total representado por el tamaño del punto sobre la ciudad.

Proceso de replicación

A la hora de instalar postgresql hay que incluir la instalación de Slony I



Dentro de C:\Program Files\PostgreSQL\11\data hay que editar el archivo pg_hba.conf. Esto es para establecer el IP de la replicación maestra y esclava, como en este caso es local entonces el IP es localhost (127.0.0.1/32). El 32 al final es del puerto 5432 de PostgreSQL.

#	TYPE	DATABASE	USER	ADDRESS	METHOD
#Maestro					
host	all	all	127.0.0.1/32	md5	
#Esclavo					
host	all	all	127.0.0.1/32	md5	
# IPv4 local connections:					
host	all	all	127.0.0.1/32	md5	
# IPv6 local connections:					
host	all	all	:::1/128	md5	
# Allow replication connections from localhost, by a user with the					
# replication privilege.					
host	replication	all	127.0.0.1/32	md5	
host	replication	all	:::1/128	md5	

```

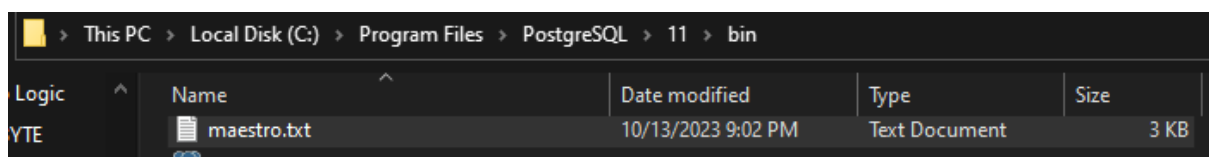
pg_hba.txt - Notepad
File Edit Format View Help
# "scram-sha-256" are preferred since they send encrypted passwords.
#
# OPTIONS are a set of options for the authentication in the format
# NAME=VALUE. The available options depend on the different
# authentication methods -- refer to the "Client Authentication"
# section in the documentation for a list of which options are
# available for which authentication methods.
#
# Database and user names containing spaces, commas, quotes and other
# special characters must be quoted. Quoting one of the keywords
# "all", "sameuser", "samerole" or "replication" makes the name lose
# its special character, and just match a database or username with
# that name.
#
# This file is read on server startup and when the server receives a
# SIGHUP signal. If you edit the file on a running system, you have to
# SIGHUP the server for the changes to take effect, run "pg_ctl reload",
# or execute "SELECT pg_reload_conf()".
#
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.


# TYPE DATABASE USER ADDRESS METHOD
#-----
#Maestro
host all all 127.0.0.1/32 md5
#Esclavo
host all all 127.0.0.1/32 md5
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
host replication all 127.0.0.1/32 md5
host replication all ::1/128 md5

```

Dentro de C:\Program Files\PostgreSQL\11\bin hay que crear 2 archivos .txt, estos van a tener las instrucciones para que Slony pueda realizar la replicación. El maestro se crea con el nombre de maestro.txt y el esclavo se crea con el nombre de esclavo.txt.

Maestro.txt



El cual debe contener estas instrucciones

```
# Define el cluster
cluster name=clusterDVD;
```

```
node 1 admin conninfo='dbname=dvdrental host=localhost user=postgres
password=rootroot';
```

```
node 2 admin conninfo='dbname=dvdrental_replica host=localhost user=postgres
password=rootroot';
```

```
init cluster (id=1, comment='Master Node');
```

```
create set (id=1, origin=1, comment='mis tablas de dvd');
```

```
set add table (set id=1, origin=1, id=1, fully qualified name='public.actor',
comment='actor');
set add table (set id=1, origin=1, id=2, fully qualified name='public.address',
comment='address');
set add table (set id=1, origin=1, id=3, fully qualified name='public.category',
comment='category');
set add table (set id=1, origin=1, id=4, fully qualified name='public.city',
comment='city');
set add table (set id=1, origin=1, id=5, fully qualified name='public.country',
comment='country');
set add table (set id=1, origin=1, id=6, fully qualified name='public.customer',
comment='customer');
set add table (set id=1, origin=1, id=7, fully qualified name='public.film',
comment='film');
set add table (set id=1, origin=1, id=8, fully qualified name='public.film_actor',
comment='film_actor');
set add table (set id=1, origin=1, id=9, fully qualified name='public.film_category',
comment='film_category');
set add table (set id=1, origin=1, id=10, fully qualified name='public.inventory',
comment='inventory');
set add table (set id=1, origin=1, id=11, fully qualified name='public.language',
comment='language');
set add table (set id=1, origin=1, id=12, fully qualified name='public.payment',
comment='payment');
set add table (set id=1, origin=1, id=13, fully qualified name='public.rental',
comment='rental');
set add table (set id=1, origin=1, id=14, fully qualified name='public.staff',
comment='staff');
set add table (set id=1, origin=1, id=15, fully qualified name='public.store',
comment='store');
```

```
store node (id=2, comment='Nodo Esclavo DVD', EVENT NODE=1);
```



```
store path (server=1, client=2, conninfo='dbname=dvdrental host=localhost  
user=postgres password=rootroot');
```

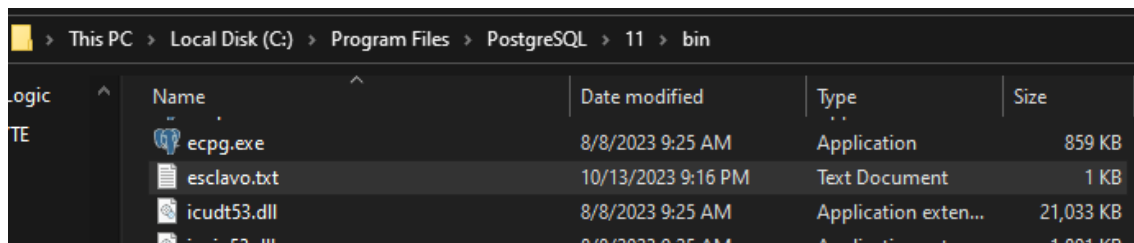
```
store path (server=2, client=1, conninfo='dbname=dvdrental_replica host=localhost  
user=postgres password=rootroot');
```

```
store listen (origin=1, provider=1, receiver=2);
```

```
store listen (origin=2, provider=2, receiver=1);
```

```
SUBSCRIBE SET (ID=1, PROVIDER=1, RECEIVER=2, FORWARD=YES);  
WAIT FOR EVENT (ORIGIN=1, CONFIRMED=ALL, WAIT ON=1);
```

Esclavo.txt



El cual debe contener estas instrucciones

```
# Define el cluster
```

```
cluster name = clusterDVD;
```

```
node 1 admin conninfo='dbname=dvdrental host=localhost user=postgres  
password=rootroot';
```

```
node 2 admin conninfo='dbname=dvdrental_replica host=localhost user=postgres  
password=rootroot';
```

```
SUBSCRIBE SET (ID = 1, PROVIDER = 1, RECEIVER = 2, FORWARD = YES);
```

Es importante tener en cuenta que dependiendo de la máquina hay que hacer modificaciones según sea necesario. El nombre del cluster puede tener cualquier nombre siempre y cuando sea el mismo en ambos .txt. La información dentro de 'conninfo' del nodo 1 será utilizada para el nodo maestro y debe tener el nombre de la base de datos, host desde el cual se conecta (localhost o IP en caso de ser conexiones remotas) y las credenciales (user y password). La información dentro de 'conninfo' del nodo 2 será utilizada para el nodo esclavo y debe tener el nombre de la base de datos, host desde el cual se conecta (localhost o IP en caso de ser conexiones remotas) y las credenciales (user y password).

Una vez se termine la configuración entonces dentro de la terminal del sistema operativo en este caso Windows se ejecutarán ciertos comandos para hacer la conexión necesaria entre los nodos para la replicación.

Terminal 1 se abre en C:\Program Files\PostgreSQL\11\bin

El comando a ejecutar es: `slonik maestro.txt`

Una vez iniciado esperará la confirmación del nodo 2

[illegible]

Terminal 2 se abre en C:\Program Files\PostgreSQL\11\bin

El comando a ejecutar es: slonik esclavo.txt

Una vez iniciado esperará la confirmación del nodo 1

[illegible]

Terminal 3 se abre en C:\Program Files\PostgreSQL\11\bin

El comando a ejecutar es: slon clusterDVD "dbname=dvdrental host=localhost user=postgres password=rootroot".

```
Command Prompt - slon clusterDVD "dbname=dvdrental host=localhost user=postgres password=rootroot"
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>cd C:\Program Files\PostgreSQL\11\bin

C:\Program Files\PostgreSQL\11\bin>slon clusterDVD "dbname=dvdrental host=localhost user=postgres password=rootroot"
2023-10-19 19:32:44 Central America Standard Time CONFIG main: slon version 2.2.7 starting up
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option vac_frequency = 3
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option log_level = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option sync_interval = 2000
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option sync_interval_timeout = 10000
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option sync_group_maxsize = 20
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option quit_sync_provider = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option remote_listen_timeout = 300
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option monitor_interval = 500
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option explain_interval = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option tcp_keepalive_idle = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option tcp_keepalive_interval = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option tcp_keepalive_count = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Integer option apply_cache_size = 100
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option log_pid = 0
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option log_timestamp = 1
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option tcp_keepalive = 1
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option monitor_threads = 1
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option enable_version_check = 1
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Boolean option remote_listen_serializable_transactions = 1
2023-10-19 19:32:44 Central America Standard Time CONFIG main: Real option real_placeholder = 0.000000
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option cluster_name = clusterDVD
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option conn_info = dbname=dvdrental host=localhost user=postgres password=rootroot
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option pid_file = [NULL]
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option log_timestamp_format = %Y-%m-%d %H:%M:%S %Z
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option archive_dir = [NULL]
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option sql_on_connection = [NULL]
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option lag_interval = [NULL]
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option command_on_logarchive = [NULL]
2023-10-19 19:32:44 Central America Standard Time CONFIG main: String option cleanup_interval = 10 minutes
2023-10-19 19:32:44 Central America Standard Time CONFIG main: local node id = 1
2023-10-19 19:32:44 Central America Standard Time INFO main: main process started
2023-10-19 19:32:44 Central America Standard Time CONFIG main: launching sched_start_mainloop
```

Terminal 4 se abre en C:\Program Files\PostgreSQL\11\bin

El comando a ejecutar es: slon clusterDVD "dbname=dvdrental_replica host=localhost user=postgres password=rootroot".

```
Command Prompt - slon clusterDVD "dbname=dvdrental_replica host=localhost user=postgres password=rootroot"
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>cd C:\Program Files\PostgreSQL\11\bin

C:\Program Files\PostgreSQL\11\bin>slon clusterDVD "dbname=dvdrental_replica host=localhost user=postgres password=rootroot"
2023-10-19 19:33:49 Central America Standard Time CONFIG main: slon version 2.2.7 starting up
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option vac_frequency = 3
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option log_level = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option sync_interval = 2000
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option sync_interval_timeout = 10000
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option sync_group_maxsize = 20
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option quit_sync_provider = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option remote_listen_timeout = 300
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option monitor_interval = 500
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option explain_interval = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option tcp_keepalive_idle = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option tcp_keepalive_interval = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option tcp_keepalive_count = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Integer option apply_cache_size = 100
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option log_pid = 0
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option log_timestamp = 1
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option tcp_keepalive = 1
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option monitor_threads = 1
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option enable_version_check = 1
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Boolean option remote_listen_serializable_transactions = 1
2023-10-19 19:33:49 Central America Standard Time CONFIG main: Real option real_placeholder = 0.000000
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option cluster_name = clusterDVD
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option conn_info = dbname=dvdrental_replica host=localhost user=postgres password=rootroot
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option pid_file = [NULL]
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option log_timestamp_format = %Y-%m-%d %H:%M:%S %Z
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option archive_dir = [NULL]
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option sql_on_connection = [NULL]
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option lag_interval = [NULL]
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option command_on_logarchive = [NULL]
2023-10-19 19:33:49 Central America Standard Time CONFIG main: String option cleanup_interval = 10 minutes
2023-10-19 19:33:49 Central America Standard Time CONFIG main: local node id = 2
2023-10-19 19:33:49 Central America Standard Time INFO main: main process started
2023-10-19 19:33:49 Central America Standard Time CONFIG main: launching sched_start_mainloop
2023-10-19 19:33:49 Central America Standard Time CONFIG main: loading current cluster configuration
2023-10-19 19:33:49 Central America Standard Time CONFIG storeNode: no id=1 no comment='Master Node'
2023-10-19 19:33:49 Central America Standard Time CONFIG storePath: pa_server=1 pa_client=2 pa_conninfo="dbname=dvdrental host=localhost user=postgres password=rootroot" pa_connretry=10
```

Mostrar funcionalidad de la replicación

En la base de datos dvdrental haremos un select para ver todos los actores, en este caso son 200 actores.

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, including the 'dvdrental' database and its 'actor' table. The main pane shows a query window with the following SQL query:

```
1 select * from actor;
```

The Data Output pane displays the results of the query, showing 200 rows of actor data. The columns are: actor_id [PK] integer, first_name character varying (45), last_name character varying (45), and last_update timestamp without time zone. The status bar at the bottom indicates 'Total rows: 200 of 200' and 'Query complete 00:00:00.053'.

actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
186	Julia	Zellweger	2013-05-26 14:47:57.62
187	Renee	Ball	2013-05-26 14:47:57.62
188	Rock	Dukakis	2013-05-26 14:47:57.62
189	Cuba	Birch	2013-05-26 14:47:57.62
190	Audrey	Bailey	2013-05-26 14:47:57.62
191	Gregory	Gooding	2013-05-26 14:47:57.62
192	John	Suvari	2013-05-26 14:47:57.62
193	Burt	Temple	2013-05-26 14:47:57.62
194	Meryl	Allen	2013-05-26 14:47:57.62
195	Jayne	Silverstone	2013-05-26 14:47:57.62
196	Bela	Walken	2013-05-26 14:47:57.62
197	Reese	West	2013-05-26 14:47:57.62
198	Mary	Kielty	2013-05-26 14:47:57.62
199	Julia	Fawcett	2013-05-26 14:47:57.62
200	Thora	Temple	2013-05-26 14:47:57.62

De la misma manera si se ejecuta un select para ver todos los actores en la réplica se comprueba que igualmente son 200 actores.

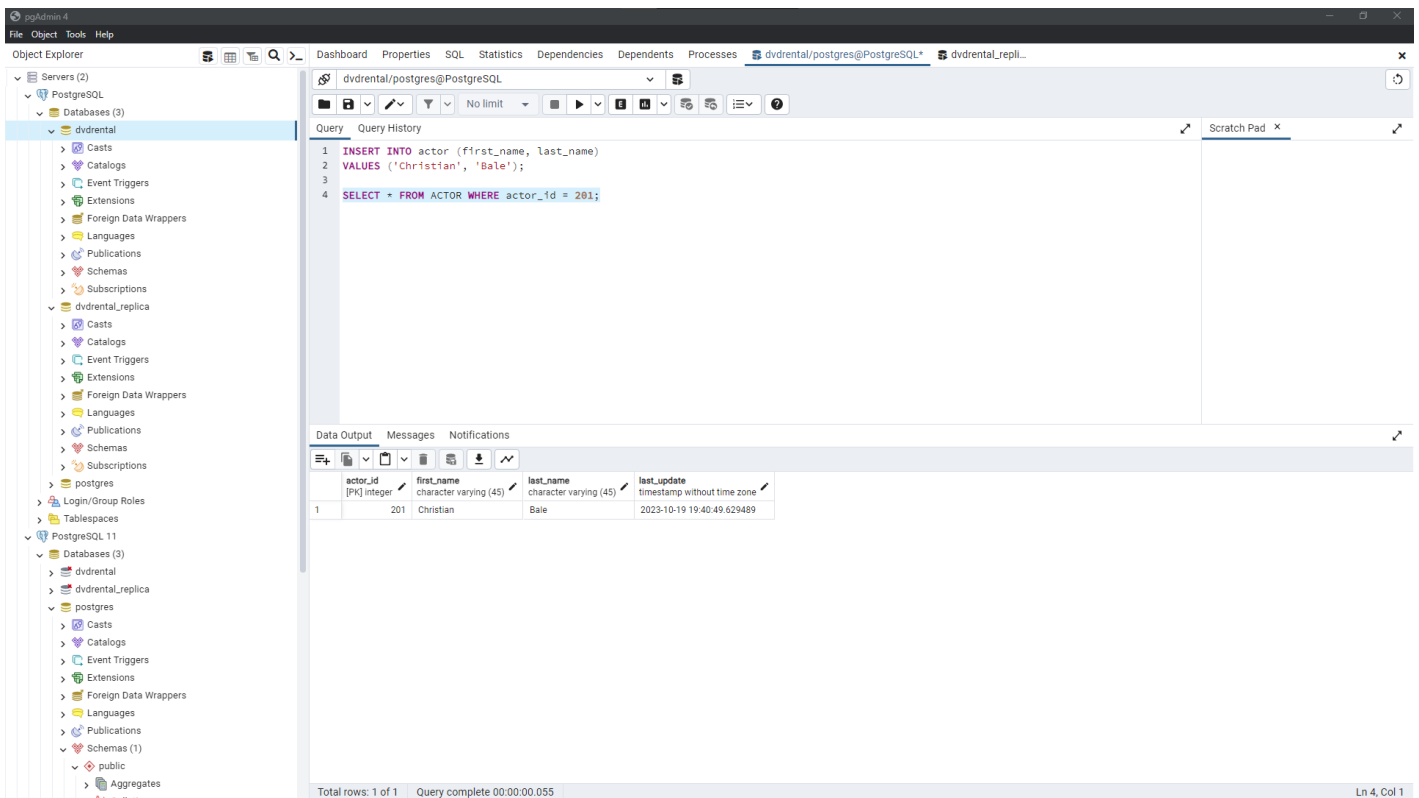
The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, including the 'dvdrental_replica' database and its 'actor' table. The main pane shows a query window with the following SQL query:

```
1 SELECT * FROM ACTOR;
```

The Data Output pane displays the results of the query, showing 201 rows of actor data. The columns are: actor_id [PK] integer, first_name character varying (45), last_name character varying (45), and last_update timestamp without time zone. The status bar at the bottom indicates 'Total rows: 201 of 201' and 'Query complete 00:00:00.054'.

actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
187	Julia	Zellweger	2013-05-26 14:47:57.62
188	Renee	Ball	2013-05-26 14:47:57.62
189	Rock	Dukakis	2013-05-26 14:47:57.62
190	Cuba	Birch	2013-05-26 14:47:57.62
191	Audrey	Bailey	2013-05-26 14:47:57.62
192	Gregory	Gooding	2013-05-26 14:47:57.62
193	John	Suvari	2013-05-26 14:47:57.62
194	Burt	Temple	2013-05-26 14:47:57.62
195	Meryl	Allen	2013-05-26 14:47:57.62
196	Jayne	Silverstone	2013-05-26 14:47:57.62
197	Bela	Walken	2013-05-26 14:47:57.62
198	Reese	West	2013-05-26 14:47:57.62
199	Mary	Kielty	2013-05-26 14:47:57.62
200	Julia	Fawcett	2013-05-26 14:47:57.62
201	Thora	Temple	2013-05-26 14:47:57.62

Ahora haremos un insert de un nuevo actor, verificamos el insert con un select de la tabla actor.



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'actor' table in the 'public' schema. The main window shows the SQL query editor with the following query:

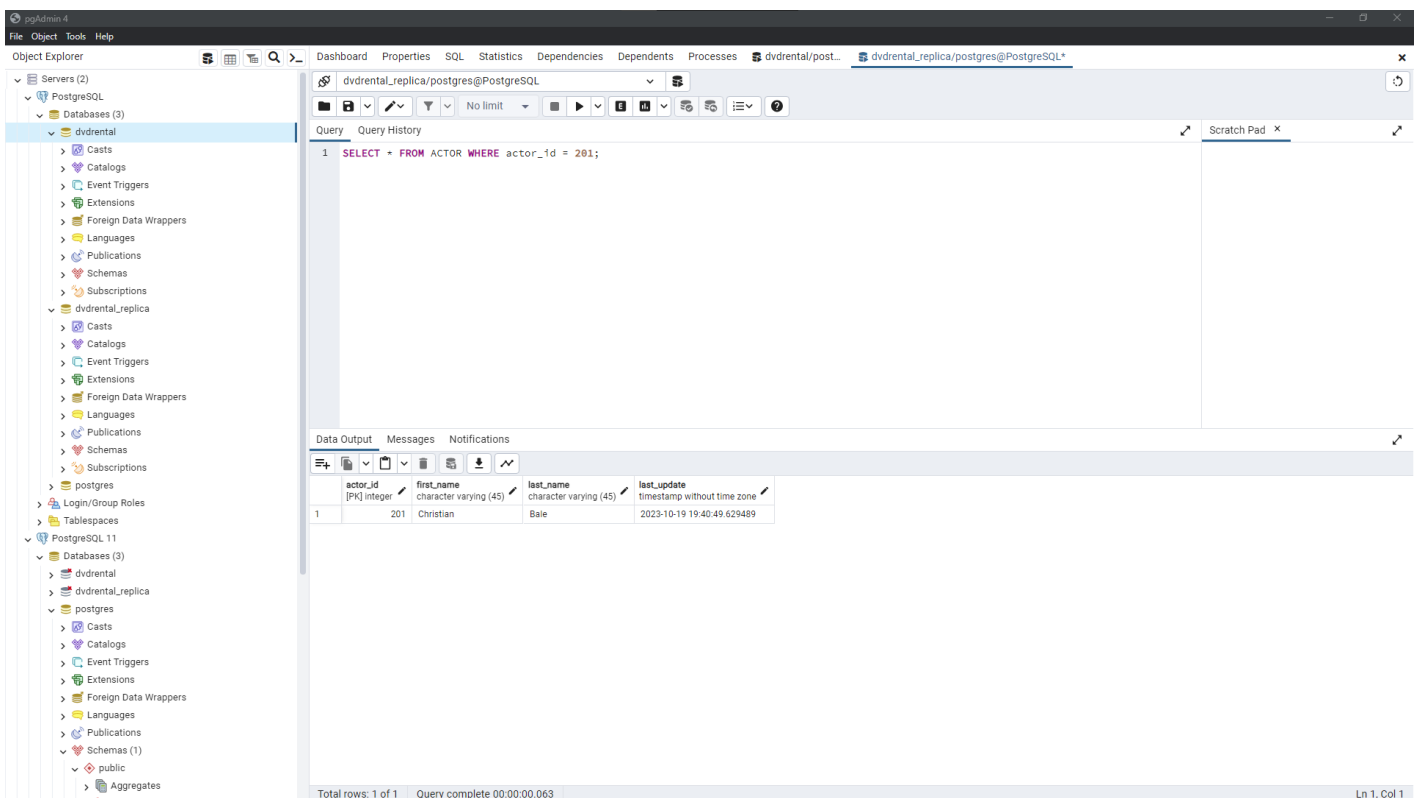
```
1 INSERT INTO actor (first_name, last_name)
2 VALUES ('Christian', 'Bale');
3
4 SELECT * FROM ACTOR WHERE actor_id = 201;
```

The Data Output tab shows the result of the query, displaying a single row with the following data:

actor_id	first_name	last_name	last_update
201	Christian	Bale	2023-10-19 19:40:49.629489

Total rows: 1 of 1 Query complete 00:00:00.055 Ln 4, Col 1

Verificamos que la réplica se haya actualizado con un select en la tabla actor.



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'actor' table in the 'public' schema. The main window shows the SQL query editor with the following query:

```
1 SELECT * FROM ACTOR WHERE actor_id = 201;
```

The Data Output tab shows the result of the query, displaying a single row with the following data:

actor_id	first_name	last_name	last_update
201	Christian	Bale	2023-10-19 19:40:49.629489

Total rows: 1 of 1 Query complete 00:00:00.063 Ln 1, Col 1

Modelo Tridimensional

Dimensiones

Dimensión Film

Nombre del atributo	Definición	Valores de ejemplo
Id_film	Un número entero único para cada registro de la tabla.	1, 43
Title	Contiene el nombre de la película	Airport Pollock, Harry Potter
Movie_category	Contiene la categoría de la película	Horror, Animation

Dimensión Actor

Nombre del atributo	Definición	Valores de ejemplo
Id_actor	Un número entero único para cada registro en la tabla.	11, 13
Actor_first_name	Contiene el nombre del actor o actriz	Nathaly, Robert
Actor_last_name	Contiene el primer apellido del actor o actriz	Portman, Downey

Dimensión Film_Actor

Nombre del atributo	Definición	Valores de ejemplo
Id_film_actor	Un número entero único para cada registro de la tabla.	3, 6
Id_film	Contiene el id_film de la dimensión film	1, 43
Id_actor	Contiene el id_actor de la dimensión film	11, 13

Dimensión Rental

Nombre del atributo	Definición	Valores de ejemplo
Rental_id	Identificador único para cada registro de la tabla.	46, 29
Rental_date	Guarda la fecha y marca de tiempo en la que se hizo un alquiler	2005-07-15 03:13:00, 2006-02-28 01:11:30

Dimensión Address

Nombre del atributo	Definición	Valores de ejemplo
City_id	Identificador único para cada registro de la tabla.	21, 14
Country_name	Nombre del país	China, Afganistán
City_name	Nombre de la ciudad	Hong Kong, Kabul

Dimensión Store

Nombre del atributo	Definición	Valores de ejemplo
Store_id	Un número entero único para cada registro de la tabla.	1, 2

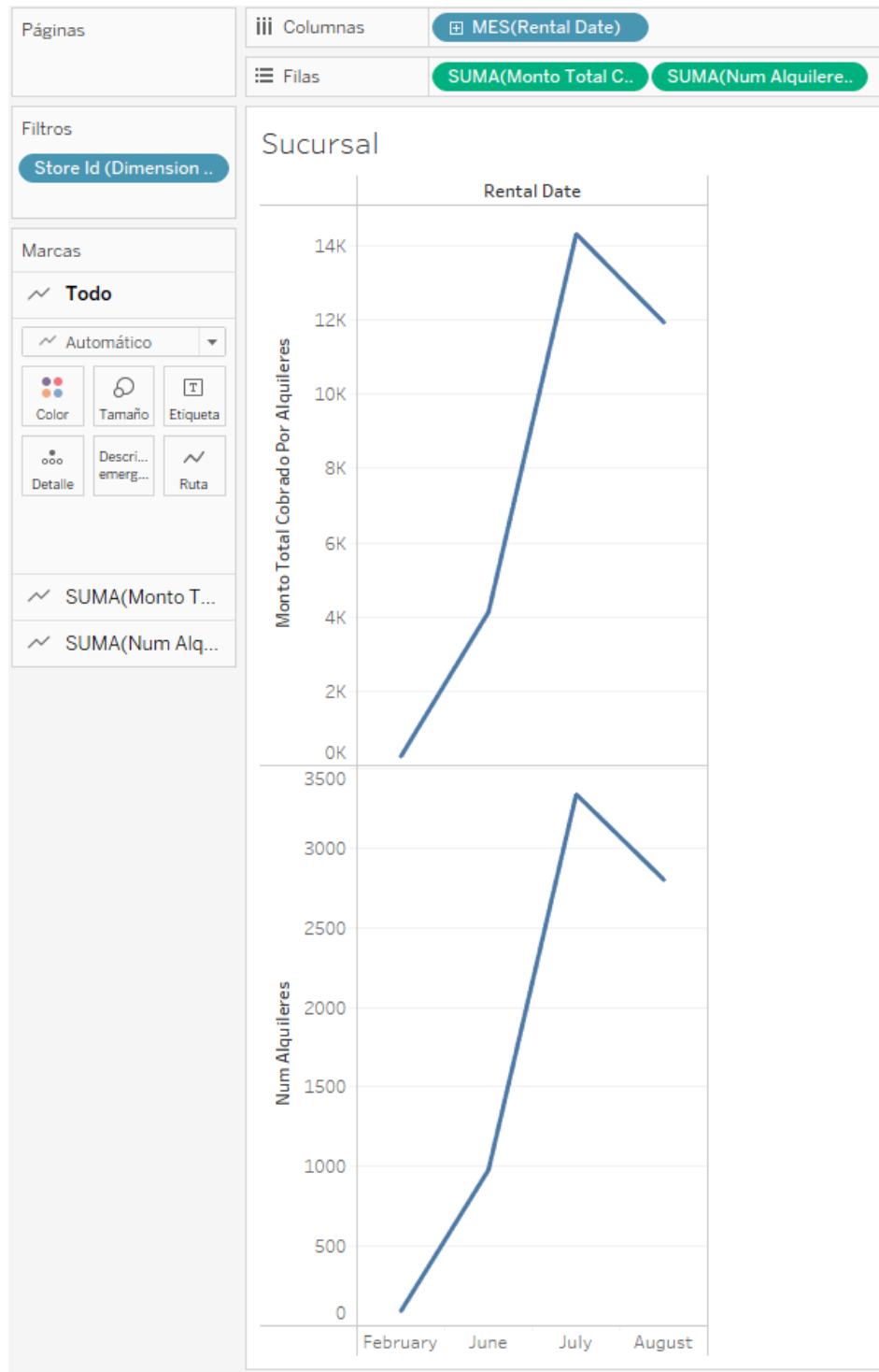
Tabla Hechos

Nombre del atributo	Definición	Valores de ejemplo
id_film	Identificador numérico que hace referencia a la llave primaria de la dimensión film.	1, 43
city_id	Identificador numérico que hace referencia a la llave primaria de la dimensión address.	21, 14
rental_id	Identificador numérico dentro de la dimensión rental.	46, 29
num_alquileres	El número total de alquileres realizados.	3001, 2131
monto_total_cobrado_por_alquileres	Los ingresos de todos los alquileres realizados.	14233, 23000

Visualización del modelo estrella

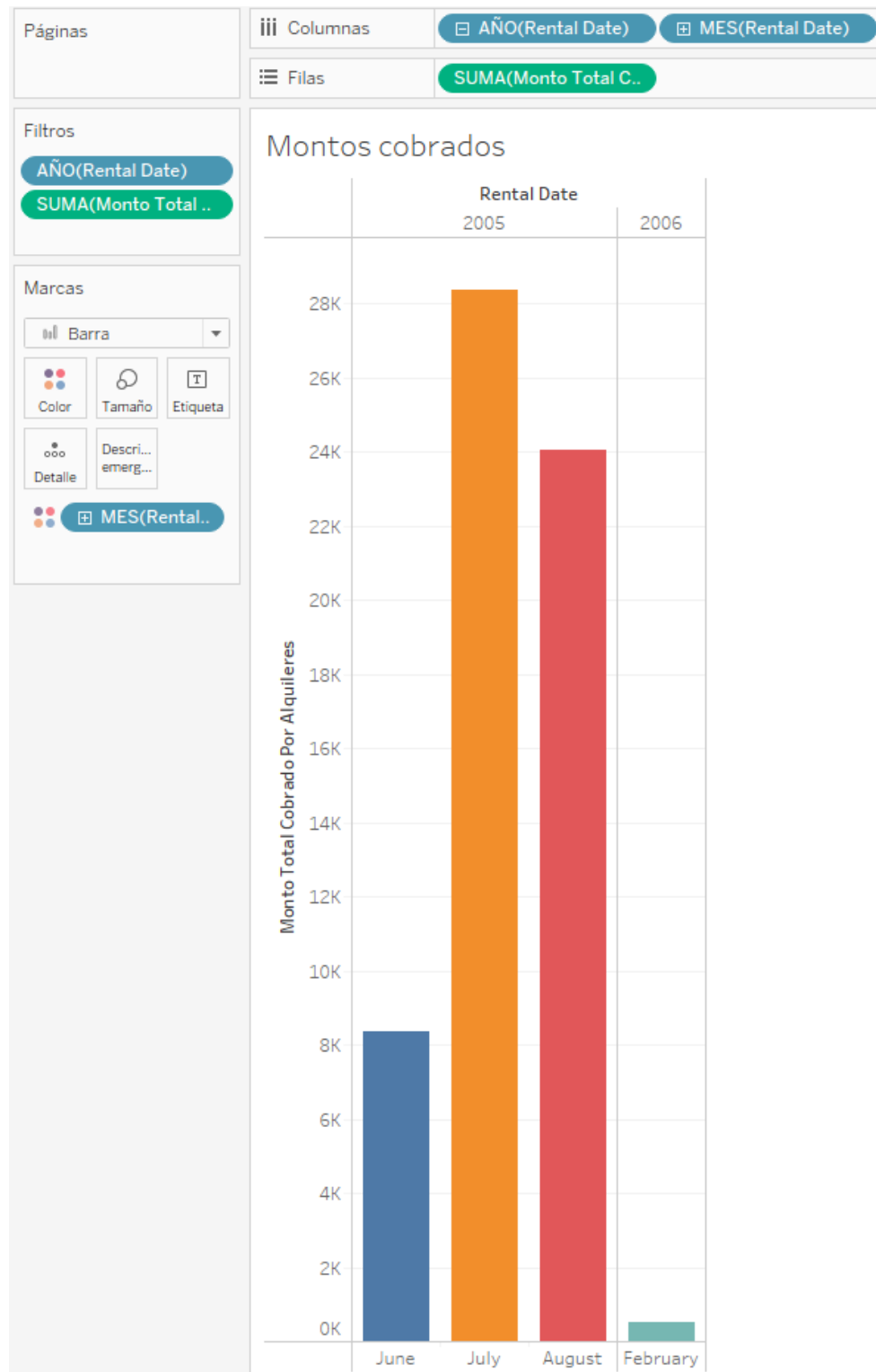
1. Para una sucursal (a seleccionar por el usuario), grafique el número de alquileres realizados y el monto cobrado por mes, sin importar el año.

Para su construcción se usó en las columnas la medida de mes (dimensión rental) para separarlo por meses del año. En las filas el monto total cobrado y el número de alquileres. En filtros el ID de la tienda para filtrar por tienda, como las tiendas no tienen nombre entonces el ID es lo único que puede diferenciarlas.



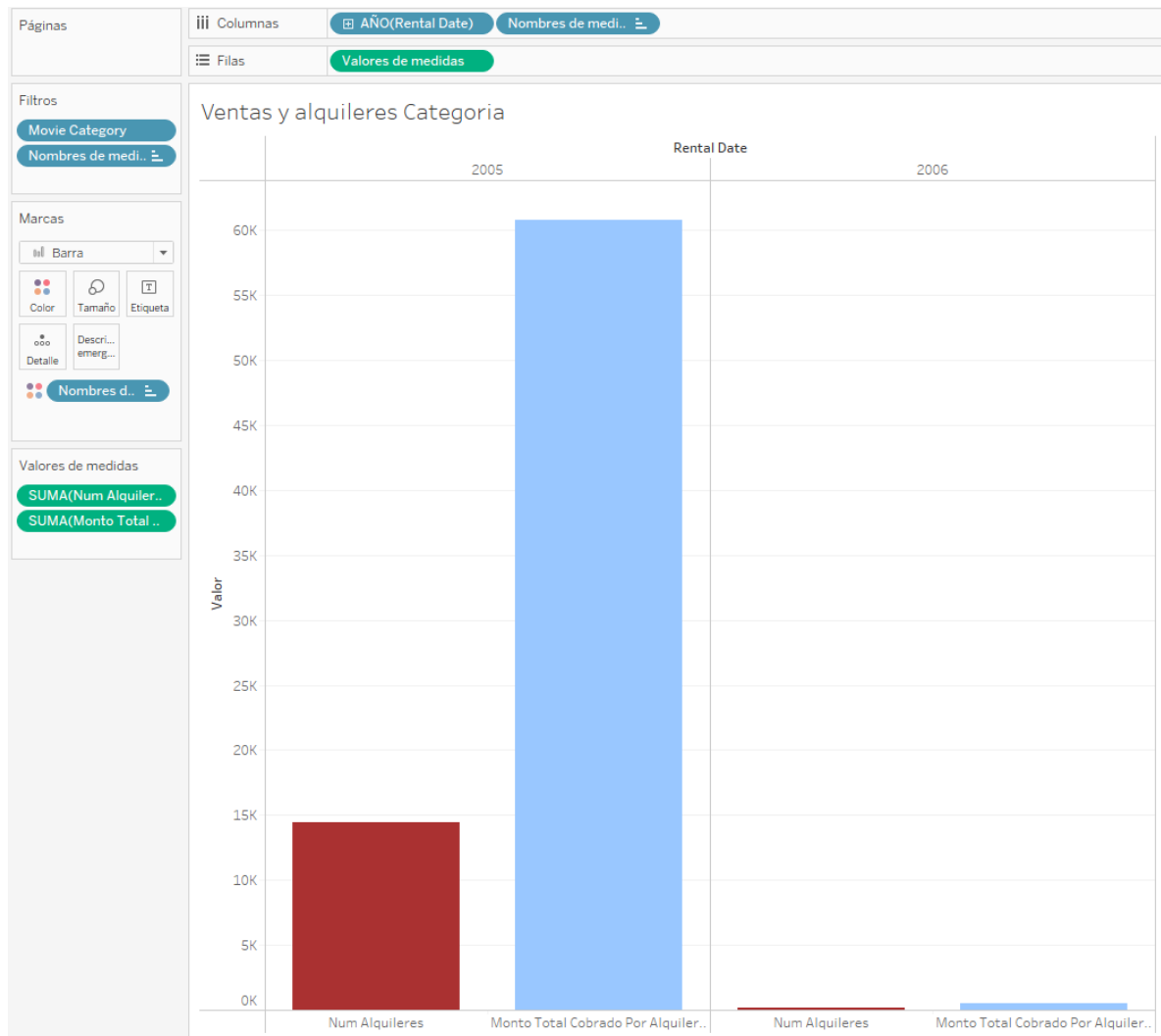
2. Graficar para un año (a seleccionar por el usuario) los montos cobrados por alquileres por mes.

Para su construcción se usó en las columnas la medida de año (dimensión rental) para mostrar el año y mes (dimensión rental) para separarlo por meses del año. En las filas el monto total cobrado. En filtros el año que se desea consultar y adicionalmente un filtro para no mostrar valores nulos.



3. Para una categoría de película (a seleccionar por el usuario), graficar el número de alquileres y el monto cobrado por año.

Para su construcción se usó en las columnas la medida de año (dimensión rental) para mostrar el año y nombres de medidas para que sea legible. En las filas se usó valores de medidas (representa el número de alquileres y el monto total cobrado). En filtros la categoría que se desea consultar.



4. Para los 10 actores con más alquileres, graficar los montos totales de alquileres por año (a seleccionar por el usuario). Incluya la opción de todos los años.

Para su construcción se usó en las columnas la medida del monto total cobrado por alquileres. En las filas el nombre y apellido del actor. En filtros el ID del actor (para mostrar los 10 actores con más alquileres) el año que se desea consultar.

Filtro [Id Actor] ×

General Comodín Condición Superior

☐ Ninguno

☒ Por campo:

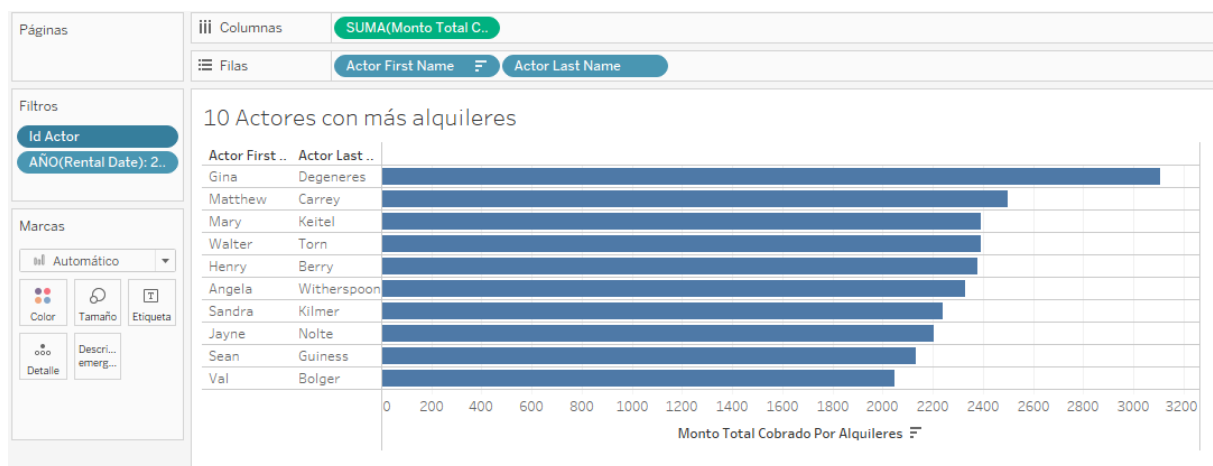
Superior 10 por

Num Alquileres Suma

☐ Por fórmula:

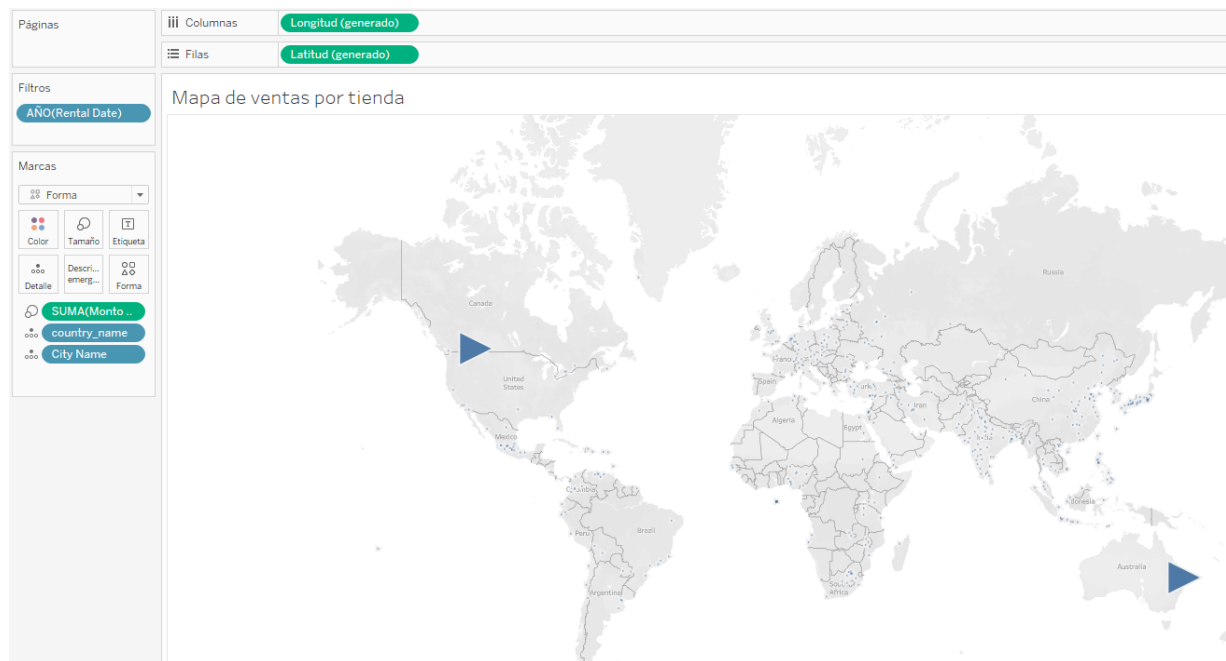
Superior 10 por

Restablecer Aceptar Cancelar Aplicar

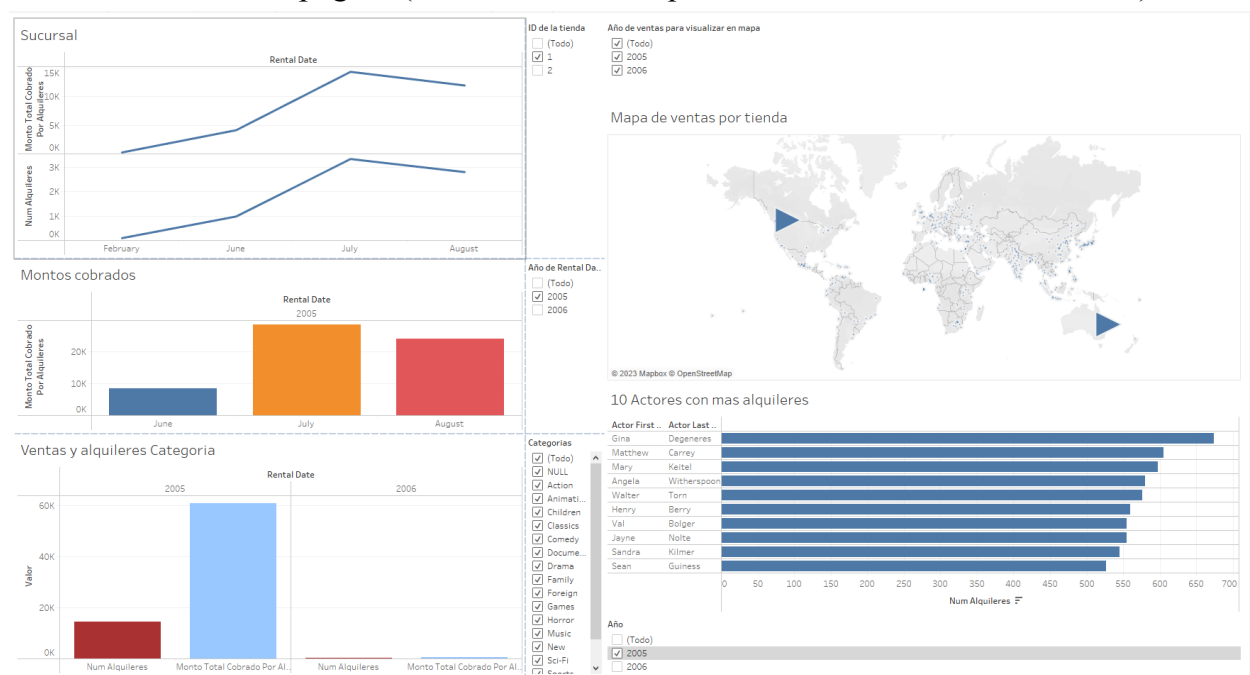


5. Despliegue un mapa de ciudades que presente por año el monto de alquiler total representado por el tamaño del punto sobre la ciudad. Incluya la opción de todos los años.

Para su construcción se usó en la columna la longitud. En la fila la latitud (generados en tableau). En filtros el año que se desea consultar. En cuanto a marcas se tiene el tamaño del punto (su tamaño está relacionado a la cantidad del monto total cobrado por alquiler en esa ciudad) y la jerarquía de la ubicación (país, ciudad) para ubicar los puntos en el mapa.



Para construir el dashboard solo hay que acomodar las páginas y añadir la opción de utilizar filtros a cada página (los filtros son los que anteriormente se establecieron).



Conclusión

Finalmente, el proyecto nos ayudó a comprender mejor el concepto de inteligencia de negocios y la utilidad de los diferentes objetivos establecidos en el proyecto.

Asimismo, restaurar una base de datos con un paquete de archivos fue una tarea que no se había realizado anteriormente, pero se comprendió que sirve como una alternativa para abordar diversos contextos. La ejecución del proceso de replicación también fue esencial para profundizar los conocimientos sobre el tema, al entender las diferentes formas de replicación y su importancia para conectar bases de datos según las necesidades específicas que se enfrenten.

Por otro lado, el análisis e implementación del modelo multidimensional fue un ejercicio que amplió nuestro entendimiento de la inteligencia de negocios. Destacó la importancia de dedicar suficiente tiempo para analizar cuidadosamente las dimensiones y medidas de interés para un proyecto. En última instancia, el uso del software Tableau demostró la relevancia de contar con una herramienta de visualización capaz de conectarse directamente a una base de datos. Esto puede ayudar a un usuario que quiera interpretar la información de manera más efectiva y comprender mejor los resultados obtenidos sin manipular la base de datos.

-- FUNCIONES Y PROCEDIMIENTOS - MODELOS TRANSACCIONAL, TABLAS OG --

```
/*-----  
* [Proceso]: insert_customer.  
* [Descripción]: inserta un nuevo cliente en la tabla "customer"  
* [Bloques relevantes]: tabla "customer", tabla "address", tabla "store".  
*  
*/
```

- * @param {INTEGER} p_store_id: (llave f6ranea) id de la tienda donde se registra el cliente.
- * @param {VARCHAR} p_first_name: nombre del cliente.
- * @param {VARCHAR} p_last_name: apellido del cliente.
- * @param {VARCHAR} p_email: correo electr3nico del cliente, acepta nulos.
- * @param {INTEGER} p_address_id: (llave f6ranea) id de la direcci3n del cliente.
- * @param {BOOLEAN} p_active: indica si un cliente est1 activo, acepta nulos.
- * @param {TIMESTAMP} p_create_date: fecha en la que se registra el cliente.
- * @param {INT} p_active_int: ??????????, acepta nulos.
- *
- * @returns {INTEGER} customer_id: NO RETORNA NING6N VALOR. Inserta una nuevo cliente en la tabla
- * | "customer", por lo que crea un nuevo id en esta.

-----*/

```
CREATE OR REPLACE PROCEDURE insert_customer(
    --Definici3n: Varibables del proceso.
    p_store_id INTEGER,
    p_first_name VARCHAR,
    p_last_name VARCHAR,
    p_email VARCHAR,
    p_address_id INTEGER,
    p_active BOOLEAN,
    p_create_date TIMESTAMP,
    p_active_int INT
)
AS $$
BEGIN
    -- Se insertan los valores ingresados en la table clientes.
    INSERT INTO customer (store_id, first_name, last_name, email, address_id,
activebool, create_date, active)
    VALUES (p_store_id, p_first_name, p_last_name, p_email, p_address_id, p_active,
p_create_date, p_active_int);
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

--Asignaci3n de creedenciales.

```
ALTER PROCEDURE insert_customer(p_store_id INTEGER, p_first_name
VARCHAR, p_last_name VARCHAR, p_email VARCHAR, p_address_id INTEGER,
p_active BOOLEAN, p_create_date TIMESTAMP, p_active_int INT)
OWNER TO video;
```



```
--Fin proceso: insert_customer.
--SELECT * FROM customer;
--SELECT * FROM address;
--CALL insert_customer(1, 'Julian', 'Navarro', "", 1, TRUE, '2023-10-10', 1)
```

```
/*-----
* [Proceso]: registrar_devolucion.
* [Descripción]: agrega un registro a la tabla "payment".
* | Calcula el monto a pagar de una devolución en base a la
* | cantidad que duro el alquiler.
* [Bloques relevantes]: tabla "payment".
*
* @param {SMALLINT} p_customer_id: id del cliente que hace la devolución.
* @param {SMALLINT} p_staff_id: id del empleado que registra la devolución.
* @param {INTEGER} p_rental_id: id del alquiler al que está asociado el alquiler.
* @param {TIMESTAMP WITHOUT TIME ZONE} p_payment_date: fecha en la
que se hace el pago.
*
* @var {NUMERIC} v_amount: variable definida en el procedure. Guarda el cálculo
del monto.
*
* @returns {INTEGER} payment_id: NO RETORNA NINGÚN VALOR. Inserta
una nuevo cliente en
* | la tabla "payment", por lo que crea un nuevo id en esta.
-----*/
CREATE OR REPLACE PROCEDURE register_devolution(
    p_customer_id SMALLINT,
    p_staff_id SMALLINT,
    p_rental_id INTEGER,
    p_payment_date TIMESTAMP WITHOUT TIME ZONE
)
AS $$
DECLARE
    v_amount numeric;
BEGIN
    -- Cálculo del monto
    SELECT EXTRACT(EPOCH FROM (p_payment_date - r.rental_date)) / 3600
/ 24 * f.rental_rate INTO v_amount
    FROM rental r
```

```

JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
WHERE r.rental_id = p_rental_id;

-- Excepción por si no se puede calcular el monto.
IF v_amount IS NULL THEN
RAISE EXCEPTION 'No se pudo calcular el monto a pagar';
END IF;

-- Inserta la devolución en la tabla payment
INSERT INTO payment (customer_id, staff_id, rental_id, amount, payment_date)
VALUES (p_customer_id, p_staff_id, p_rental_id, v_amount, p_payment_date);
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

--Asignación de credenciales.
ALTER PROCEDURE register_devolution(p_customer_id SMALLINT, p_staff_id
SMALLINT, p_rental_id INTEGER, p_payment_date TIMESTAMP WITHOUT
TIME ZONE
)
OWNER TO video;

--Fin proceso: registrar_devolucion
--call register_devolution(341::smallint, 2::smallint, 1778::integer, '2005-06-20
15:13:00'::timestamp);
--select * from payment where customer_id = 341 and staff_id = 2 and rental_id =
1778;

/*-----
* [Proceso]: registerRental.
* [Descripción]: ingresa un nuevo alquiler a la tabla de rentals.
* |
* [Bloques relevantes]: tabla "rental".
*
* @param {INTEGER} p_inventory_id: (llave foranea) id del inventario asociado al
alquiler.
* @param {SMALLINT} p_customer_id: (llave foranea) id del cliente que solicitó el
alquiler.

```

* @param {SMALLINT} p_staff_id: (llave foranea) id del empleado que autorizó el alquiler.

* @param {TIMESTAMP WITHOUT TIME ZONE} p_rental_date: fecha y hora en la que se registró el alquiler.

* @param {TIMESTAMP WITHOUT TIME ZONE} p_return_date: fecha y hora en la que se hace la devolución del alquiler.

*

* @returns {INTEGER} rental_id: NO RETORNA NINGÚN VALOR. Inserta un nuevo alquiler en la tabla

* | "rental", por lo que crea un nuevo id en esta.

-----*/

CREATE OR REPLACE PROCEDURE registerRental (

--Definición: Varibables del proceso.

p_rental_date TIMESTAMP WITHOUT TIME ZONE,

p_inventory_id INTEGER,

p_customer_id SMALLINT,

p_return_date TIMESTAMP WITHOUT TIME ZONE,

p_staff_id SMALLINT

)

AS \$\$

BEGIN

-- Se hace el insert de los valores ingresados a la tabla de rentals.

INSERT INTO rental (rental_id, rental_date, inventory_id, customer_id,
return_date, staff_id)

VALUES(nextval('rental_rental_id_seq'), p_rental_date, p_inventory_id,
p_customer_id, p_return_date, p_staff_id);

COMMIT;

END;

\$\$ LANGUAGE plpgsql;

--Asignación de credenciales.

ALTER PROCEDURE registerRental(p_rental_date TIMESTAMP WITHOUT
TIME ZONE, p_inventory_id INTEGER, p_customer_id SMALLINT, p_return_date
TIMESTAMP WITHOUT TIME ZONE, p_staff_id SMALLINT)

OWNER TO video;

--Fin proceso: registerRental.

--CALL registerRental ('2010-01-01 15:30:30'::timestamp, 1525::integer,
408::smallint, '2010-01-06 15:00:00'::timestamp, 2::smallint);

--SELECT * FROM rental where rental_date = '2010-01-01 15:30:30';

-- CREACIÓN DE ROLES Y USUARIOS - ASIGNACIÓN DE PERMISOS Y REVOKES --

/*Crea un rol temporal para asignar ownership
* en caso de que se necesite dropear un role.
*/

CREATE ROLE temp_elim;

/*Se trasfiere el ownership de EMP a temp_elim
* Se eliminan los objetos de EMP y después el
* rol en sí
*/

REASSIGN OWNED BY EMP TO temp_elim;
DROP OWNED BY EMP;
DROP ROLE EMP;

/*Se trasfiere el ownership de empleado1
* a temp_elim. Se eliminan los objetos de
* empleado1 y después el rol en sí.
*/

REASSIGN OWNED BY empleado1 TO temp_elim;
DROP OWNED BY empleado1;
DROP ROLE empleado1;

/*Se trasfiere el ownership de ADMIN a
* temp_elim. Se eliminan los objetos de
* ADMIN y después el rol en sí.
*/

REASSIGN OWNED BY ADMIN TO temp_elim;
DROP OWNED BY ADMIN;
DROP ROLE ADMIN;

/*Se trasfiere el ownership de administrador1
* a temp_elim. Se eliminan los objetos de
* administrador1 y después el rol en sí.
*/

REASSIGN OWNED BY administrador1 TO temp_elim;
DROP OWNED BY administrador1;
DROP ROLE administrador1;

```
/*Se trasfiere el ownership de video a
* temp_elim. Se eliminan los objetos de
* video y después el rol en sí.
*/
```

```
REASSIGN OWNED BY video TO temp_elim;
DROP OWNED BY video;
DROP ROLE video;
```

```
/*Se trasfiere el ownership de temp_elim
* a postgres. Se eliminan los objetos de
* temp_elim y después el rol en sí.
*/
```

```
REASSIGN OWNED BY temp_elim TO postgres;
DROP OWNED BY temp_elim;
DROP ROLE temp_elim;
```

```
/*Linea para ver los roles de la BD*/
-- SELECT rolname FROM pg_roles;
```

```
/*Lineas para probar funcionalidad de los roles y users*/
--select * from actor;
--select * from get_film('Airport Pollock');
--call insert_customer(1, 'Juliadfgfaan', 'Navaasdgrrosa', 'nasgs@gmail.com', 1,
TRUE, '2023-10-10', 1);
```

```
-----
-----
```

```
/*Creal el role EMP*/
create role EMP;
```

```
/*Asigna los permisos para el rol EMP sobre la función get_fim*/
grant execute on function get_film(p_titulo text) to EMP;
```

```
/*Asigna los permisos para el rol EMP sobre la procedure registerRental*/
grant execute on procedure registerRental (
    p_rental_date TIMESTAMP WITHOUT TIME ZONE, p_inventory_id
INTEGER,
```

```
        p_customer_id SMALLINT, p_return_date TIMESTAMP WITHOUT TIME
ZONE,
        p_staff_id SMALLINT
    ) to EMP;
```

```
/*Asigna los permisos para el rol EMP sobre la procedure registrar_devolucion*/
grant execute on procedure registrar_devolucion(p_rental_id INT) to EMP;
```

```
/*Le quita los permisos de ejecutar el procedure insert_customer a EMP*/
```

```
REVOKE EXECUTE ON PROCEDURE insert_customer(
    p_store_id INTEGER,
    p_first_name VARCHAR,
    p_last_name VARCHAR,
    p_email VARCHAR,
    p_address_id INTEGER,
    p_active BOOLEAN,
    p_create_date TIMESTAMP,
    p_active_int INT)
FROM EMP;
```

```
/*Remueve la posibilidad ejecutar el procedure insert_customer desde el esquema
public*/
```

```
REVOKE EXECUTE ON PROCEDURE insert_customer(
    p_store_id INTEGER,
    p_first_name VARCHAR,
    p_last_name VARCHAR,
    p_email VARCHAR,
    p_address_id INTEGER,
    p_active BOOLEAN,
    p_create_date TIMESTAMP,
    p_active_int INT)
FROM public;
```

```
-----

/*Crea el rol ADMIN*/
```

```
create role ADMIN;
```

```
/*Asigna todos los permisos de EMP a ADMIN*/
```

```
grant EMP to ADMIN;
```

```
/*Permite ejecutar el procedure insert_customer desde ADMIN*/  
grant execute on procedure insert_customer(  
    p_store_id INTEGER,  
    p_first_name VARCHAR,  
    p_last_name VARCHAR,  
    p_email VARCHAR,  
    p_address_id INTEGER,  
    p_active BOOLEAN,  
    p_create_date TIMESTAMP,  
    p_active_int INT  
) to ADMIN;
```

```
/*Crea el user video, sin capacidad de logearse en la DB*/  
create user video nologin;
```

```
/*Le asigna el status de SUPERUSER a video*/  
alter user video with superuser;
```

```
/*Se le conceden todos los permisos sobre la base de datos a video*/  
grant all privileges on all tables in schema public to video;  
grant all privileges on database dvdrental to video;
```

```
/*Se crea un usuario EMP con contraseña*/  
create user empleado1 with password '123';  
grant EMP to empleado1;
```

```
/*Se crea un usuario ADMIN con contraseña*/  
create user administrador1 with password '123';  
grant ADMIN to administrador1;
```

-- MODELO MULTIDIMENSIONAL - DIMENSIONES Y TABLA DE HECHOS --

-- se eliminan las tablas de las dimensiones y la tabla de hechos

```
drop table dimension_film cascade;
drop table dimension_address cascade;
drop table dimension_rental cascade;
drop table dimension_store cascade;
drop table dimension_actor cascade;
drop table dimension_film_actor cascade;
drop table tabla_hechos cascade;
```

-- tabla de dimension film

```
create table dimension_film(
  id_film integer primary key,
  title character varying,
  movie_category character varying
);
```

-- tabla de dimension address (ciudad)

```
create table dimension_address(
  city_id integer primary key,
  country_name character varying,
  city_name character varying
);
```

-- tabla de dimension actor

```
create table dimension_actor(
  id_actor integer primary key,
  actor_first_name character varying,
  actor_last_name character varying
);
```

-- tabla de dimension film en relacion con actor (para poder hacer la relacion de muchos a muchos)

```
CREATE TABLE dimension_film_actor (
  id_film_actor serial PRIMARY KEY,
  id_film integer,
  id_actor integer,
```



```
foreign key (id_film) references dimension_film(id_film),
foreign key (id_actor) references dimension_actor(id_actor)
);
```

```
-- table de dimension rental
create table dimension_rental(
rental_id integer primary key,
rental_date timestamp without time zone
);
```

```
-- tabla de dimension store
create table dimension_store(
store_id integer primary key
);
```

```
-- tabla de hechos
create table tabla_hechos(
-- llaves foraneas
id_film integer,
city_id integer,
rental_id integer,
store_id integer,
-- medidas de interes
num_alquileres integer,
monto_total_cobrado_por_alquileres numeric,
-- referencias a sus respectivas dimensiones
foreign key (id_film) references dimension_film(id_film),
foreign key (city_id) references dimension_address(city_id),
foreign key (rental_id) references dimension_rental(rental_id),
foreign key (store_id) references dimension_store(store_id)
);
```

```
-- Procedimientos almacenados para alimentar las tablas de dimension y la tabla de hechos
```

```
/* Procedimiento para alimentar la tabla de dimension film
```

```
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension film
```

```
* Parametros: No recibe parametros
```

```
* Salida: Actualiza la tabla de dimension film con los datos de la tabla film de interes para el modelo estrella */
```

```

CREATE OR REPLACE PROCEDURE alimentar_dimension_film()
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO dimension_film (id_film, title, movie_category)
    -- id de la pelicula, titulo de la pelicula, categoria de la pelicula
    SELECT film.film_id, film.title, category.name
    FROM category
    -- se hace un join con la tabla film_category para obtener el id de la categoria de la
    pelicula
    INNER JOIN film_category ON category.category_id = film_category.category_id
    -- se hace un join con la tabla film para obtener el id de la pelicula y el titulo de la
    pelicula
    INNER JOIN film ON film_category.film_id = film.film_id;
END;
$$;

```

```

/* Procedimiento para alimentar la tabla de dimension actor
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension
* actor
* Parametros: No recibe parametros
* Salida: Actualiza la tabla de dimension actor con los datos de la tabla actor de
* interes para el modelo estrella
*/

```

```

CREATE OR REPLACE PROCEDURE alimentar_dimension_actor()
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO dimension_actor (id_actor, actor_first_name, actor_last_name)
    -- id del actor, nombre del actor, apellido del actor
    SELECT actor_id, first_name, last_name
    FROM actor;
END;
$$;

```

```

/* Procedimiento para alimentar la tabla de dimension film_actor
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension
* film_actor (relacion entre film y actor)
* El objetivo de esta tabla es poder hacer la relacion de muchos a muchos entre las
* tablas de dimension film y actor.
* Parametros: No recibe parametros

```

* Salida: Actualiza la tabla de dimension film_actor con los datos de la tabla
* film_actor de interes para el modelo estrella
*/

```
CREATE OR REPLACE PROCEDURE alimentar_dimension_film_actor()  
LANGUAGE plpgsql AS $$  
BEGIN  
    -- Insertar datos en dimension_film_actor desde la tabla film_actor  
    INSERT INTO dimension_film_actor (id_film, id_actor)  
    SELECT film_id, actor_id  
    FROM film_actor;  
END;  
$$;
```

/* Procedimiento para alimentar la tabla de dimension address (direcccion)
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension
* address
* Parametros: No recibe parametros
* Salida: Actualiza la tabla de dimension address con los datos de la tabla address de
* interes para el modelo estrella
*/

```
CREATE OR REPLACE PROCEDURE alimentar_dimension_address()  
LANGUAGE plpgsql AS $$  
BEGIN  
    INSERT INTO dimension_address (city_id, country_name, city_name)  
    SELECT DISTINCT ON (city.city_id)  
        city.city_id, country.country, city.city  
    FROM address  
    INNER JOIN city ON address.city_id = city.city_id  
    INNER JOIN country ON city.country_id = country.country_id;  
END;  
$$;
```

/* Procedimiento para alimentar la tabla de dimension rental
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension
* rental
* Parametros: No recibe parametros
* Salida: Actualiza la tabla de dimension rental con los datos de la tabla rental de
* interes para el modelo estrella
*/

```
CREATE OR REPLACE PROCEDURE alimentar_dimension_rental()
```

```
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO dimension_rental (rental_id, rental_date)
    SELECT rental.rental_id, rental.rental_date
    FROM rental;
END;
$$;
```

```
/* Procedimiento para alimentar la tabla de dimension store
* Descripcion: Este procedimiento se encarga de alimentar la tabla de dimension store
* Parametros: No recibe parametros
* Salida: Actualiza la tabla de dimension store con los datos de la tabla store de
* interes para el modelo estrella
*/
```

```
CREATE OR REPLACE PROCEDURE alimentar_dimension_store ()
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO dimension_store (store_id)
    SELECT store_id
    FROM store;
END;
$$;
```

```
/* Procedimiento para alimentar la tabla de hechos
* Descripcion Inserta los datos en la tabla de hechos con los datos de las tablas de
* dimension y la tablas de dvd rental
* Parametros: No recibe parametros
* Salida: Actualiza la tabla de hechos
*/
```

```
CREATE OR REPLACE PROCEDURE alimentar_tabla_hechos()
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO tabla_hechos (id_film, city_id, rental_id, store_id, num_alquileres,
    monto_total_cobrado_por_alquileres)
    SELECT dimension_film.id_film, city.city_id, dimension_rental.rental_id,
    store.store_id, COUNT(rental.rental_id), SUM(payment.amount)
    FROM dimension_film
    INNER JOIN film ON dimension_film.id_film = film.film_id
    INNER JOIN inventory ON inventory.film_id = film.film_id
    INNER JOIN store ON store.store_id = inventory.store_id
```

```
INNER JOIN rental ON rental.inventory_id = inventory.inventory_id
INNER JOIN dimension_rental ON dimension_rental.rental_id = rental.rental_id
INNER JOIN address ON store.address_id = address.address_id
INNER JOIN city ON city.city_id = address.city_id
INNER JOIN payment ON payment.rental_id = dimension_rental.rental_id
GROUP BY dimension_film.id_film, city.city_id, dimension_rental.rental_id,
store.store_id;
END;
$$;
```

```
CALL alimentar_dimension_film();
CALL alimentar_dimension_actor();
CALL alimentar_dimension_film_actor();
CALL alimentar_dimension_address();
CALL alimentar_dimension_rental();
CALL alimentar_dimension_store();
CALL alimentar_tabla_hechos();
```

```
-- Tabla de dimensiones dimension_film
SELECT * FROM dimension_film;
-- Tabla de dimensiones dimension_address
SELECT * FROM dimension_address;
-- Tabla de dimensiones dimension_actor
SELECT * FROM dimension_actor;
-- Tabla de relación dimension_film_actor
SELECT * FROM dimension_film_actor;
-- Tabla de dimensiones dimension_rental
SELECT * FROM dimension_rental;
-- Tabla de dimensiones dimension_store
SELECT * FROM dimension_store;
-- Tabla de hechos
SELECT * FROM tabla_hechos;
```