



	Rubrica	G11
Proyecto II	insertar cliente	3
	registrar alquiler	3
	registrar devolución	1
Bases de datos II	buscar una película	1
	emp	1
	adm	1
	video	0.5
Grupo 20	empleado1	1
	administrador1	1
	replica	30
	modelo multidimensional	20
Profesor: Alberto Shum Chan	sucursal	5
	año	5
	categoría	4
	actores	5
	mapa	7
Estudiantes: Alberto Zumbado Abarca Jonathan Quesada Salas	documentación	5
		93.5

Introducción:

El proyecto programado consistirá en instanciar una base de datos esclava la cual va a tener la replicación de la base de datos dvdrental, dicha replicación puede realizarse de la forma que se quiera, ya sea de manera sincrónica, o bien asincrónica. Cabe resaltar que por motivos del proyecto, la opción de hacer la replicación de una manera asincrónica resultó más factible de usar, adyacente a esto se debe de establecer distintos tipos de procedimientos, usuarios y roles, estos dichos procedimientos pueden cumplir funciones generales como puede ser insertar un nuevo cliente, registrar, registrar un alquiler, registrar una devolución y buscar una película, adicionalmente a esto, se debe de realizar un modelo estrella donde se deban crearse las dimensiones de interés y alimentar las mismas por medio de procedimientos almacenados. Para luego teniendo el modelo multidimensional se pueda desarrollar la conexión del servidor, la base de datos al software Tableau, siendo que esta nos va a servir para poder expresar, graficar de una mejor manera las medidas de interés con respecto a condiciones dadas en la especificación del proyecto, para que de esta manera se pueda desarrollar de una mejor manera nuestra inteligencia de negocios.

Descripción del proyecto:

En el dicho proyecto consistirá de 4 puntos principales:

1. Funciones o procedimientos almacenados que afecten el sistema transaccional y Seguridad
2. Replicación
3. Modelo multidimensional
4. Visualización y acceso por medio de interfaz gráfica al modelo estrella

PARTE I:

Se deben de crear los siguientes procedimientos para desarrollar correctamente el sistema transaccional:

- Insertar un nuevo cliente
- Registrar un alquiler
- Registrar una devolución
- Buscar una película

Cabe recalcar que dichos procedimientos deben de estar debidamente documentadas que incluya una descripción, descripción de parámetros, descripción de salida y descripción de bloques relevantes.

En la parte de Seguridad se deben de 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.

Dicho rol debe de tener acceso a los siguientes procedimientos:

- 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.

Dicho rol debe de tener acceso a los siguientes procedimientos:

- Insertar un nuevo cliente

En la parte de Seguridad de tener acceso a 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

PARTE II:

Se debe de establecer una instancia esclava de la base de datos, la réplica puede estar en la misma máquina.

La replicación puede usarse cualquier tipo de método de replicación en este caso este grupo uso Slony, se establece como una replicación asincrónica.

Algo más a tener en cuenta es que las tablas y el modelo estrella requerido para el proyecto debe de establecerse en la instancia de la réplica.

PARTE III:

Se debe de desarrollar un modelo multidimensional que va a tener 2 medidas de interés, las cuales son:

- Número de alquileres
- Monto total cobrado por alquileres

Algo que tener en cuenta que en el modelo multidimensional se ocupan dimensiones en específico para poder desarrollar el modelo estrella, dichas dimensiones son las siguientes:

- Película (Film): Los parámetros de dicha dimensión serían:
 - Categoría (categoría_nombre)
 - Filme (Title)
 - Actores (Actor_nombre)
- Lugar (Address): Los parámetros de dicha dimensión serían:
 - País (country_name)
 - Ciudad (city_name)
- Fecha (Rental): Su jerarquía sería de año, mes y día.
 - Rental_date
- Sucursal (Store): No contiene jerarquías

Adicionalmente a la creación de dichas dimensiones, se requiere crear los procedimientos requeridos para poder alimentar las dimensiones y la tabla de hechos en cuestión con todos sus respectivos datos que existen en un principio en la base de datos dvdrental.

- pr_insertar_dim_film
- pr_insertar_dim_address
- pr_ingresar_dim_rental
- pr_ingresar_dim_store
- pr_ingresar_hechos

PARTE IV:

Se debe diseñar e implementar un dashboard que resuma información importante sobre los datos utilizando el software **Tableau**, dicha información debe de ser la siguiente:

- 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.

-- 1.1 Funciones o procedimientos almacenados que afecten el sistema transaccional

/**

* 1. Insertar un nuevo cliente.

* Inserta en la tabla de customer un nuevo cliente

* Entradas: ID de la sucursal, Nombre del cliente, Apellido del cliente,
Email, ID de la direccion, ID del activo.

* Salidas: No tiene, pero se actualiza la tabla customer

*/

create or replace procedure propr_nuevo_cliente(

nc_store_id smallint,

nc_first_name character varying,

nc_last_name character varying,

nc_email character varying,

nc_address_id smallint,

nc_active integer

```

)
as $$
begin
    -- Se insertan los valores en la tabla
    insert into customer(store_id, first_name, last_name, email, address_id, active)
    values (nc_store_id, nc_first_name, nc_last_name, nc_email, nc_address_id,
nc_active);

    -- Se confirman los cambios
    commit;
end;
$$ language plpgsql;

```

```

/**
 * 2. Registrar un alquiler.
 * Inserta en la tabla de rental un nuevo alquiler
 * Entradas: Fecha del alquiler, ID del inventario, ID del cliente, Fecha de retorno, ID
del personal
 * Salidas: No tiene, pero se actualiza la tabla rental
 */

```

```

create or replace procedure registrar_alquiler(
    ra_rental_date timestamp without time zone,
    ra_inventory_id integer,
    ra_customer_id smallint,
    ra_return_date timestamp without time zone,
    ra_staff_id smallint
)
as $$
begin
    -- Se insertan los valores en la tabla
    insert into rental (rental_date, inventory_id, customer_id, return_date, staff_id)
    values (ra_rental_date, ra_inventory_id, ra_customer_id, ra_return_date,

```

```

        ra_staff_id);

        -- Se confirman los cambios
    commit;
end;
$$ language plpgsql;

/**
 * 3. Registrar una devolución.
 * Actualiza el valor del return_date cierto alquiler en especifico
 * Entradas: ID del alquiler
 * Salidas: No tiene, pero se actualiza las fechas de retorno de la tabla rental
 */
create or replace procedure registrar_devolucion(rd_rental_id integer)
as $$
begin
    -- Se actualiza las columnas de la fila especifica
    update film set film.return_date = now()::timestamp without time zone,
    film.last_update = now()::timestamp without time zone
    where film.rental_id = rd_rental_id;

    -- Se confirman los cambios
    commit;
end;
$$ language plpgsql;

/**
 * 4. Buscar una película.
 * Selecciona los valores de una pelicula
 * Entradas: ID de la pelicula

```

* Salidas: Cursor que contiene los datos de la pelicula en especifica

*/

create or replace function buscar_pelicula(p_film_id integer) returns refcursor as \$\$

declare

-- Definicion del cursor de salida

ref_cursor refcursor = 'pelicula_cursor';

begin

-- Se la fila de la pelicula en el cursor

open ref_cursor for select * from film where film.film_id = p_film_id;

return ref_cursor;

end;

\$\$ language plpgsql;

begin;

select buscar_pelicula(2::integer);

fetch all in pelicula_cursor;

commit;

-- 1.2 Seguridad

/**

* 1. Creacion rol EMP

*/

-- Creacion del rol

create role EMP;

```
-- Acceso a las funciones y procedimientos almacenados especificos
grant execute on function buscar_pelicula(p_film_id integer) to EMP;
grant execute on procedure registrar_alquiler(
    ra_rental_date timestamp without time zone,
    ra_inventory_id integer,
    ra_customer_id smallint,
    ra_return_date timestamp without time zone,
    ra_staff_id smallint
), registrar_devolucion(rd_rental_id integer) to EMP;
```

```
/**
```

```
 * 2. Creacion rol ADMIN
```

```
 */
```

```
-- Creacion del rol
```

```
create role admin in role EMP; -- Extiende las funcionalidades de EMP
```

```
-- Acceso al procedimiento de ingresar un nuevo cliente
```

```
grant execute on procedure propr_nuevo_cliente(
    nc_store_id smallint,
    nc_first_name character varying,
    nc_last_name character varying,
    nc_email character varying,
    nc_address_id smallint,
    nc_active integer
) to admin;
```

```
/**
```

```
 * 3. Creacion de los usuarios
```

```
 */
```


-- Creacion del usuario video

```
create role video1 nologin;
```

```
alter user video1 with superuser;
```

```
grant all privileges on all tables in schema public to video1;
```

```
grant all privileges on database dvdrental to video1;
```

-- Creacion del usuario empleado1

```
create role empleado1;
```

```
grant EMP to empleado1; --Se asigan los privilegios de emp
```

-- Creacion del usuario administrador1

```
create role administrador1;
```

```
grant admin to administrador1; --Se asigan los privilegios de admin
```

-- Scripts extra para eliminacion de ciertos roles

```
revoke execute on procedure registrar_alquiler(
```

```
    ra_rental_date timestamp without time zone,
```

```
    ra_inventory_id integer,
```

```
    ra_customer_id smallint,
```

```
    ra_return_date timestamp without time zone,
```

```
    ra_staff_id smallint
```

```
), registrar_devolucion(rd_rental_id integer) from EMP;
```

```
revoke execute on function buscar_pelicula(p_film_id integer) from EMP;
```

```
revoke execute on procedure propr_nuevo_cliente(
```

```
    nc_store_id smallint,
```

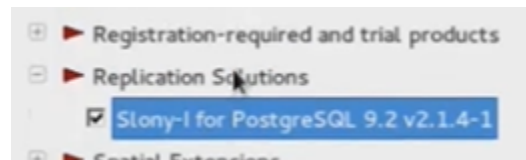
```
    nc_first_name character varying,
```

```
    nc_last_name character varying,
```

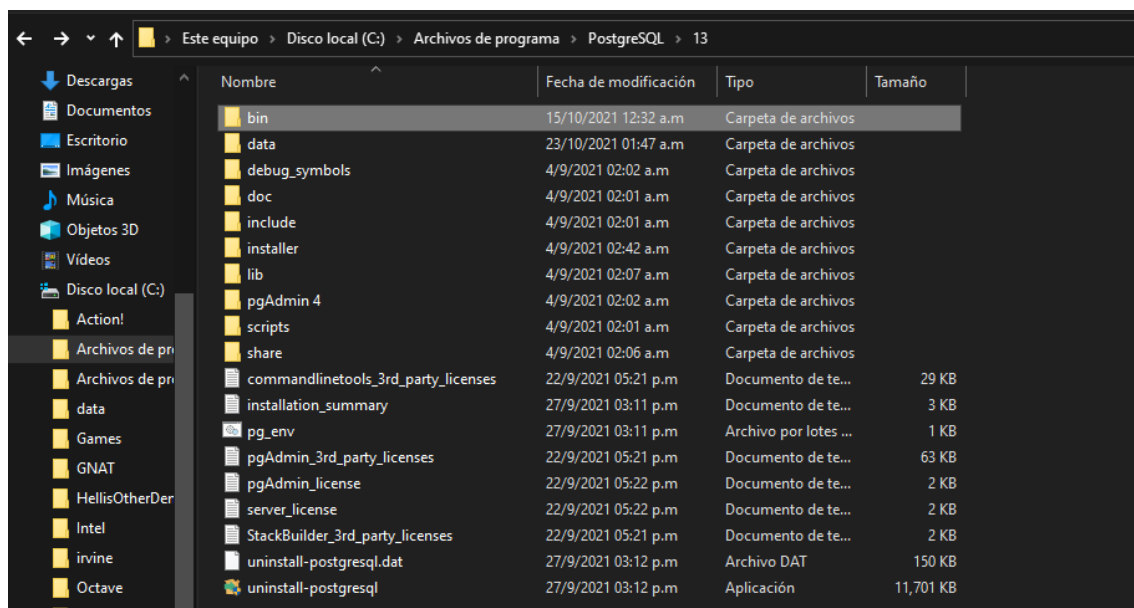
nc_email character varying,
nc_address_id smallint,
nc_active integer
) from admin;

-- 2. Replicación (Con Slony-1)

El primer paso a seguir para poder empezar realizar la replicación con Slony descargar PostgreSQL con la extensión de Slony.




El siguiente paso a seguir dirigirse a la carpeta “bin” de PostgreSQL:



Al entrar a la carpeta “bin” se procede a crear 2 archivos de extensión .txt, los siguientes serán “replicacion.txt” que va a ser nuestro cluster maestro y “slave.txt” será el archivo donde se va a instanciar la replicación.

El archivo “replicacion.txt” debe de contener lo siguiente:

 replicacion: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
|cluster name = slony_rp;
```

```
node 1 admin conninfo = 'dbname=dvdrental host=localhost user=postgres password=root';
node 2 admin conninfo = 'dbname=slave host=localhost user=postgres password=root';
```

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


```
create set (id=1, origin=1, comment='All pgbench tables');
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 = 'Slave node', event node=1);
store path (server = 1, client = 2, conninfo='dbname=dvdrental host=localhost user=postgres password=root');
store path (server = 2, client = 1, conninfo='dbname=slave host=localhost user=postgres password=root');
```

```
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);
```

Y el archivo “slave.txt” debe de contener lo siguiente:

 slave: Bloc de notas

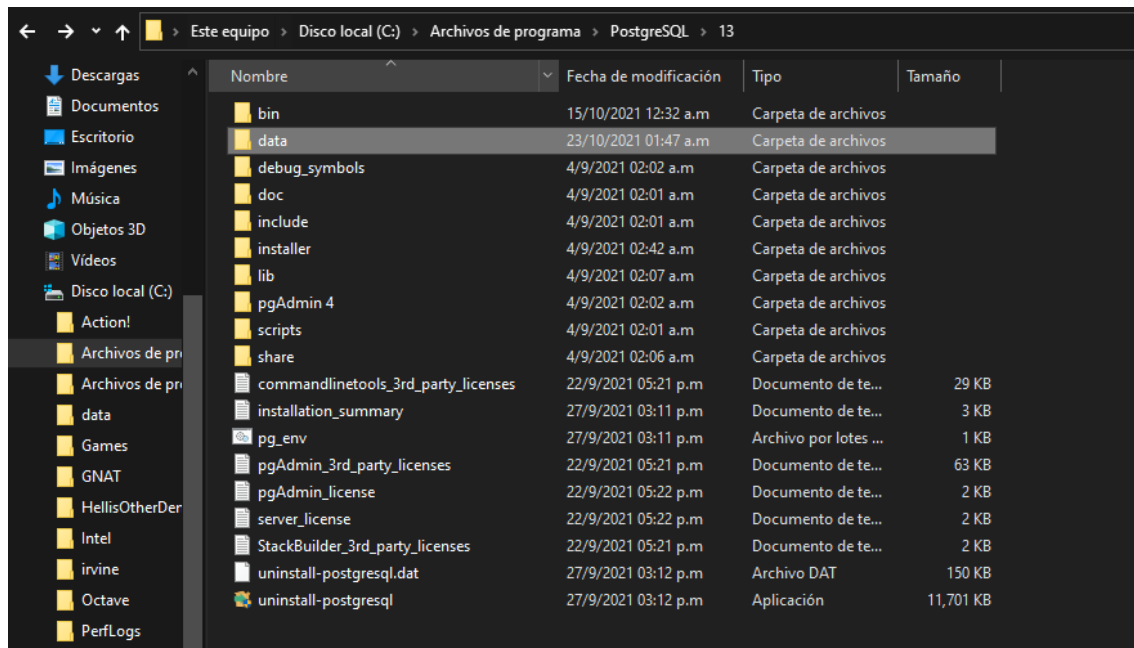
Archivo Edición Formato Ver Ayuda

```
|cluster name = slony_rp;
```

```
node 1 admin conninfo = 'dbname=dvdrental host=localhost user=postgres password=root';
node 2 admin conninfo = 'dbname=slave host=localhost user=postgres password=root';
```

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

Ya habiendo creados los anteriores archivos de extensión .txt en la carpeta “bin”, se deberá dirigir a la carpeta “data” de PostgreSQL:



Ya estando en dicha carpeta se debe ingresar en el archivo “pg_hba” y hacer scoll hasta abajo del archivo y hacer pequeñas modificaciones para establecer la IP de la replicación esclava y el maestro, dichas modificaciones son las siguientes:

```
# TYPE DATABASE USER ADDRESS METHOD
# IPv4 local connections:
host all all 127.0.0.1/32 md5
#Esclavo
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
```

Ya teniendo modificados los archivos respectivos, se procede a abrir la terminal y poner el siguiente comando para inicializar el archivo maestro que establecimos anteriormente, en la carpeta bin de PostgreSQL:

slonik replicacion.txt

```
Símbolo del sistema - slonik.exe replicacion.txt
Microsoft Windows [Versión 10.0.19042.1200]
(c) Microsoft Corporation. Todos los derechos reservados.

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

C:\Program Files\PostgreSQL\11\bin>slonik.exe replicacion.txt
replicacion.txt:6: Possible unsupported PostgreSQL version (111300) 11.13, defaulting to 8.4 support
replicacion.txt:25: Possible unsupported PostgreSQL version (111300) 11.13, defaulting to 8.4 support
replicacion.txt:29: waiting for event (1,50000000035) to be confirmed on node 2
replicacion.txt:29: waiting for event (1,50000000035) to be confirmed on node 2
```

Ya habiendo ejecutado el comando anterior se procede a abrir otra terminal, para inicializar al esclavo con el siguiente comando, en la carpeta bin de PostgreSQL:

slonik slave.txt

```
Símbolo del sistema - slonik.exe slave.txt
Microsoft Windows [Versión 10.0.19042.1200]
(c) Microsoft Corporation. Todos los derechos reservados.

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

C:\Program Files\PostgreSQL\11\bin>slonik.exe slave.txt
waiting for events (2,50000000001) only at (2,0) to be confirmed on node 1
waiting for events (2,50000000001) only at (2,0) to be confirmed on node 1
```

Ya teniendo corriendo los dos anteriores comandos se procede a inicializar el cluster del esclavo para proceder a realizar o correr la réplica de parte del master, con el siguiente comando en otra terminal por aparte, en la carpeta bin de PostgreSQL:

slon slony_rp "dbname=dvdrental user=postgres password=root"

```

[1] Símbolo del sistema - slon.exe slony_rp "dbname=dydental user=postgres password=root"
Microsoft Windows [Versión 10.0.19042.1200]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Usuario>cd C:\Program Files\PostgreSQL\11\bin

C:\Program Files\PostgreSQL\11\bin>slon.exe slony_rp "dbname=dydental user=postgres password=root"
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: slon version 2.2.7 starting up
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option vac_frequency = 3
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option log_level = 0
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option sync_interval = 2000
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option sync_interval_timeout = 10000
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option sync_group_maxsize = 20
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option quit_sync_provider = 0
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option remote_listen_timeout = 300
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option monitor_interval = 500
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option explain_interval = 0
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Integer option apply_cache_size = 100
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option log_pid = 0
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option log_timestamp = 1
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option tcp_keepalive = 1
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option monitor_threads = 1
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option enable_version_check = 1
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Boolean option remote_listen_serializable_transactions = 1
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: Real option real_placeholder = 0.000000
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option cluster_name = slony_rp
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option conn_info = dbname=dydental user=postgres password=root
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option pid_file = [NULL]
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option log_timestamp_format = XY-Xm-Xd Xh:SM:XS KZ
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option archive_dir = [NULL]
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option sql_on_connection = [NULL]
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option lag_interval = [NULL]
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option command_on_logarchive = [NULL]
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: String option cleanup_interval = 10 minutes
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: local node id = 1
2021-10-26 11:09:24 Hora estándar, América Central INFO main: main process started
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: launching sched_start_mainloop
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: loading current cluster configuration
2021-10-26 11:09:24 Hora estándar, América Central CONFIG storeNode: no_id=2 no_comment='slave node'
2021-10-26 11:09:24 Hora estándar, América Central CONFIG storePath: pa_server=2 pa_client=1 pa_conninfo='dbname=slave host=localhost user=postgres password=root' pa_connretry=10
2021-10-26 11:09:24 Hora estándar, América Central CONFIG storeListen: ll_origin=2 ll_receiver=1 ll_provider=2
2021-10-26 11:09:24 Hora estándar, América Central CONFIG storeSet: set_id=1 set_origin=1 set_comment='All pgbench tables'
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: last local event sequence = 50000000035
2021-10-26 11:09:24 Hora estándar, América Central CONFIG main: configuration complete - starting threads
2021-10-26 11:09:24 Hora estándar, América Central INFO localListenThread: thread starts
2021-10-26 11:09:24 Hora estándar, América Central CONFIG version for "dbname=dydental user=postgres password=root" is 111300
2021-10-26 11:09:24 Hora estándar, América Central CONFIG enableNode: no_id=2
2021-10-26 11:09:24 Hora estándar, América Central INFO main: running scheduler mainloop
2021-10-26 11:09:24 Hora estándar, América Central INFO remoteWorkerThread_2: thread starts
2021-10-26 11:09:24 Hora estándar, América Central INFO remoteListenThread_2: thread starts

```

Luego de haber ejecutado en anterior comando, también se ocupa realizar lo mismo con el esclavo, con el siguiente comando, en la carpeta bin de PostgreSQL:

slon slony_rp "dbname=slave user=postgres password=root"

```

[1] Símbolo del sistema - slon.exe slony_rp "dbname=slave user=postgres password=root"
Microsoft Windows [Versión 10.0.19042.1200]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Usuario>cd C:\Program Files\PostgreSQL\11\bin

C:\Program Files\PostgreSQL\11\bin>slon.exe slony_rp "dbname=slave user=postgres password=root"
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: slon version 2.2.7 starting up
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option vac_frequency = 3
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option log_level = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option sync_interval = 2000
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option sync_interval_timeout = 10000
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option sync_group_maxsize = 20
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option quit_sync_provider = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option remote_listen_timeout = 300
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option monitor_interval = 500
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option explain_interval = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option tcp_keepalive_idle = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option tcp_keepalive_interval = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option tcp_keepalive_count = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Integer option apply_cache_size = 100
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option log_pid = 0
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option log_timestamp = 1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option tcp_keepalive = 1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option monitor_threads = 1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option enable_version_check = 1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Boolean option remote_listen_serializable_transactions = 1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: Real option real_placeholder = 0.000000
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option cluster_name = slony_rp
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option conn_info = dbname=slave user=postgres password=root
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option pid_file = [NULL]
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option log_timestamp_format = XY-Xm-Xd Xh:SM:XS KZ
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option archive_dir = [NULL]
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option sql_on_connection = [NULL]
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option lag_interval = [NULL]
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option command_on_logarchive = [NULL]
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: String option cleanup_interval = 10 minutes
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: local node id = 2
2021-10-26 11:10:17 Hora estándar, América Central INFO main: main process started
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: launching sched_start_mainloop
2021-10-26 11:10:17 Hora estándar, América Central CONFIG storeNode: no_id=1 no_comment='Master Node'
2021-10-26 11:10:17 Hora estándar, América Central CONFIG storePath: pa_server=1 pa_client=2 pa_conninfo='dbname=dydental host=localhost user=postgres password=root' pa_connretry=10
2021-10-26 11:10:17 Hora estándar, América Central CONFIG storeListen: ll_origin=1 ll_receiver=2 ll_provider=1
2021-10-26 11:10:17 Hora estándar, América Central CONFIG storeSet: set_id=1 set_origin=1 set_comment='All pgbench tables'
2021-10-26 11:10:17 Hora estándar, América Central MAIN remoteWorker wakeup: node 1 - no worker thread
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: last local event sequence = 5000000001
2021-10-26 11:10:17 Hora estándar, América Central CONFIG main: configuration complete - starting threads
2021-10-26 11:10:17 Hora estándar, América Central INFO localListenThread: thread starts
2021-10-26 11:10:17 Hora estándar, América Central CONFIG version for "dbname=slave user=postgres password=root" is 111300
2021-10-26 11:10:17 Hora estándar, América Central CONFIG enableNode: no_id=1
2021-10-26 11:10:17 Hora estándar, América Central INFO main: running scheduler mainloop
2021-10-26 11:10:17 Hora estándar, América Central INFO remoteWorkerThread_1: thread starts

```

Ya como prueba que la replicación fue exitosa, se procede a insertar datos en alguna tabla, para que de esta manera se pueda apreciar que se haya replicado en la instancia esclava:

PASO I: Se inserta datos en alguna tabla en la instancia master:

Query EditorQuery HistoryExplainNotifications

```
1 CALL public.propr_nuevo_cliente(
2     1::smallint,
3     'A',
4     'B',
5     'ab@gmail.com',
6     1::smallint,
7     1::integer
8 )
```

Data OutputMessages

CALL

Query returned successfully in 73 msec.

✓ Query returned successfully in 73 msec.

PASO II: Se verifica que los datos agregados se hayan insertado en la tabla respectiva:

Query EditorQuery HistoryExplainNotifications

```
1 select * from customer order by customer_id desc;
```

Data OutputMessages

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp without time zone
1	600	1	A	B	ab@gmail.com	1	true	2021-10-26	2021-10-26 11:13:47.6968
2	599	2	Austin	Cintron	austin.cintron@sakilacustomer.org	605	true	2006-02-14	2013-05-26 14:49:45.738
3	598	1	Wade	Delvalle	wade.delvalle@sakilacustomer.org	604	true	2006-02-14	2013-05-26 14:49:45.738
4	597	1	Freddie	Duggan	freddie.duggan@sakilacustomer.org	603	true	2006-02-14	2013-05-26 14:49:45.738
5	596	1	Enrique	Forsythe	enrique.forsythe@sakilacustomer.org	602	true	2006-02-14	2013-05-26 14:49:45.738
6	595	1	Terrence	Gunderson	terrence.gunderson@sakilacustomer.org	601	true	2006-02-14	2013-05-26 14:49:45.738
7	594	1	Eduardo	Hiatt	eduardo.hiatt@sakilacustomer.org	600	true	2006-02-14	2013-05-26 14:49:45.738
8	593	2	Rene	Mcalister	rene.mcalister@sakilacustomer.org	599	true	2006-02-14	2013-05-26 14:49:45.738
9	592	1	Terrance	Roush	terrance.roush@sakilacustomer.org	598	true	2006-02-14	2013-05-26 14:49:45.738
10	591	1	Kent	Arsenault	kent.arsenault@sakilacustomer.org				

✓ Successfully run. Total query runtime: 121 msec. 600 rows affected.

PASO III: Se verifica que en la instancia esclava se haya insertado dichos datos y aparte que los datos que contiene en el master se puedan reflejar en la instancia esclava:

slave/postgres@PostgreSQL 11

[Query Editor](#)
[Query History](#)
[Explain](#)
[Notifications](#)

```

1 select * from customer order by customer_id desc;

```

Data Output

Messages

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp without time zone
1	600	1	A	B	ab@gmail.com	1	true	2021-10-26	2021-10-26 11:13:47.6968
2	599	2	Austin	Cintron	austin.cintron@sakilacustomer.org	605	true	2006-02-14	2013-05-26 14:49:45.738
3	598	1	Wade	Delvalle	wade.delvalle@sakilacustomer.org	604	true	2006-02-14	2013-05-26 14:49:45.738
4	597	1	Freddie	Duggan	freddie.duggan@sakilacustomer.org	603	true	2006-02-14	2013-05-26 14:49:45.738
5	596	1	Enrique	Forsythe	enrique.forsythe@sakilacustomer.org	602	true	2006-02-14	2013-05-26 14:49:45.738
6	595	1	Terrence	Gunderson	terrence.gunderson@sakilacustomer.org	601	true	2006-02-14	2013-05-26 14:49:45.738
7	594	1	Eduardo	Hiatt	eduardo.hiatt@sakilacustomer.org	600	true	2006-02-14	2013-05-26 14:49:45.738
8	593	2	Rene	Mcalister	rene.mcalister@sakilacustomer.org	599	true	2006-02-14	2013-05-26 14:49:45.738
9	592	1	Terrance	Roush	terrance.roush@sakilacustomer.org	598	true	2006-02-14	2013-05-26 14:49:45.738
10	591	1	Kent	Arsenault	kent.arsenault@sakilacustomer.org				

Successfully run. Total query runtime: 141 msec. 600 rows affected.

-- 3. Modelo multidimensional

-- Eliminacion de las tablas

```
drop table dim_film cascade;
```

```
drop table dim_address cascade;
```

```
drop table dim_rental cascade;
```

```
drop table dim_store cascade;
```

```
drop table hechos cascade;
```

/**

* 1. Creacion de las tablas de dimension

*/

-- Dimension Film

```
create table dim_film(
```

id_film integer primary key,

title character varying,


```
    categoria_nombre character varying,  
    actor_nombre character varying  
);
```

-- Dimension Address

```
create table dim_address(  
    address_id integer primary key,  
    city_name character varying,  
    country_name character varying  
);
```

-- Dimension Rental

```
create table dim_rental(  
    rental_id integer primary key,  
    rental_date timestamp without time zone  
);
```

-- Dimension Store

```
create table dim_store(  
    store_id integer primary key  
);
```

/**

* 2. Creacion de la tabla de hechos

*/

```
create table hechos(  
    id_film integer,  
    address_id integer,  
    rental_id integer,  
    store_id integer,  
    numero_alquileres integer,
```

```

        monto_total_alquiler numeric,
foreign key (id_film) references dim_film(id_film),
foreign key (address_id) references dim_address(address_id),
foreign key (rental_id) references dim_rental(rental_id),
foreign key (store_id) references dim_store(store_id)
);

```

```

/**

```

```

* 3. Funciones para inyectar las tablas

```

```

*/

```

```

/**

```

```

* 1. Insertar dimension film.

```

```

* Inserta en la tabla de dim_film la informcion a partir de las tablas del modelo
relacional

```

```

* Entradas: No tiene

```

```

* Salidas: No tiene, pero se actualiza la tabla dim_film

```

```

*/

```

```

create or replace procedure pr_insertar_dim_film()

```

```

language plpgsql as $$

```

```

begin

```

```

    insert into dim_film (id_film, title, categoria_nombre, actor_nombre)

```

```

    select f.film_id, f.title as nombre_pelicula ,

```

```

    c.name as nombre_categoria, STRING_AGG(a.first_name, ', ') from category c

```

```

    -- Se realiza la conexion entre el ID de la pelicula, el nombre de la pelicula

```

```

    -- el nombre de la categoria y los nombres de los actores

```

```

    inner join film_category fm on c.category_id = fm.category_id

```

```

    inner join film f on fm.film_id = f.film_id

```

```

    inner join film_actor fa on f.film_id = fa.film_id

```

```

    inner join actor a on fa.actor_id = a.actor_id

```

```

        group by f.film_id, f.title, c.name
        order by f.film_id, f.title, c.name;

        -- Se confirman los cambios
        commit;

end;$$

call pr_insertar_dim_film();
select * from dim_film;

/**
 * 2. Insertar dimension address.
 * Inserta en la tabla de dim_address la informcion a partir de las tablas del modelo
relacional
 * Entradas: No tiene
 * Salidas: No tiene, pero se actualiza la tabla dim_address
 */
create or replace procedure pr_insertar_dim_address()
language plpgsql as $$
begin

    insert into dim_address(address_id, city_name, country_name)
    select co.country_id, STRING_AGG(ci.city, ', '), co.country as nombre_pais
    from address a

        -- Se realiza la conexion entre el ID del pais, el nombre del pais
        -- y el nombre de las ciudades
    inner join city ci on a.city_id = ci.city_id
    inner join country co on ci.country_id = co.country_id

```

```

        group by co.country_id, co.country
        order by co.country_id, co.country;

        -- Se confirman los cambios
        commit;

end;$$

call pr_insertar_dim_address();
select * from dim_address;

/**
 * 3. Insertar dimension rental.
 * Inserta en la tabla de dim_rental la informcion a partir de las tablas del modelo
relacional
 * Entradas: No tiene
 * Salidas: No tiene, pero se actualiza la tabla dim_rental
 */
create or replace procedure pr_ingresar_dim_rental()
language plpgsql as $$
begin

        -- Inserta los valores de ID del alquiler y la fecha del alquiler
        insert into dim_rental(rental_id, rental_date)
        select r.rental_id, r.rental_date from rental r;

        -- Se confirman los cambios
        commit;

end;$$

call pr_ingresar_dim_rental();

```

```
select * from dim_rental;
```

```
/**
```

```
 * 4. Insertar dimension store.
```

```
 * Inserta en la tabla de dim_store la informcion a partir de las tablas del modelo  
relacional
```

```
 * Entradas: No tiene
```

```
 * Salidas: No tiene, pero se actualiza la tabla dim_store
```

```
 */
```

```
create or replace procedure pr_ingresar_dim_store ()
```

```
language plpgsql as $$
```

```
begin
```

```
    -- Inserta los valores de ID de la sucursal
```

```
    insert into dim_store(store_id)
```

```
    select store_id from store;
```

```
    -- Se confirman los cambios
```

```
    commit;
```

```
end;$$
```

```
call pr_ingresar_dim_store();
```

```
select * from dim_store;
```

```
/**
```

```
 * 5. Insertar tabla hechos.
```

```
 * Inserta en la tabla de hechos la informcion a partir de las tablas del modelo  
relacional
```

```
    y las tablas de dimension
```

```
 * Entradas: No tiene
```

```
 * Salidas: No tiene, pero se actualiza la tabla de hechos
```

```
 */
```

```

create or replace procedure pr_ingresar_hechos()
language plpgsql as $$
begin

    insert into hechos (id_film, address_id, rental_id, store_id, numero_alquileres,
monto_total_alquiler)
    select fh.id_film, c.country_id, rh.rental_id, s.store_id,
    count(r.rental_id), sum(p.amount)
    from dim_film fh

    -- Se realiza la conexion entre las tablas del modelo transaccional ya que tienen
los mismos

    -- ID que en las tablas de dimension
    inner join film f on fh.id_film = f.film_id
    inner join inventory i on i.film_id = f.film_id
    inner join store s on s.store_id = i.store_id
    inner join rental r on r.inventory_id = i.inventory_id
    inner join dim_rental rh on rh.rental_id = r.rental_id
    inner join payment p on p.rental_id = rh.rental_id
    inner join address a on s.address_id = a.address_id
    inner join city ci on ci.city_id = a.city_id
    inner join country c on c.country_id = ci.country_id

    group by fh.id_film, f.film_id, f.title, i.inventory_id, s.store_id, rh.rental_id,
    c.country_id;

    -- Se confirman los cambios
    commit;

end;$$

call pr_ingresar_hechos();

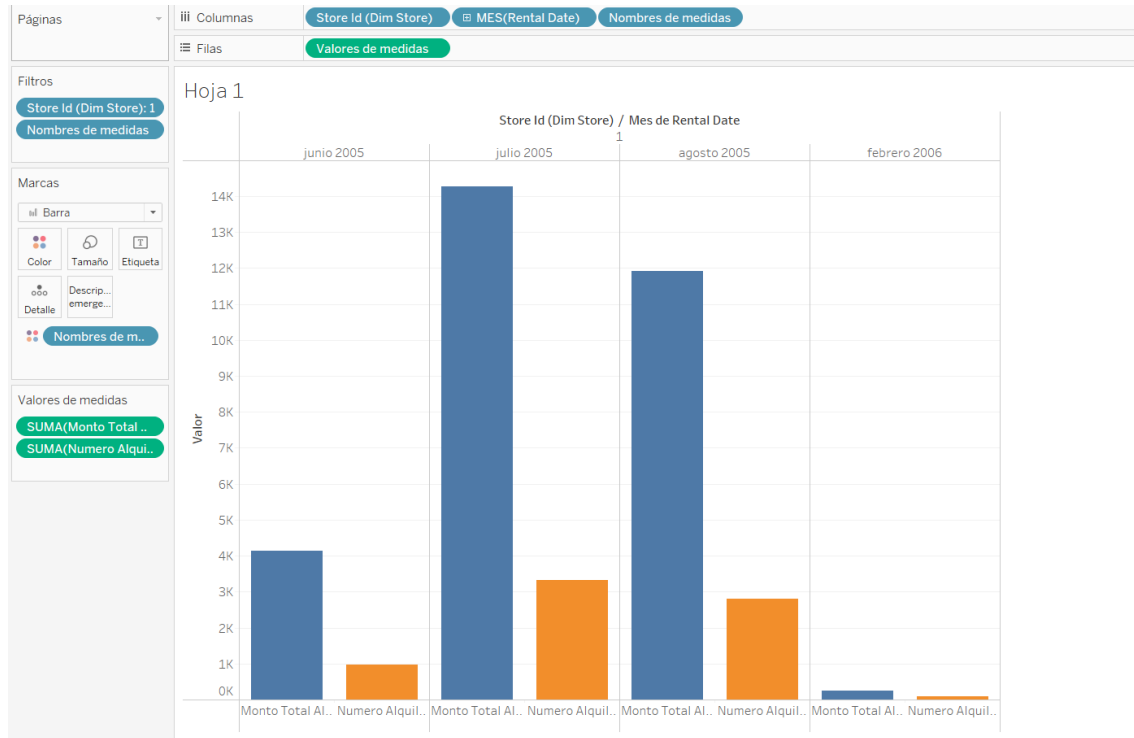
select * from hechos;

```

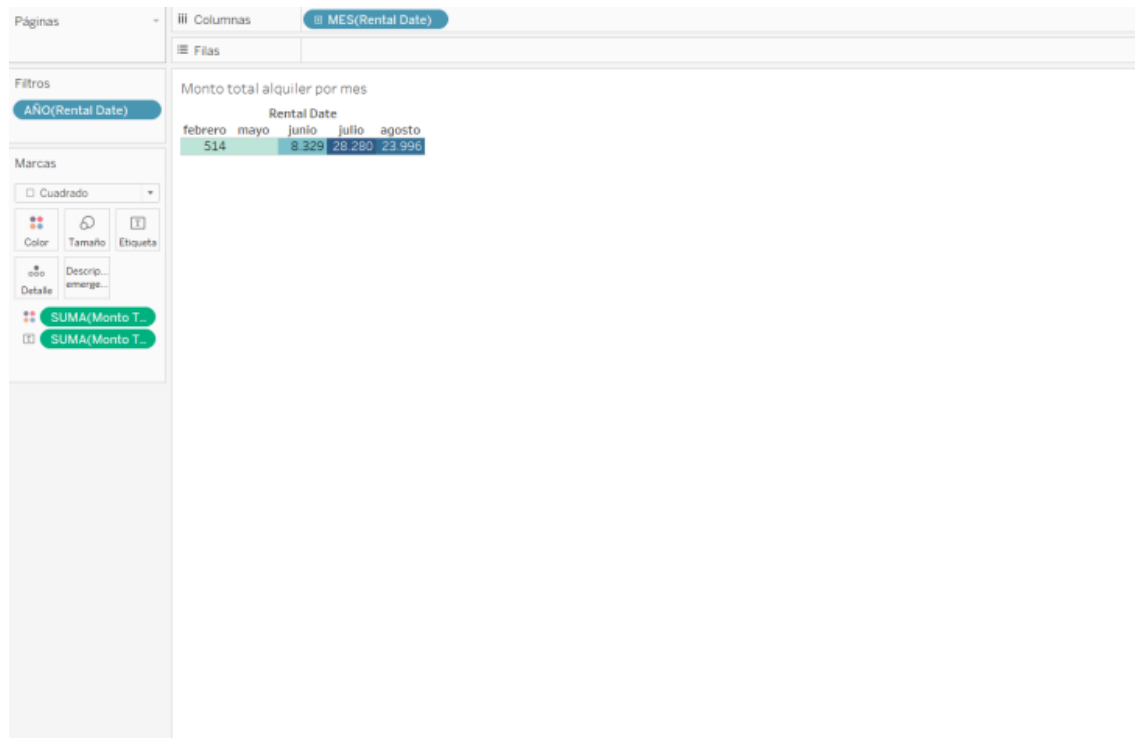
-- 4. Visualización y acceso por medio de interfaz gráfica al modelo estrella

Hojas donde se representen lo solicitado sobre el punto de visualización:

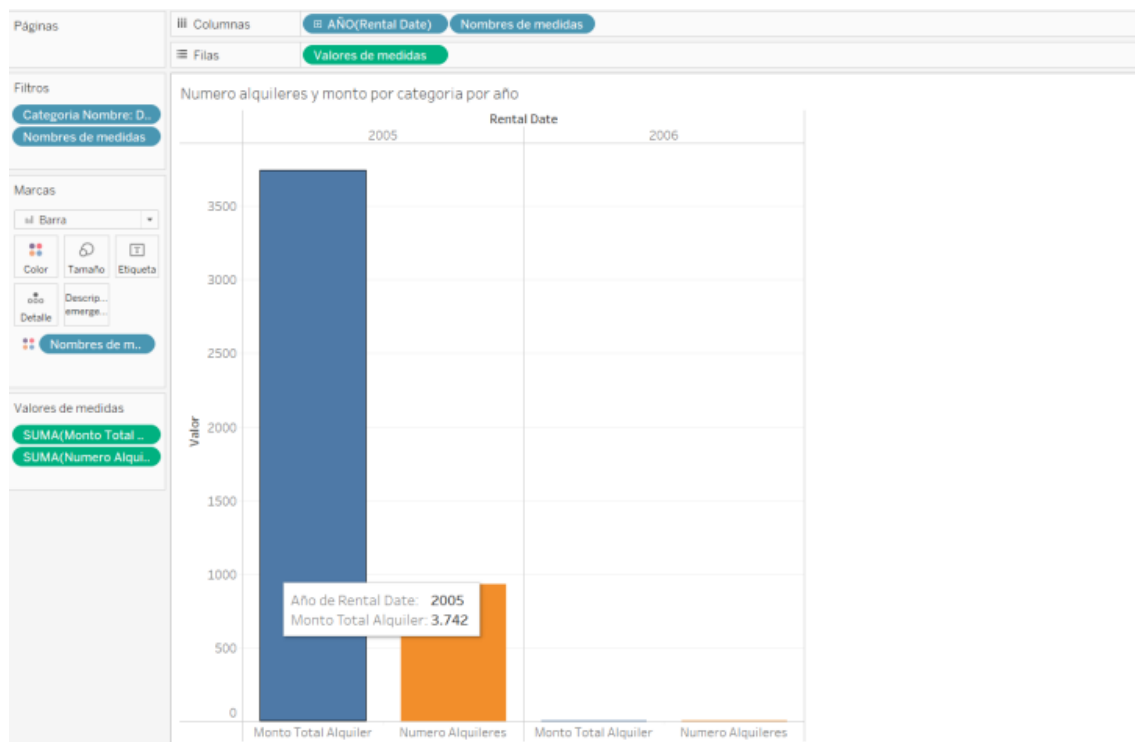
1. Para una sucursal (a seleccionar por el usuario), grafique el número de alquileres realizados y el monto cobrado por mes, sin importar el año:



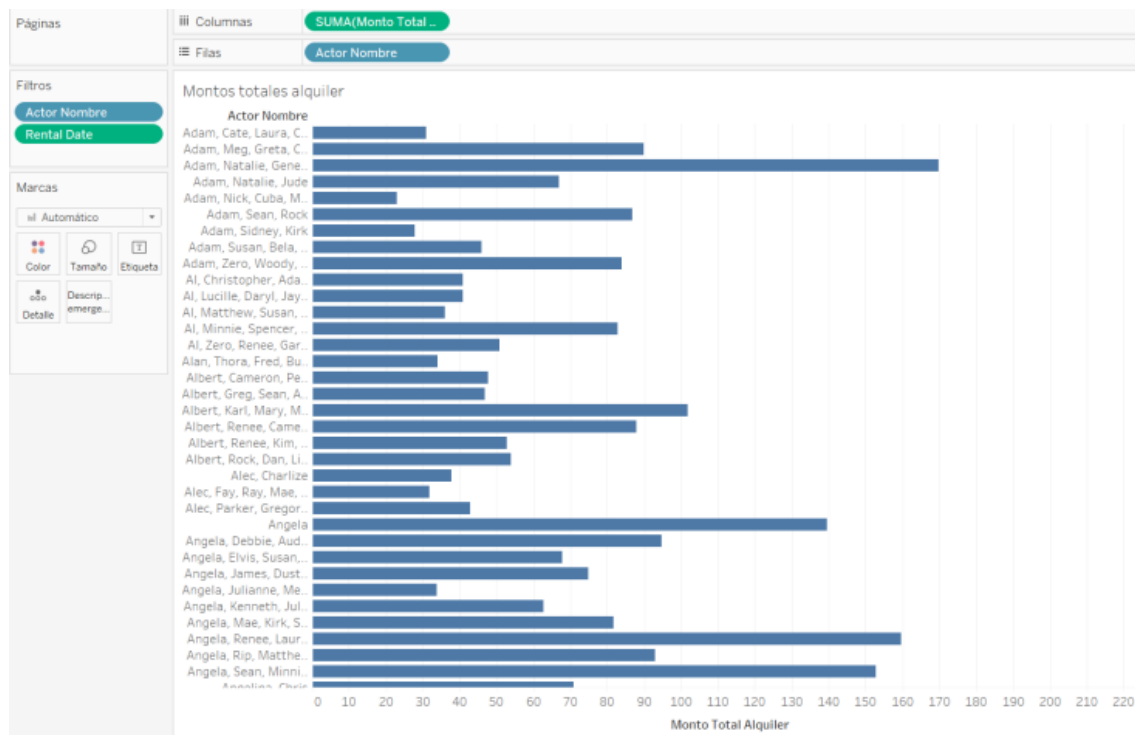
2. Graficar para un año (a seleccionar por el usuario) los montos cobrados por alquileres por mes:



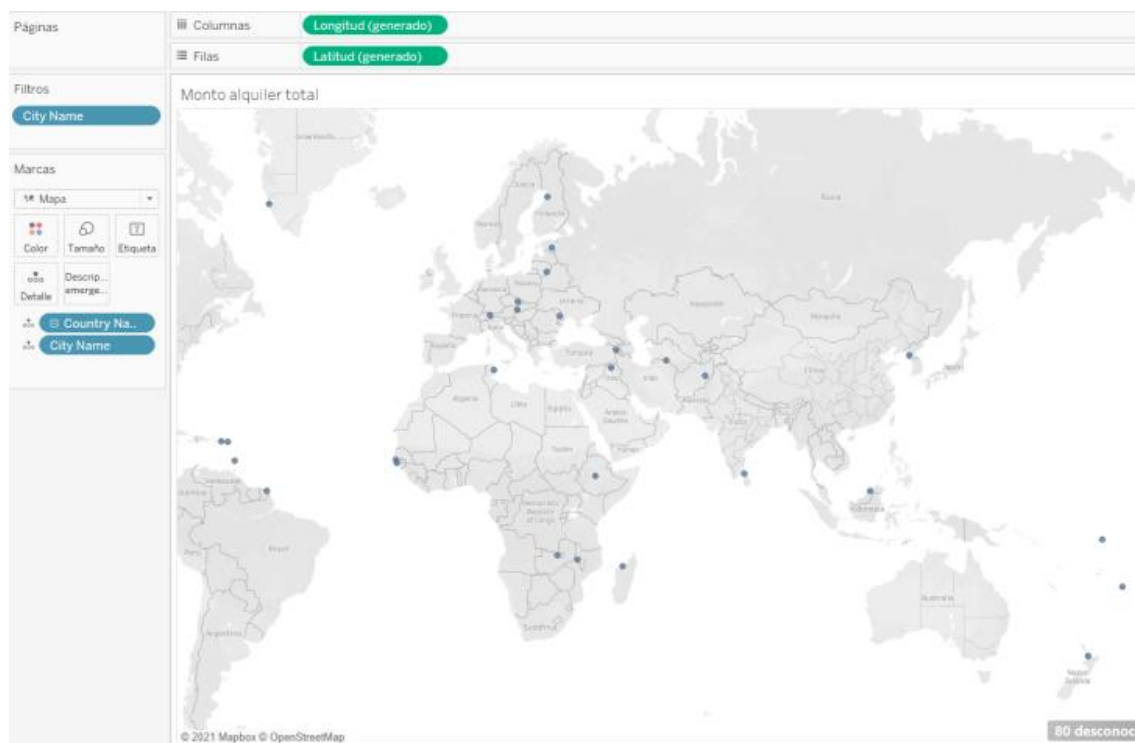
3. Para una categoría de película (a seleccionar por el usuario), graficar el número de alquileres y el monto cobrado por año:



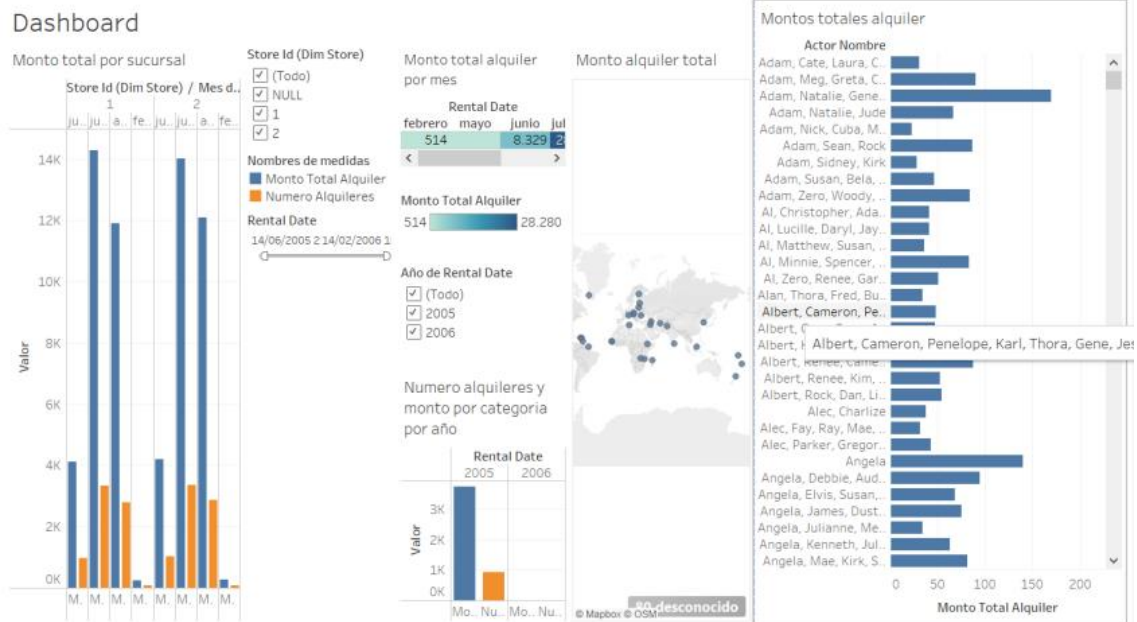
4. Para los 10 actores con más alquileres, graficar los montos totales de alquileres por año (a seleccionar por el usuario). Incluya la opción de todos los años:



5. Despliegue un mapa de ciudades que presente por año el monto de alquiler total representado por el tamaño del punto sobre la ciudad:



El respectivo DASHBOARD:



Conclusión:

Con lo que respecta al proyecto como tal nos dio una perspectiva más aterrizada con lo que respecta a la inteligencia de negocios y los objetivos que tiene como tal, adicionalmente el tema de replicación nos hace pensar sobre las extensas posibilidades que esta puedan contener con respecto a otros métodos de replicación ya que lo que hicimos fue una pequeña prueba de como poder hacer algo a pequeña escala a comparación a la inteligencia de negocios de una empresa, lo de Tableau nos hizo darnos cuenta que poder graficar todo el conjunto de datos puede ser bastante enriquecedor en cuanto respecta a la información que se quiera investigar, o bien la información que el cliente tenerle prioridad, ya que dependiendo de cómo se realice el análisis de las dimensiones y medidas pueden llegar a repercutir a información errónea, o bien información no solicitada.

Referencias:

La referencia nos fue útil para desarrollar la replicación en este caso con Slony

- <https://www.youtube.com/watch?v=Eh6nGFaq4AU>

La referencia nos fue útil para poder instalar correctamente PostgreSQL, para poder realizar la replicación de Slony

- https://www.youtube.com/watch?v=2TOjA_Dw1kw

Este vídeo nos fue de utilidad para poder informarnos de cómo realizar un modelo multidimensional

- <https://www.youtube.com/watch?v=RXElduq3jP0>

Este vídeo nos sirvió para saber cómo conectar Tableau con el PostgreSQL

- <https://www.youtube.com/watch?v=HL49JwM0vKA>

Este vídeo nos sirvió para saber como manejar las dimensiones, medidas y graficas en Tableau

- <https://www.youtube.com/watch?v=xhpHlai8TKY&t>

Este vídeo nos fue de utilidad para poder diseñar de una mejor manera el modelo estrella en el modelo multidimensional:

- <https://www.youtube.com/watch?v=tIVHCvZQHo>

Para desarrollar la replicación correctamente con Slony se consultó está referencia para consultar comandos útiles para desarrollar la misma:

- <https://slony.info/documentation/1.2/firstdb.html>