



**Universidad Autónoma del Estado de México**  
**Unidad Académica Profesional Tianguistenco**

**Ingeniería en software**

**Unidad de aprendizaje:**

**Datawarehouse**

**Profesor:**

Julieta Garcilazo Reyes

**Alumno:**

Diego Hernández Romero

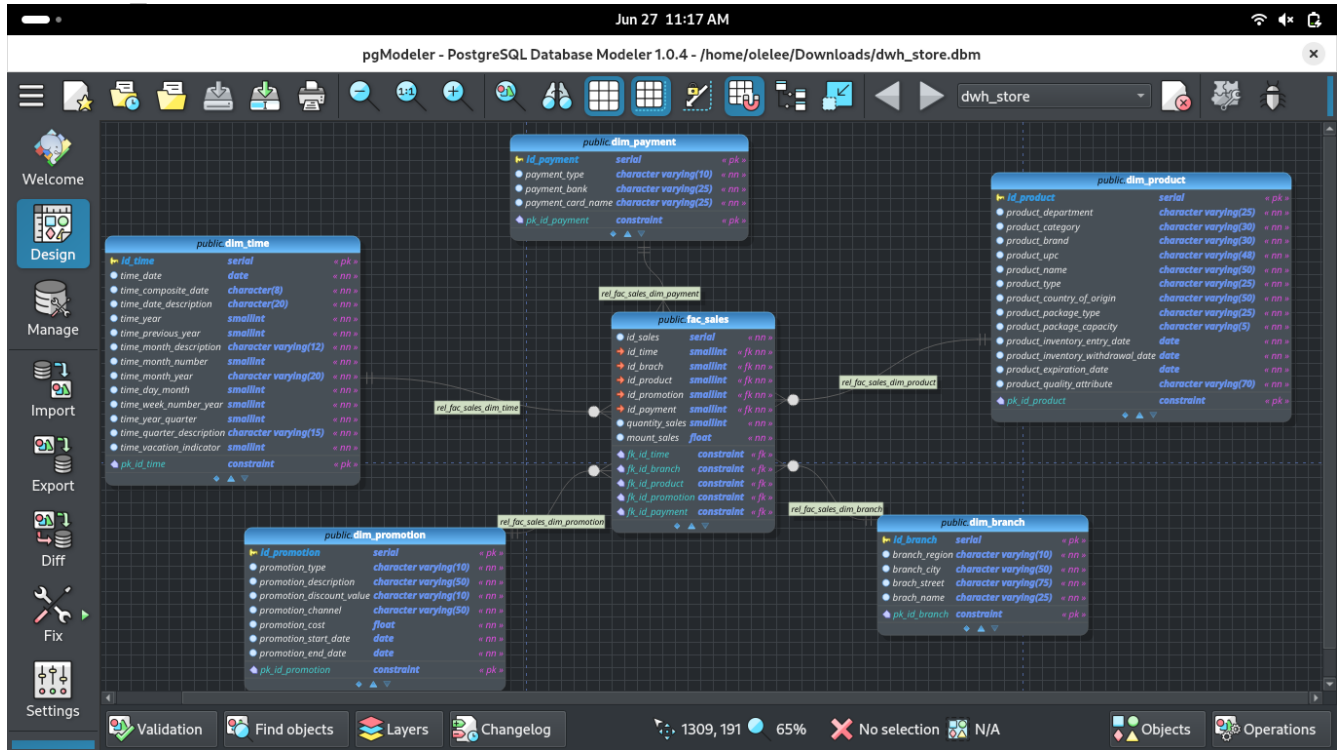
**Grupo:**

S7

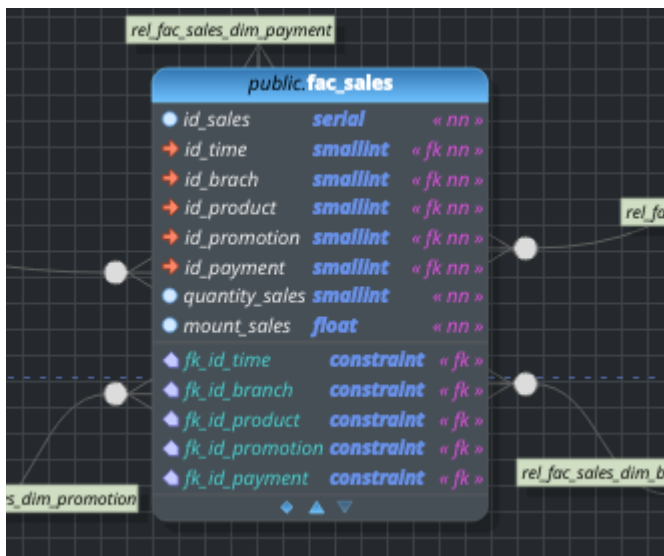
**Fecha de entrega: 27/06/2025**

Como primer paso se utilizó pgmodeler para moldear las dimensiones conforme a los diferentes puntos que se obtuvieron del documento donde se establecía la información a obtener.

Dadas las características se optó por realizar 5 dimensiones junto con 1 de hechos, estas dimensiones son: dim\_time, dim\_promotion, dim\_branch, dim\_product, dim\_payment y fact sales.



Dentro de estas dimensiones se establecieron los tipos de datos, las llaves y relaciones y los diferentes puntos a obtener, además de que tiene comentarios sobre cada punto de las dimensiones, a continuación, cada una de las dimensiones con sus características.



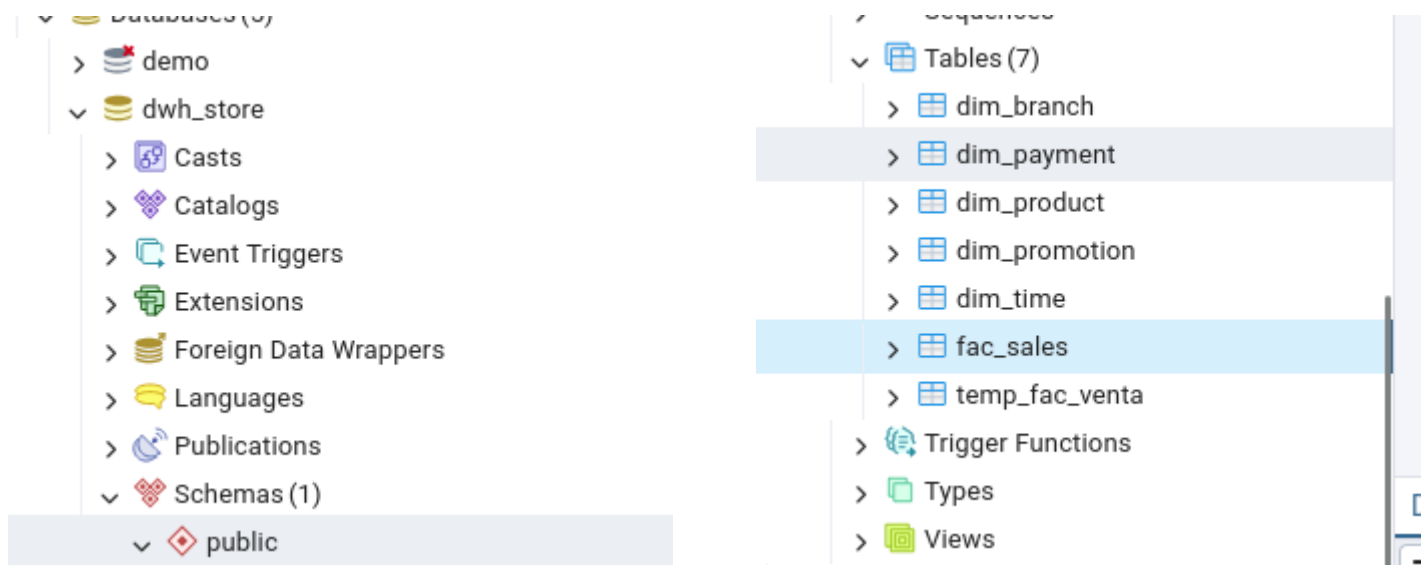
public.dlm_time		
<b>Id_time</b>	serial	« pk »
time_date	date	« nn »
time_composite_date	character(8)	« nn »
time_date_description	character(20)	« nn »
time_year	smallint	« nn »
time_previous_year	smallint	« nn »
time_month_description	character varying(12)	« nn »
time_month_number	smallint	« nn »
time_month_year	character varying(20)	« nn »
time_day_month	smallint	« nn »
time_week_number_year	smallint	« nn »
time_year_quarter	smallint	« nn »
time_quarter_description	character varying(15)	« nn »
time_vacation_indicator	smallint	« nn »
pk_id_time	constraint	« pk »

public.dlm_product		
<b>Id_product</b>	serial	« pk »
product_department	character varying(25)	« nn »
product_category	character varying(30)	« nn »
product_brand	character varying(30)	« nn »
product_upc	character varying(48)	« nn »
product_name	character varying(50)	« nn »
product_type	character varying(25)	« nn »
product_country_of_origin	character varying(50)	« nn »
product_package_type	character varying(25)	« nn »
product_package_capacity	character varying(5)	« nn »
product_inventory_entry_date	date	« nn »
product_inventory_withdrawal_date	date	« nn »
product_expiration_date	date	« nn »
product_quality_attribute	character varying(70)	« nn »
pk_id_product	constraint	« pk »

public.dlm_payment		
<b>Id_payment</b>	serial	« pk »
payment_type	character varying(10)	« nn »
payment_bank	character varying(25)	« nn »
payment_card_name	character varying(25)	« nn »
pk_id_payment	constraint	« pk »

public.dlm_branch		
<b>Id_branch</b>	serial	« pk »
branch_region	character varying(10)	« nn »
branch_city	character varying(50)	« nn »
branch_street	character varying(75)	« nn »
branch_name	character varying(25)	« nn »
pk_id_branch	constraint	« pk »

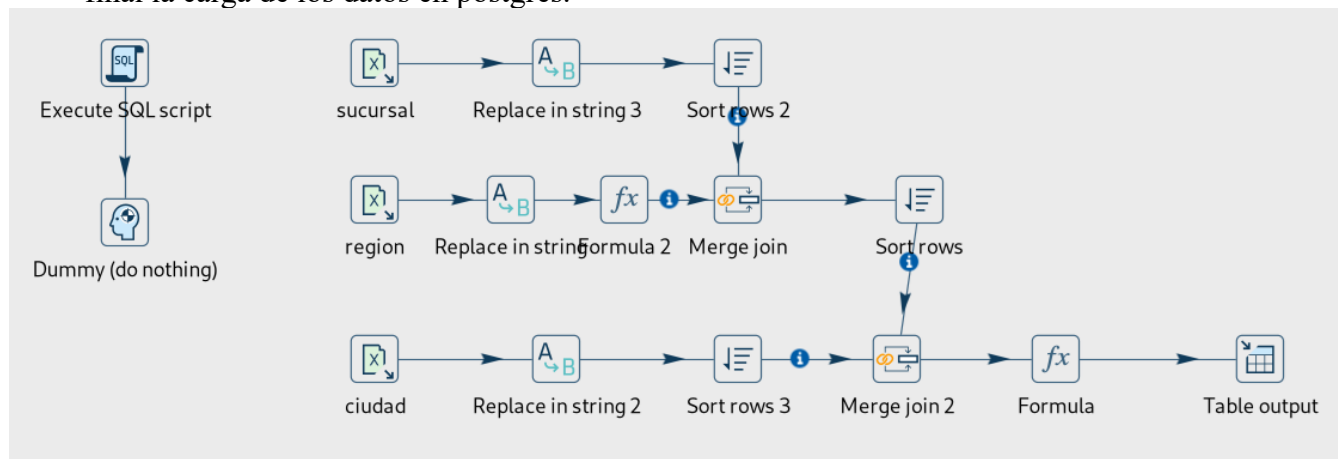
Al tener dicho modelado procedemos a conectarnos a postgres para realizar la importación de las dimensiones, esto fue bastante sencillo y quedo de esta forma dentro de postgres.



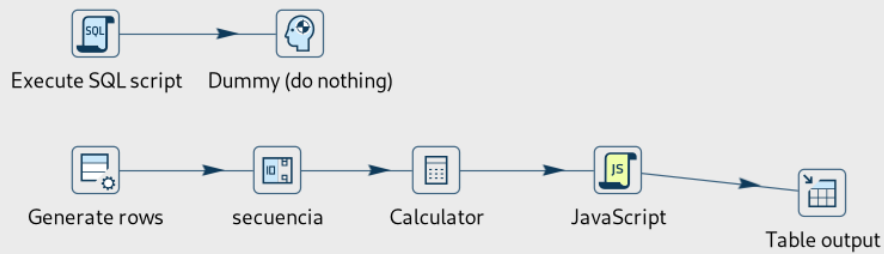
Además, se adjuntó una tabla extra temporal exactamente igual a la de fact\_sales para hacer la carga de datos posteriormente.

Una vez cargadas las dimensiones procedemos a realizar la carga de los datos conforme lo indicado en cada una de las dimensiones.

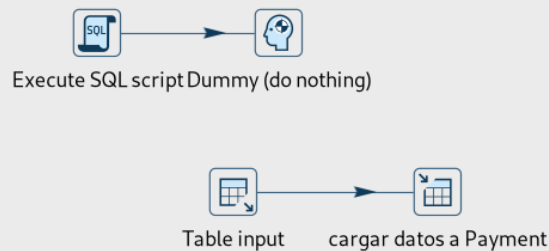
Se empezó con la carga de la dim\_branch, esta se realizó con tres documentos de excel, donde se unieron respectivamente con merge join mediante un pipeline en apache hop y al final la carga de los datos en postgres.



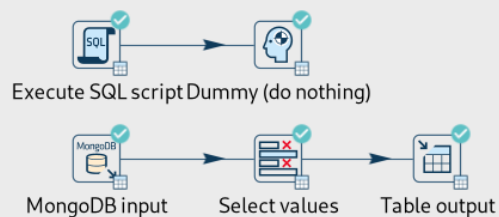
Para la dim\_time se utilizó igualmente apache hop, pero esta vez se utilizó la generación de filas para generación de datos mediante secuencias, calculadora y javascript, el resultado se cargó en la dimensión dim\_time y este fue el pipeline.



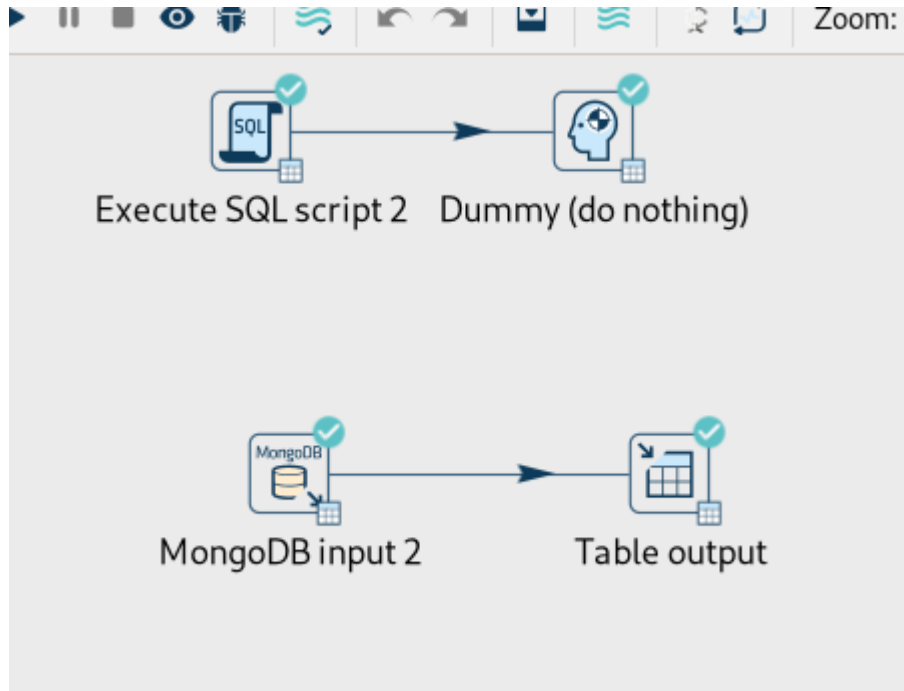
Para dim\_payment se utilizó apache hop, pero esta vez se utilizó la obtención de los datos de otra tabla de una base diferente y se utilizó dicha información para cargarla en la dim\_payment.



Para dim\_promotion se utilizó la carga de datos a través de mongoDB, la cual es a través de una tabla de mongo y se realizó lo mismo con el anterior, pero esta vez a través de mongo.



Para dim\_product se utilizó igual la carga a través de mongoDB, igual mediante la colección productos, esto se realizó mediante un pipeline.



Para la carga final de fact\_sales se realizó mediante un excel donde se cargaron 50 filas para diferentes años, además se utilizó dicho documento para realizar una carga en una tabla temporal, donde se relacionaban todos esos datos y se cargó con un pipeline, después se cargaron a fact\_sales con otro pipeline y se pudieron generar las vistas

Jun 27 2:00 PM

Hop

Project: Fact\_sales Environment: e\_ e\_ e\_

filename x

Zoom: 100% Unit test:

Metrics Logging

#	Transform Name	Copy	Input	Read	Written	Output	Updated	Rejected	Errors	Buffers Input	Buffers Output	Duration	Speed	Status
1	2017	0	50	0	50	0	0	0	0	0	0	0.015"	185	Finished
2	Table output	0	0	530	530	530	0	0	0	0	0	0.465"	686	Finished
3	2018	0	50	0	50	0	0	0	0	0	0	0.027"	167	Finished
4	2019	0	50	0	50	0	0	0	0	0	0	0.013"	179	Finished
5	2020	0	50	0	50	0	0	0	0	0	0	0.074"	133	Finished
6	Execute SQL script	0	0	0	1	0	0	0	0	0	0	0.000"	33	Finished

Después se generaron las vistas y la inserción en fac\_sales para generar ya los reportes finales.

A continuación los resultados

The screenshot shows a software interface for executing SQL scripts. At the top, the window title is "Execute SQL script". Below the title bar, there are two input fields: "Transform name" with the value "Execute SQL script" and "Connection" with the value "fact\_sales\_2". A note below these fields states: "SQL script to execute. (statements separated by ;) Question marks will be replaced by arguments." The main text area contains two SQL statements. The first is an INSERT statement into the "fac\_sales" table. The second is a CREATE OR REPLACE VIEW statement for "reporte\_ventas\_marzo\_2025". To the left of the main text area, there is a "Parameters" panel with a table header "Field" and a single row with the value "1". At the bottom of the window, there are three buttons: "OK", "Cancel", and "Connect".

Transform name: Execute SQL script

Connection: fact\_sales\_2

SQL script to execute. (statements separated by ;) Question marks will be replaced by arguments.

```
INSERT INTO fac_sales (id_payment, id_promotion, id_brach, id_product, id_time, quantity_sales, mount_sales)
SELECT
    id_pago,
    id_promocion,
    id_sucursal,
    id_producto,
    id_tiempo,
    cantidad,
    total

CREATE OR REPLACE VIEW reporte_ventas_marzo_2025 AS
SELECT
    COALESCE(SUM(quantity_sales), 0) AS cantidad_total,
    COALESCE(SUM(mount_sales), 0) AS monto_total
FROM fac_sales cc
LEFT JOIN dim_time cd ON cc.id_time = cd.id_time
WHERE cd.time_month_description = 'Marzo' AND cd.time_year = 2025;
```

Parameters:

Field
1

Buttons: OK, Cancel, Connect