

# Estrategia - Decisiones tomadas

En el presente informe se detallarán las decisiones de diseño tomadas para la creación del nuevo Schema normalizado y la migración de los datos.

Los pasos a seguir son los siguientes:

## Entrega 2:

### Creación de tablas:

1. Se decidió crear primero las tablas que no poseen claves foráneas, es decir, aquellas entidades independientes que no dependen de otras para su definición.
2. Una vez creadas, se procederá a generar el resto de las tablas, respetando el orden lógico de dependencias entre ellas para garantizar la integridad referencial desde el inicio.

### Migración de tabla maestra:

Este punto se va a desarrollar tabla por tabla. La idea es primero realizar una consulta manual a la Tabla Maestra para obtener los datos necesarios y luego generar la Query de inserción de los datos.

Por ejemplo, Provincia:

```

SELECT DISTINCT Sucursal_Provincia
FROM gd_esquema.Maestra
UNION
SELECT DISTINCT Cliente_Provincia
FROM gd_esquema.Maestra
UNION
SELECT DISTINCT Proveedor_Provincia
FROM gd_esquema.Maestra;

```

121 %

Results Messages

	Sucursal_Provincia
3	Capital Federal
4	Catamarca
5	Chaco
6	Chubut
7	Cordoba
8	Corrientes

Vemos que tenemos datos erróneos:

22	Santa Fe
23	Santa; Del Estero
24	Tierra Del Fue;
25	Tucuman

Los normalizamos y generamos Query de inserción:

```
PRINT 'Insertando datos en tabla Provincia...';

=====

INSERT INTO Provincia(nombre_prov)
SELECT DISTINCT
CASE
    WHEN p = 'Santia; Del Estero' THEN 'Santiago del Estero'
    WHEN p = 'Tierra Del Fue;' THEN 'Tierra del Fuego'
    ELSE p
END
FROM (
    SELECT Sucursal_Provincia AS p FROM gd_esquema.Maestra
    UNION
    SELECT Cliente_Provincia FROM gd_esquema.Maestra
    UNION
    SELECT Proveedor_Provincia FROM gd_esquema.Maestra
) AS todas
WHERE p is not null;

=====
-- Inserts para tabla Localidad
```

121 %

Messages

Insertando datos en tabla Provincia...

(24 rows affected)

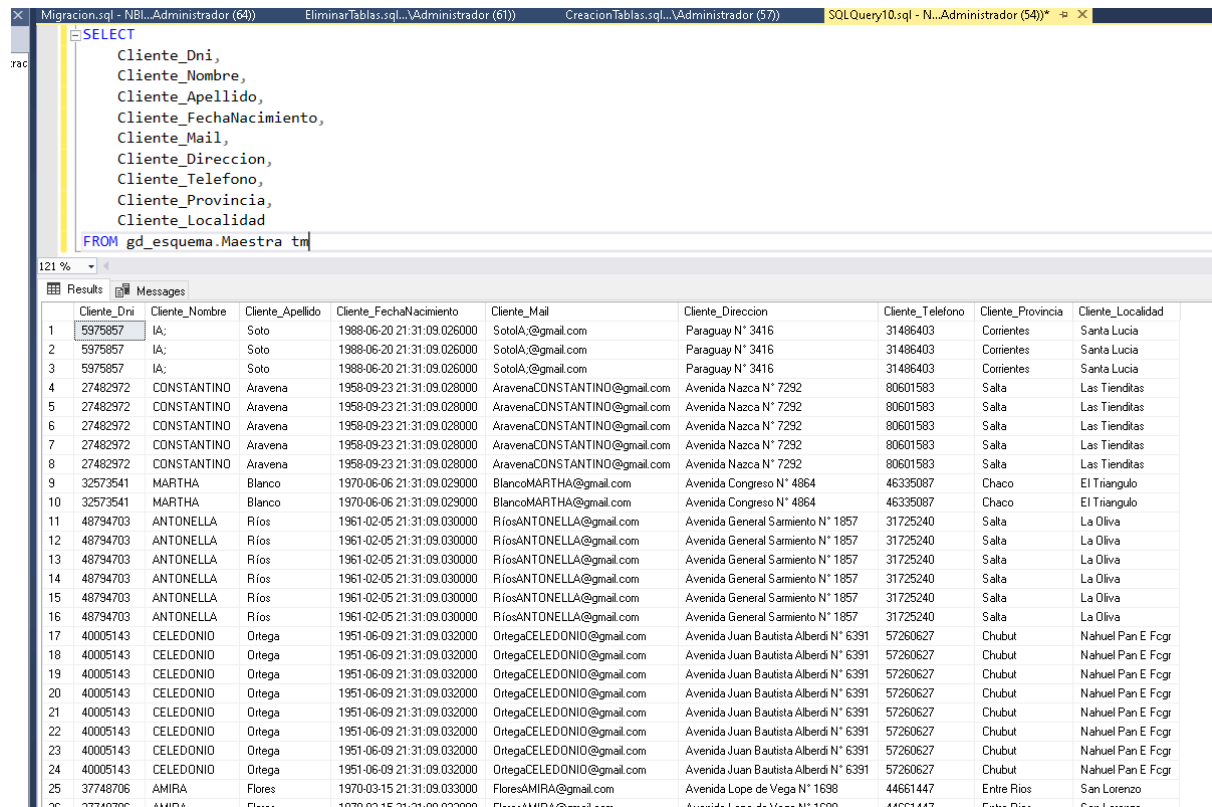
Completion time: 2025-05-31T23:42:25.1369087-03:00

En este documento se van a detallar los casos particulares.  
Los que sean similares o tengan una solución similar se ignorarán

Ejemplo: Para Localidad también debemos realizar un Union de 3  
consultas.

Caso tabla clientes:

Al consultar los datos de los clientes en la tabla maestra, vemos que tenemos muchos registros repetidos múltiples veces::



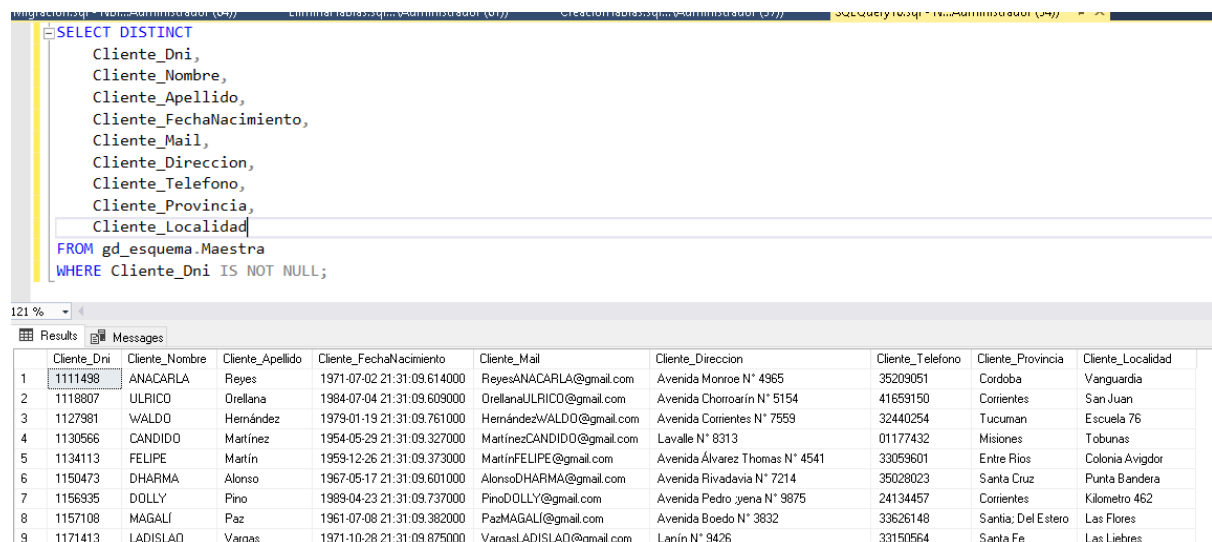
The screenshot shows a SQL query window with the following query:

```
SELECT
    Cliente_Dni,
    Cliente_Nombre,
    Cliente_Apellido,
    Cliente_FechaNacimiento,
    Cliente_Mail,
    Cliente_Direccion,
    Cliente_Telefono,
    Cliente_Provincia,
    Cliente_Localidad
FROM gd_esquema.Maestra tm
```

The results pane shows a table with 10 columns: Cliente\_Dni, Cliente\_Nombre, Cliente\_Apellido, Cliente\_FechaNacimiento, Cliente\_Mail, Cliente\_Direccion, Cliente\_Telefono, Cliente\_Provincia, and Cliente\_Localidad. The table contains 26 rows, with the first row (DNI 5975857) appearing 3 times and the last row (DNI 37748706) appearing 1 time. The data is as follows:

Cliente_Dni	Cliente_Nombre	Cliente_Apellido	Cliente_FechaNacimiento	Cliente_Mail	Cliente_Direccion	Cliente_Telefono	Cliente_Provincia	Cliente_Localidad
5975857	IA:	Soto	1988-06-20 21:31:09.026000	SotoIA@gmail.com	Paraguay N° 3416	31486403	Corrientes	Santa Lucia
5975857	IA:	Soto	1988-06-20 21:31:09.026000	SotoIA@gmail.com	Paraguay N° 3416	31486403	Corrientes	Santa Lucia
5975857	IA:	Soto	1988-06-20 21:31:09.026000	SotoIA@gmail.com	Paraguay N° 3416	31486403	Corrientes	Santa Lucia
27482972	CONSTANTINO	Aravena	1958-09-23 21:31:09.028000	AravenaCONSTANTINO@gmail.com	Avenida Nazca N° 7292	80601583	Salta	Las Tienditas
27482972	CONSTANTINO	Aravena	1958-09-23 21:31:09.028000	AravenaCONSTANTINO@gmail.com	Avenida Nazca N° 7292	80601583	Salta	Las Tienditas
27482972	CONSTANTINO	Aravena	1958-09-23 21:31:09.028000	AravenaCONSTANTINO@gmail.com	Avenida Nazca N° 7292	80601583	Salta	Las Tienditas
27482972	CONSTANTINO	Aravena	1958-09-23 21:31:09.028000	AravenaCONSTANTINO@gmail.com	Avenida Nazca N° 7292	80601583	Salta	Las Tienditas
27482972	CONSTANTINO	Aravena	1958-09-23 21:31:09.028000	AravenaCONSTANTINO@gmail.com	Avenida Nazca N° 7292	80601583	Salta	Las Tienditas
32573541	MARTHA	Blanco	1970-06-06 21:31:09.029000	BlancoMARTHA@gmail.com	Avenida Congreso N° 4864	46335087	Chaco	El Triangulo
32573541	MARTHA	Blanco	1970-06-06 21:31:09.029000	BlancoMARTHA@gmail.com	Avenida Congreso N° 4864	46335087	Chaco	El Triangulo
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
48794703	ANTONELLA	Rios	1961-02-05 21:31:09.030000	RiosANTONELLA@gmail.com	Avenida General Sarmiento N° 1857	31725240	Salta	La Oliva
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
40005143	CELEDONIO	Ortega	1951-06-09 21:31:09.032000	OrtegaCELEDONIO@gmail.com	Avenida Juan Bautista Alberdi N° 6391	57260627	Chubut	Nahuel Pan E Fcgr
37748706	AMIRA	Flores	1970-03-15 21:31:09.033000	FloresAMIRA@gmail.com	Avenida Lope de Vega N° 1698	44661447	Entre Rios	San Lorenzo
37748706	AMIRA	Flores	1970-03-15 21:31:09.033000	FloresAMIRA@gmail.com	Avenida Lope de Vega N° 1698	44661447	Entre Rios	San Lorenzo

Para resolver esto podemos hacer un DISTINCT por DNI:



The screenshot shows a SQL query window with the following query:

```
SELECT DISTINCT
    Cliente_Dni,
    Cliente_Nombre,
    Cliente_Apellido,
    Cliente_FechaNacimiento,
    Cliente_Mail,
    Cliente_Direccion,
    Cliente_Telefono,
    Cliente_Provincia,
    Cliente_Localidad
FROM gd_esquema.Maestra
WHERE Cliente_Dni IS NOT NULL;
```

The results pane shows a table with 10 columns: Cliente\_Dni, Cliente\_Nombre, Cliente\_Apellido, Cliente\_FechaNacimiento, Cliente\_Mail, Cliente\_Direccion, Cliente\_Telefono, Cliente\_Provincia, and Cliente\_Localidad. The table contains 9 rows, each representing a unique client. The data is as follows:

Cliente_Dni	Cliente_Nombre	Cliente_Apellido	Cliente_FechaNacimiento	Cliente_Mail	Cliente_Direccion	Cliente_Telefono	Cliente_Provincia	Cliente_Localidad
1111498	ANACARLA	Reyes	1971-07-02 21:31:09.614000	ReyesANACARLA@gmail.com	Avenida Monroe N° 4965	35209051	Cordoba	Vanguardia
1118807	ULRICO	Orellana	1984-07-04 21:31:09.609000	OrellanaULRICO@gmail.com	Avenida Chorroarín N° 5154	41659150	Corrientes	San Juan
1127981	WALDO	Hernández	1979-01-19 21:31:09.761000	HernándezWALDO@gmail.com	Avenida Corrientes N° 7559	32440254	Tucuman	Escuela 76
1130566	CANDIDO	Martínez	1954-05-29 21:31:09.327000	MartínezCANDIDO@gmail.com	Lavalle N° 8313	01177432	Misiones	Tobunas
1134113	FELIPE	Martín	1959-12-26 21:31:09.373000	MartínFELIPE@gmail.com	Avenida Álvarez Thomas N° 4541	33059601	Entre Rios	Colonia Avigdor
1150473	DHARMA	Alonso	1967-05-17 21:31:09.601000	AlonsoDHARMA@gmail.com	Avenida Rivadavia N° 7214	35028023	Santa Cruz	Punta Bandera
1156935	DOLLY	Pino	1989-04-23 21:31:09.737000	PinoDOLLY@gmail.com	Avenida Pedro yena N° 9875	24134457	Corrientes	Kilometro 462
1157108	MAGALI	Paz	1961-07-08 21:31:09.382000	PazMAGALI@gmail.com	Avenida Boedo N° 3832	33626148	Santa; Del Estero	Las Flores
1171413	LADISLAO	Vargas	1971-10-28 21:31:09.875000	VargasLADISLAO@gmail.com	Lanín N° 9426	33150564	Santa Fe	Las Liebres

Luego una IA nos recomienda usar una partición por DNI para asegurarnos de que no haya duplicados. Ya que si bien a simple vista parece que no hay duplicados. Podría haberlos si algún campo para un mismo DNI tiene alguna variación.

Tenemos dos opciones:

### ¿Cuándo usar cada uno?

- Usá `DISTINCT` si **todas las columnas de un dni son idénticas** (verificalo antes).
- Usá `ROW_NUMBER()` si hay **diferencias sutiles** entre filas del mismo `dni` y querés aplicar una lógica clara para seleccionar una.

Elegimos utilizar `DISTINCT` por la simplicidad, pero primero verificamos que no haya registros duplicados:

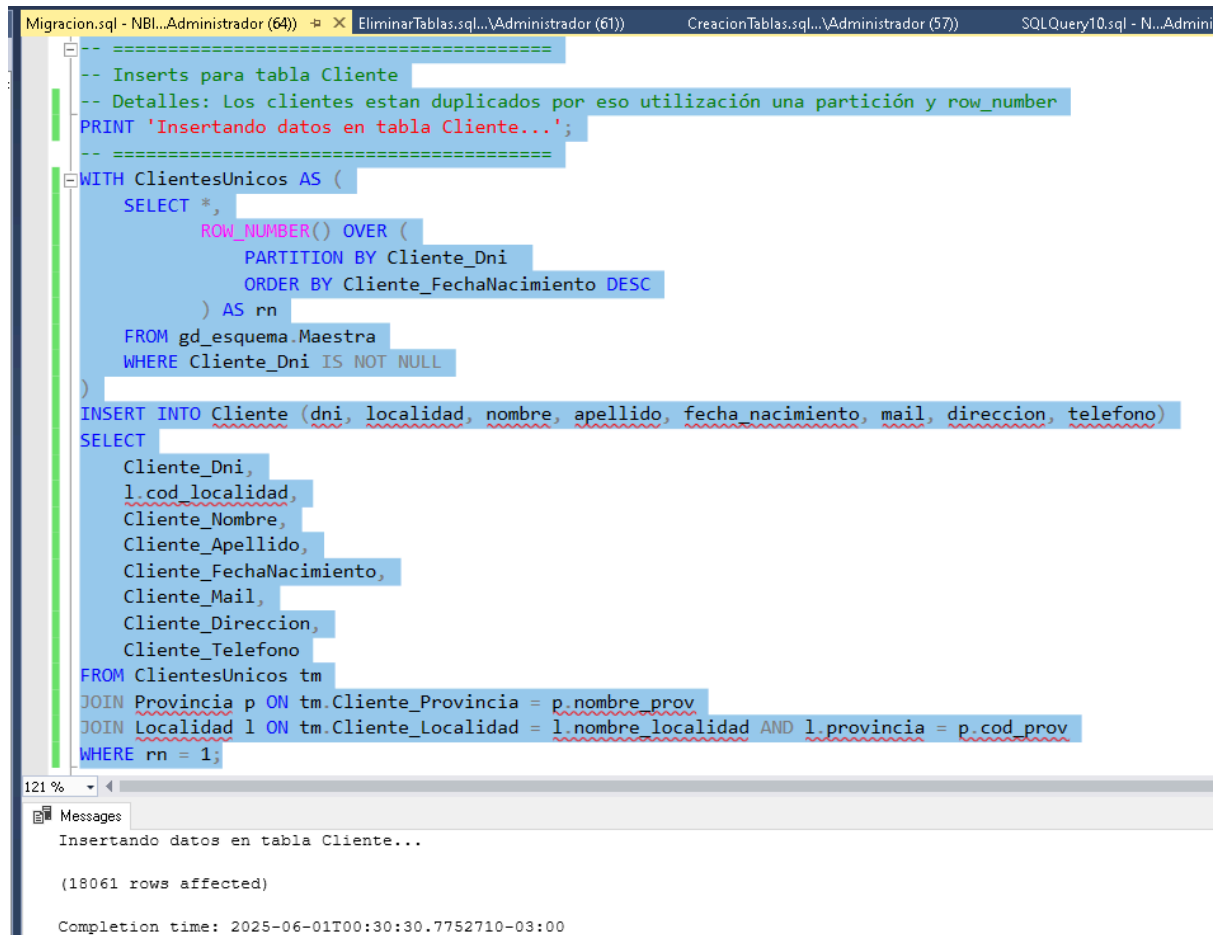
```
trac
SELECT
  Cliente_Dni,
  COUNT(DISTINCT CONCAT(
    Cliente_Nombre, '|',
    Cliente_Apellido, '|',
    Cliente_FechaNacimiento, '|',
    Cliente_Mail, '|',
    Cliente_Direccion, '|',
    Cliente_Telefono, '|',
    Cliente_Provincia, '|',
    Cliente_Localidad
  )) AS versiones_distintas
FROM gd_esquema.Maestra
WHERE Cliente_Dni IS NOT NULL
GROUP BY Cliente_Dni
HAVING COUNT(DISTINCT CONCAT(
  Cliente_Nombre, '|',
  Cliente_Apellido, '|',
  Cliente_FechaNacimiento, '|',
  Cliente_Mail, '|',
  Cliente_Direccion, '|',
  Cliente_Telefono, '|',
  Cliente_Provincia, '|',
  Cliente_Localidad
)) > 1;
```

121 %

Results Messages

Cliente_Dni	versiones_distintas
-------------	---------------------

Pero pensándolo bien, es más seguro utilizar una partición con Row\_number, ya que verificar la integridad de los datos manualmente es muy propenso a errores. Y por otro lado, si por alguna razón me agregan datos de clientes en la tabla maestra, debería volver a hacer la verificación manual. En cambio con row\_number, me aseguro de no tener clientes duplicados.



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a messages pane. The query editor contains a SQL script for inserting data into the 'Cliente' table. The script uses a Common Table Expression (CTE) named 'ClientesUnicos' to select unique records based on 'Cliente\_Dni' using the 'ROW\_NUMBER()' function. The 'INSERT INTO' statement then inserts data from 'ClientesUnicos' into the 'Cliente' table, joining with 'Provincia' and 'Localidad' tables. The messages pane shows the execution results, including the message 'Insertando datos en tabla Cliente...', the number of rows affected (18061), and the completion time.

```
-- =====
-- Inserts para tabla Cliente
-- Detalles: Los clientes estan duplicados por eso utilización una partición y row_number
PRINT 'Insertando datos en tabla Cliente...';
-- =====

WITH ClientesUnicos AS (
    SELECT *,
           ROW_NUMBER() OVER (
               PARTITION BY Cliente_Dni
               ORDER BY Cliente_FechaNacimiento DESC
           ) AS rn
    FROM gd_esquema.Maestra
    WHERE Cliente_Dni IS NOT NULL
)
INSERT INTO Cliente (dni, localidad, nombre, apellido, fecha nacimiento, mail, direccion, telefono)
SELECT
    Cliente_Dni,
    l.cod_localidad,
    Cliente_Nombre,
    Cliente_Apellido,
    Cliente_FechaNacimiento,
    Cliente_Mail,
    Cliente_Direccion,
    Cliente_Telefono
FROM ClientesUnicos tm
JOIN Provincia p ON tm.Cliente_Provincia = p.nombre_prov
JOIN Localidad l ON tm.Cliente_Localidad = l.nombre_localidad AND l.provincia = p.cod_prov
WHERE rn = 1;
```

121 %

Messages

Insertando datos en tabla Cliente...

(18061 rows affected)

Completion time: 2025-06-01T00:30:30.7752710-03:00

## Caso Detalle\_Pedido y Sillon:

Nos llama la atención que tenemos la misma cantidad de Detalle\_pedido y Sillon luego de la migración:

Insertando datos en tabla Sillon...

(218674 rows affected)

Insertando datos en tabla Detalle\_Pedido...

(218674 rows affected)

Intentamos encontrar la razón pero no la encontramos. Creemos que se debe a un error de diseño de las relaciones.

## Caso Detalle\_Factura:

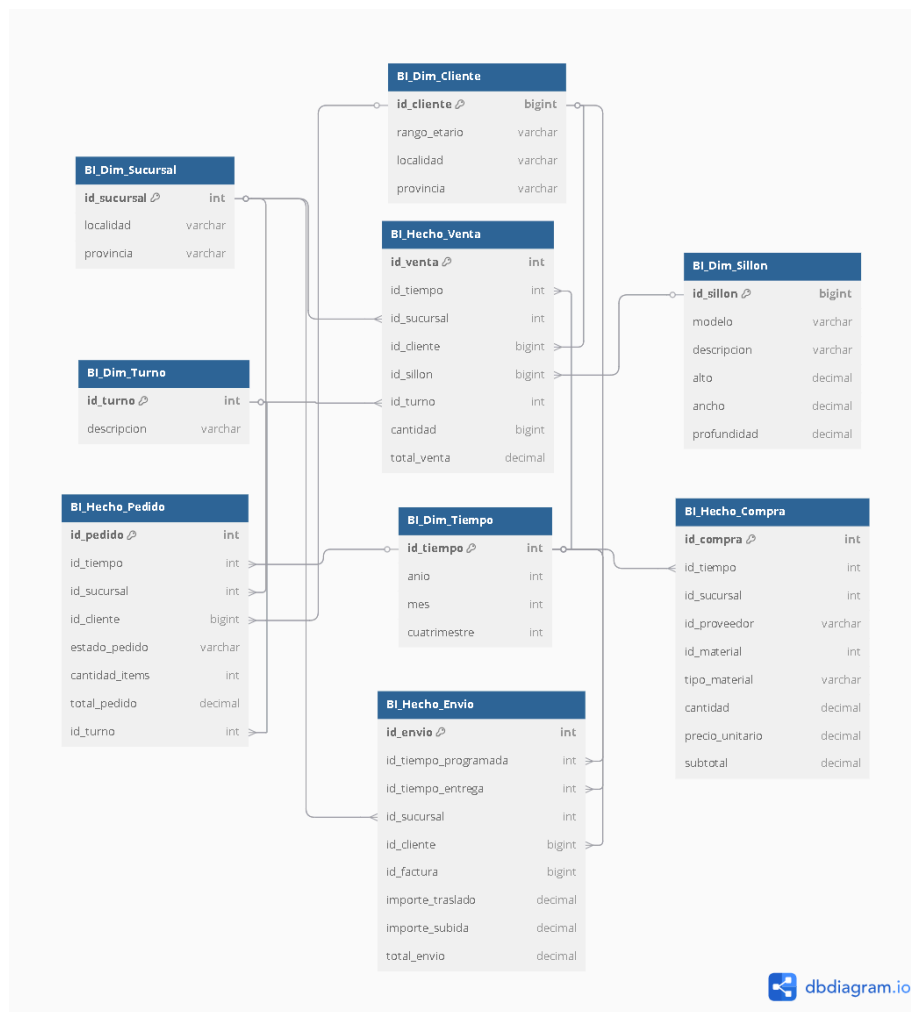
Vemos que no tenemos registros al realizar la migración:

Insertando datos en tabla Detalle\_Factura...  
(0 rows affected)

Creemos que es por lo mismo que Detalle\_pedido, tenemos una relación mal definida.

## Entrega 3:

Se diseñó el siguiente modelo BI:



## Tablas de Dimensiones

### BI\_Dim\_Tiempo

Dimensión temporal. Solicitado en el enunciado ya que se debe analizar la información por año/cuatrimstre/mes.

### BI\_Dim\_Sucursal

Dimensión demografica. Solicitado en el enunciado y las vistas requieren análisis por sucursal y provincia (ej: ganancias por sucursal, compras por sucursal y cuatrimestre, cumplimiento de envíos por sucursal).

### BI\_Dim\_Cliente

Dimensión de clientes. solicitado en el enunciado para conocer el rango etario y la localidad del cliente (ej: rendimiento de modelos según edad/localidad).

### BI\_Dim\_Turno

Otra dimensión temporal. El enunciado solicita análisis por turno de venta (ej: volumen de pedidos por turno).

### BI\_Dim\_Sillon

Dimensión de producto. El enunciado lo requiere y se requiere analizar los modelos de sillon.

### BI\_Dim\_TipoMaterial

El enunciado solicita un análisis de compras por tipo de material (ej: vista 8).

### BI\_Dim\_EstadoPedido

Solicitado por enunciado para poder calcular la conversión de pedidos según estado, esta dimensión permite categorizar y analizar el pipeline de pedidos (ej: pendiente, confirmado, entregado, etc.).



## Tablas de Hechos

### BI\_Hecho\_Venta

Registra las ventas realizadas a partir de las facturas. Es clave para calcular ingresos, ganancias y analizar qué modelos se venden más, por cliente, sucursal o período.

### BI\_Hecho\_Pedido

Guarda los pedidos realizados, estén o no facturados. Permite analizar el volumen de pedidos, los estados y los tiempos desde que se hace hasta que se entrega.

### BI\_Hecho\_Compra

Registra las compras de materiales por parte de cada sucursal. Sirve para calcular egresos, analizar costos y ver qué materiales se compran más.

### BI\_Hecho\_Envio

Almacena información sobre los envíos de pedidos. Se usa para ver si se entregaron a tiempo, cuánto costaron y cómo se comportan según la localidad del cliente.