

# BASES DE DATOS ANALÍTICAS

## Práctica 4

Enero 2025



Víctor Suesta Arribas

# ÍNDICE

## ÍNDICE

[Ejercicio 1. Análisis de datos mediante autoconsumo](#)

[Ejercicio 2. Análisis de datos multidimensional](#)

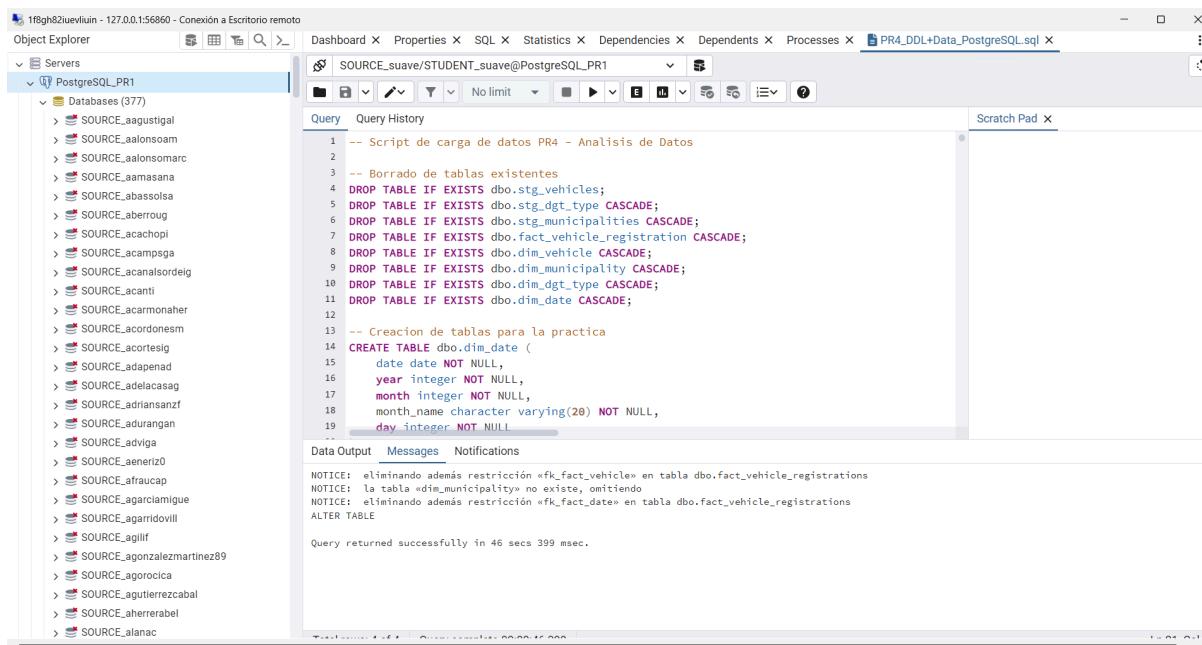
[Ejercicio 3. Análisis de datos mediante Dashboards](#)

# Ejercicio 1. Análisis de datos mediante autoconsumo

En este ejercicio mi objetivo ha sido verificar que el modelo de datos se ha desplegado correctamente en PostgreSQL y que las tablas del esquema `dbo` están disponibles y contienen información coherente para el análisis posterior. Para ello he seguido una secuencia de validación basada en: (1) ejecución del script de creación y carga, (2) verificación de la estructura de una dimensión clave, (3) listado de tablas en `dbo`, (4) volumetría estimada por tabla, (5) revisión de la estructura de la tabla de hechos y (6) análisis de estadísticas con `pg_stats` para evaluar completitud y cardinalidad en `dim_vehicle`.

## 1.0 D0 — Ejecución del script de creación y carga (PR4\_DDL+Data\_PostgreSQL.sql)

En primer lugar ejecuté el script de la práctica para asegurar la creación (o recreación) de las tablas y la carga de datos en el esquema `dbo`. Esta ejecución es imprescindible para garantizar que el modelo de PR4 esté disponible con la estructura esperada (incluyendo atributos necesarios para el ejercicio).



```
-- Script de carga de datos PR4 - Analisis de Datos
-- Borrado de tablas existentes
DROP TABLE IF EXISTS dbo.stg_vehicles;
DROP TABLE IF EXISTS dbo.stg_dgt_type CASCADE;
DROP TABLE IF EXISTS dbo.stg_municipalities CASCADE;
DROP TABLE IF EXISTS dbo.fact_vehicle_registration CASCADE;
DROP TABLE IF EXISTS dbo.dim_vehicle CASCADE;
DROP TABLE IF EXISTS dbo.dim_municipality CASCADE;
DROP TABLE IF EXISTS dbo.dim_dgt_type CASCADE;
DROP TABLE IF EXISTS dbo.dim_date CASCADE;

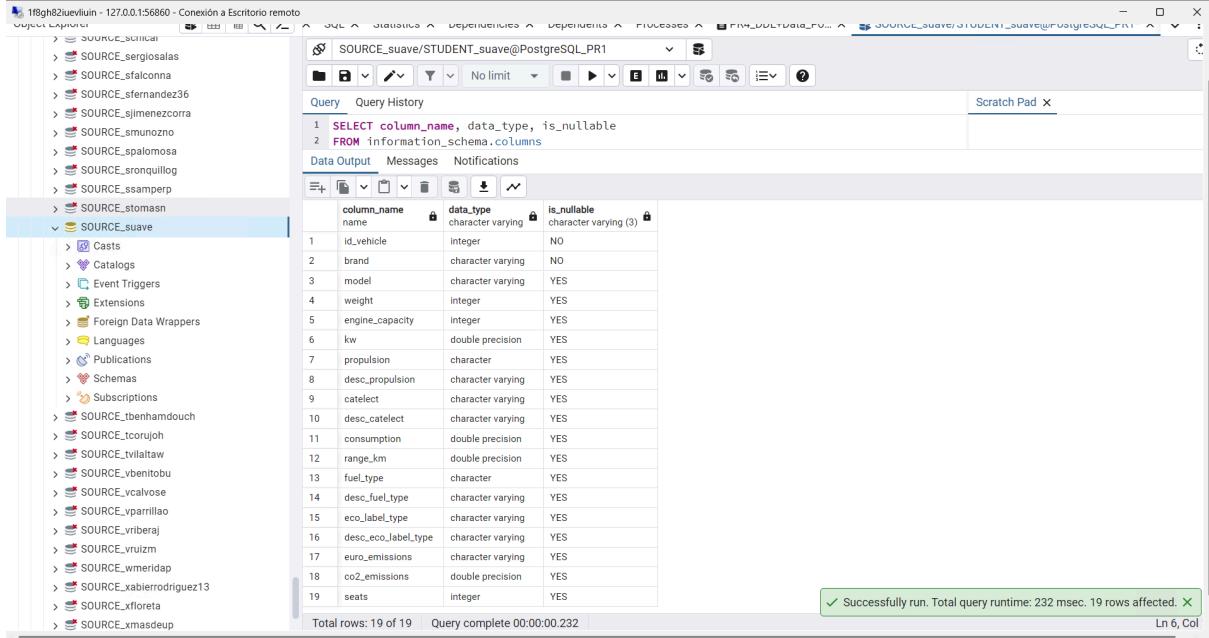
-- Creacion de tablas para la practica
CREATE TABLE dbo.dim_date (
    date date NOT NULL,
    year integer NOT NULL,
    month integer NOT NULL,
    month_name character varying(20) NOT NULL,
    day integer NOT NULL
);

ALTER TABLE dbo.dim_date
    ADD CONSTRAINT PK_dim_date PRIMARY KEY (date);
```

**Evidencia (Figura D0):** ejecución correcta del script `PR4_DDL+Data_PostgreSQL.sql` en PostgreSQL.

### 1.0.1 D0.1 — Verificación de estructura en `dbo.dim_vehicle`

Tras la ejecución del script, verifiqué la estructura de `dbo.dim_vehicle` consultando el diccionario de datos mediante `information_schema.columns`. Este paso confirma que la dimensión contiene los atributos esperados por el enunciado, entre ellos `id_vehicle` y `co2_emissions`, necesarios para la consulta posterior sobre `pg_stats`.



```

18gh82iuevluin - 127.0.0.1:56860 - Conexión a Escritorio remoto
Query History
Scratch Pad X

Query
SELECT column_name, data_type, is_nullable
FROM information_schema.columns

Data Output
Messages
Notifications

column_name          data_type      is_nullable
name
1 id_vehicle         integer        NO
2 brand              character varying NO
3 model              character varying YES
4 weight              integer        YES
5 engine_capacity    integer        YES
6 kw                 double precision YES
7 propulsion         character      YES
8 desc_propulsion   character varying YES
9 catelect           character varying YES
10 desc_catelect     character varying YES
11 consumption        double precision YES
12 range_km           double precision YES
13 fuel_type          character      YES
14 desc_fuel_type    character varying YES
15 eco_label_type    character varying YES
16 desc_eco_label_type character varying YES
17 euro_emissions    character varying YES
18 co2_emissions     double precision YES
19 seats              integer        YES

Total rows: 19 of 19  Query complete 00:00:00.232

```

✓ Successfully run. Total query runtime: 232 msec. 19 rows affected. X  
Ln 6, Col

**Evidencia (Figura D0.1):** listado de columnas de `dbo.dim_vehicle` donde se observa la presencia de `id_vehicle` y `co2_emissions`.

## 1.1 D1 — Listado de tablas en el esquema `dbo` (`pg_tables`)

A continuación comprobé que el esquema `dbo` contiene las tablas del modelo, consultando el catálogo del sistema `pg_tables` y filtrando por el esquema correspondiente.

```

1 SELECT
2   schemaname,
3   tablename,
4   tableowner,
5   hasindexes,
6   hasrules,
7   hastriggers
8 FROM pg_tables
9 WHERE schemaname = 'dbo'
10 ORDER BY tablename;
11

```

schemaname	tablename	tableowner	hasindexes	hasrules	hastriggers
dbo	dgt_type	STUDENT_suave	false	false	false
dbo	dim_date	STUDENT_suave	true	false	true
dbo	dim_dgt_type	STUDENT_suave	true	false	true
dbo	dim_municipalities	STUDENT_suave	true	false	true
dbo	dim_vehicle	STUDENT_suave	true	false	true
dbo	fact_vehicle_registration	STUDENT_suave	true	false	true
dbo	fact_vehicle_registrations	STUDENT_suave	true	false	true
dbo	municipalities	STUDENT_suave	false	false	false
dbo	stg_dgt_type	STUDENT_suave	false	false	false
dbo	stg_municipalities	STUDENT_suave	false	false	false
dbo	stg_vehicles	STUDENT_suave	false	false	false
dbo	vehicles	STUDENT_suave	false	false	false

Total rows: 12 of 12    Query complete 00:00:00.157    Successfully run. Total query runtime: 157 msec. 12 rows affected.    Ln 11, Col

**Evidencia (Figura D1):** listado de tablas existentes en `dbo` obtenido desde `pg_tables`.

## 1.2 D2 — Volumetría estimada por tabla (pg\_stat\_user\_tables)

Para evaluar de forma inicial la volumetría de cada tabla, realicé un cruce entre el catálogo `pg_tables` y las estadísticas `pg_stat_user_tables`, utilizando `n_live_tup` como estimación del número de tuplas vivas (filas) por tabla. Esta comprobación permite identificar rápidamente tablas con mayor peso y detectar posibles tablas vacías tras la carga.

```

1 SELECT
2   t.schemaname,
3   t.tablename,
4   COALESCE(s.n_live_tup, 0) AS total_registros_estimado
5 FROM pg_tables t
6 LEFT JOIN pg_stat_user_tables s
7   ON s.schemaname = t.schemaname
8   AND s.relname = t.tablename
9 WHERE t.schemaname = 'dbo'
10 ORDER BY total_registros_estimado ASC, t.tablename ASC;
11

```

schemaname	tablename	total_registros_estimado
dbo	dgt_type	0
dbo	fact_vehicle_registrations	0
dbo	municipalities	0
dbo	vehicles	0
dbo	dim_dgt_type	111
dbo	stg_dgt_type	112
dbo	dim_vehicle	1257
dbo	dim_date	2181
dbo	dim_municipalities	8132
dbo	stg_municipalities	8132
dbo	fact_vehicle_registration	1721704
dbo	stg_vehicles	1721704

Total rows: 12 of 12    Query complete 00:00:00.154    Successfully run. Total query runtime: 154 msec. 12 rows affected.    Ln 11, Col

**Evidencia (Figura D2):** listado de tablas en `dbo` con la columna `total_registros_estimado` ordenada ascendenteamente.

### 1.3 D3 — Estructura de `dbo.fact_vehicle_registration` (`information_schema.columns`)

Dado que `fact_vehicle_registration` es la tabla de hechos principal, revisé su definición para documentar las columnas disponibles, sus tipos de datos y si admiten valores nulos. Esta verificación es necesaria para evitar supuestos incorrectos en análisis posteriores y para confirmar que la tabla está alineada con el modelo desplegado.

The screenshot shows a PostgreSQL client interface with the following details:

- Left Panel (Object Explorer):** Shows a tree view of database objects under the schema `SOURCE_suave`, which is selected.
- Top Bar:** Shows the connection information "1tnv6goj3zsty8 - 127.0.0.1:57998 - Conexión a Escritorio remoto".
- Toolbar:** Includes icons for Object Explorer, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a scratch pad.
- Query History:** Displays the executed SQL query.
- Data Output:** Shows the results of the query in a table format.
- Messages:** Shows a message indicating the query was successfully run.
- Notifications:** Shows a message indicating 7 rows affected.
- Bottom Status:** Shows "Total rows: 7 of 7" and "Query complete 00:00:00.299".

```
1 SELECT
2   ordinal_position,
3   column_name,
4   data_type,
5   is_nullable
6 FROM information_schema.columns
7 WHERE table_schema = 'dbo'
8   AND table_name  = 'fact_vehicle_registration'
9 ORDER BY ordinal_position;
10
```

ordinal_position	column_name	data_type	is_nullable
1	id_registration	integer	NO
2	registration_date	date	NO
3	vehicle_id	integer	NO
4	id_dgt_type	integer	NO
5	municipalities_id	bigint	NO
6	new_used	boolean	YES
7	quantity	integer	NO

**Evidencia (Figura D3):** resultado de `information_schema.columns` para `dbo.fact_vehicle_registration` con `column_name`, `data_type` e `is_nullable`.

### 1.4 D4 — Estadísticas con `pg_stats` sobre `dbo.dim_vehicle`

Finalmente, ejecuté la consulta del enunciado sobre `pg_stats` para analizar, en atributos seleccionados de `dbo.dim_vehicle`, la fracción de nulos, la cardinalidad estimada y los valores más frecuentes. Este análisis sirve como perfilado estadístico inicial de la dimensión.

```

1 SELECT attname, null_frac, n_distinct, most_common_vals
2 FROM pg_stats
3 WHERE tablename = 'dim_vehicle'
4 AND schemaname = 'dbo'
5 AND attname IN ('id_vehicle', 'brand', 'model', 'co2_emissions');
6

```

attname	null_frac	n_distinct	most_common_vals
co2_emissions	0.4892909	343	{0,119,110,104,99,115,139,135,107,105,98,109,113,120,95,100,114,112,108,159,145,117,132,129,140,94,139,123,103,106,131,118,149,116,90,96}
id_vehicle	0	-1	[null]
brand	0	607	{--PEUGEOT,CITROEN,MERCEDES-BENZ,BMW,HONDA,TOYOTA,PIAGGIO,YAMAHA,VOLKSWAGEN,OPEL,AUDI,RENAULT,SEAT,SUZUKI,NISSAN,FORD,I
model	0.0004266756	-0.9699633	{--AD260SY/PS,AD260SY/PS CNG,ND,70C18,CRAFTER,100E,180E,CANTER,SPRINTER,TGS,3E20,40C14N,CARRETILLA ELEVADORA,CICLOMOT

Total rows: 4 of 4    Query complete 00:00:00.149    Successfully run. Total query runtime: 149 msec. 4 rows affected.

**Evidencia (Figura D4):** salida de `pg_stats` para `id_vehicle`, `brand`, `model` y `co2_emissions`.

#### 1.4.1 Interpretación de las columnas de `pg_stats`

- **null\_frac:** fracción estimada de valores nulos en la columna (aproximación entre 0 y 1).
- **n\_distinct:** estimación del número de valores distintos.
  - Si **n\_distinct > 0**, es una estimación directa del número de distintos.
  - Si **n\_distinct < 0**, el valor negativo representa aproximadamente la fracción de distintos respecto al total de filas.
  - Si **n\_distinct = -1**, suele indicar una columna prácticamente única (distintos ≈ número de filas).
- **most\_common\_vals:** lista de valores más frecuentes estimados para esa columna.

#### 1.4.2 Interpretación por atributo según el resultado obtenido

- **id\_vehicle:** al no presentar nulos y mostrar comportamiento de alta unicidad, es consistente con un identificador de la dimensión.
- **brand:** presenta completitud alta y una cardinalidad significativa; los valores más frecuentes permiten identificar las marcas predominantes.
- **model:** muestra cardinalidad muy alta (comportamiento cercano a único en gran parte de los registros), lo cual es esperable al tratarse de un descriptor más granular.
- **co2\_emissions:** presenta una proporción elevada de valores ausentes (según `null_frac`), lo que indica que el dato de emisiones no está disponible para una parte relevante del conjunto. La cardinalidad y los valores más frecuentes permiten caracterizar los niveles de emisiones presentes en los registros informados.

## 1.5 Conclusión

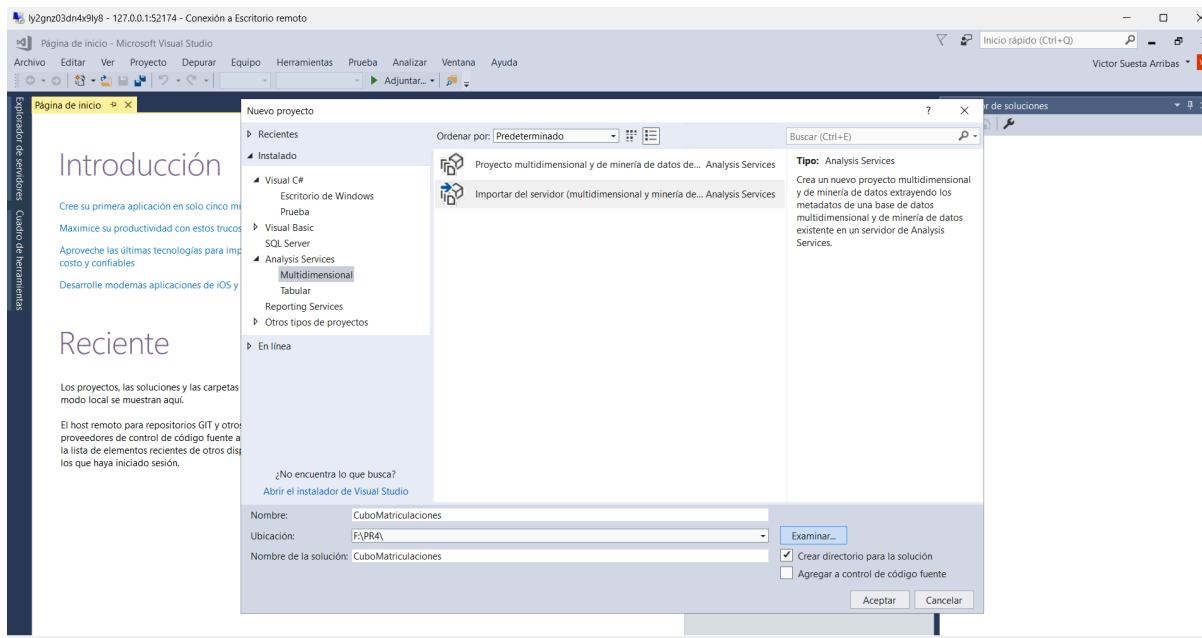
Con las evidencias D0, D0.1, D1, D2, D3 y D4 he validado que el modelo de PR4 se ha desplegado correctamente en PostgreSQL: el esquema `dbo` contiene las tablas necesarias, la dimensión `dim_vehicle` incluye los atributos esperados por el enunciado, la tabla de hechos `fact_vehicle_registration` presenta una estructura definida y coherente, y las estadísticas de `pg_stats` permiten documentar completitud y cardinalidad iniciales para variables clave de `dim_vehicle`. Esta verificación deja el entorno preparado y trazable para los ejercicios posteriores.

# Ejercicio 2. Análisis de datos multidimensional

En este ejercicio mi objetivo ha sido diseñar y desplegar un cubo multidimensional (OLAP) en SQL Server Analysis Services a partir del modelo en estrella basado en **FACT\_vehicle\_registration** y sus dimensiones asociadas, con el fin de realizar análisis multidimensionales mediante el explorador de cubos de Visual Studio. Para asegurar un resultado correcto y trazable, he documentado el proceso completo siguiendo el orden del enunciado: (1) creación del proyecto y configuración de destino/origen, (2) creación de la vista de origen de datos (DSV) del modelo estrella, (3) creación del cubo, (4) configuración de dimensiones y jerarquías (DIM\_DATE y DIM\_DGT\_TYPE), (5) procesamiento y despliegue del cubo, (6) configuración adicional de DIM\_Vehicle y DIM\_Municipality, y (7) explotación del cubo con consultas en el explorador.

## 2.1 F1 — Creación del proyecto (Visual Studio / Analysis Services)

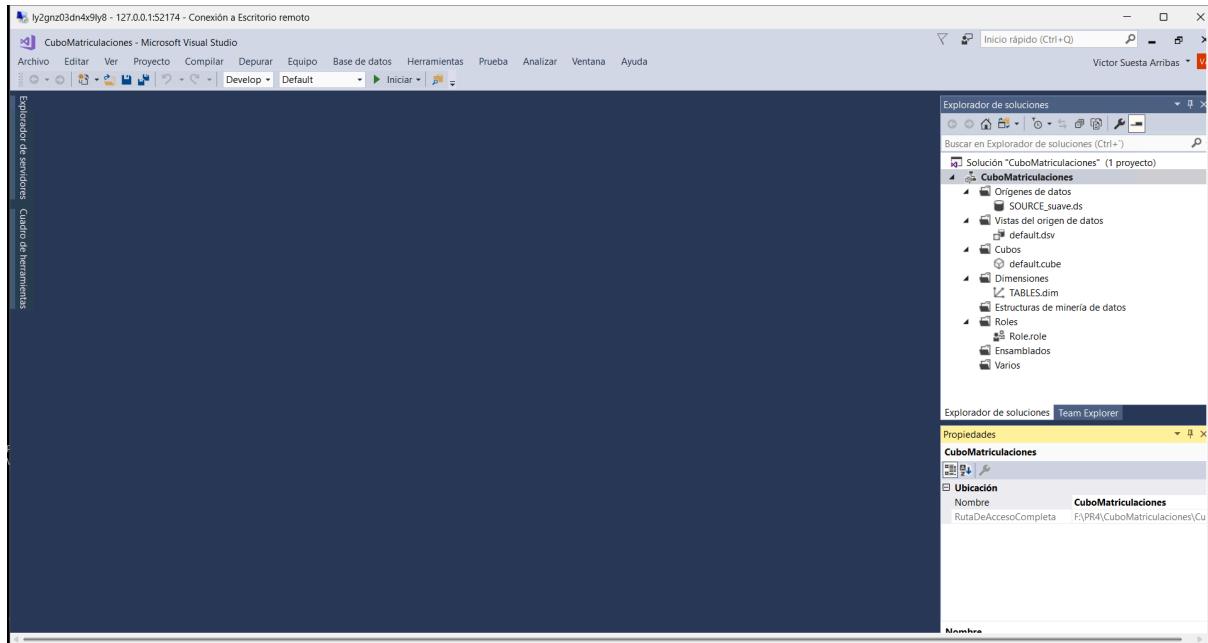
En primer lugar creé un proyecto nuevo en Visual Studio utilizando la plantilla **Analysis Services** → **Multidimensional** → **Proyecto multidimensional y de minería de datos**, asignando un nombre representativo y una ubicación dentro de la unidad de trabajo. Este paso es esencial porque garantiza que el proyecto se genera con el tipo correcto y queda preparado para importar o construir los componentes de Analysis Services.



**Evidencia (Figura F1):** creación del proyecto con la plantilla de Analysis Services y definición de nombre/ubicación.

### 2.1.1 F2 — Verificación del proyecto y artefactos en el Explorador de soluciones

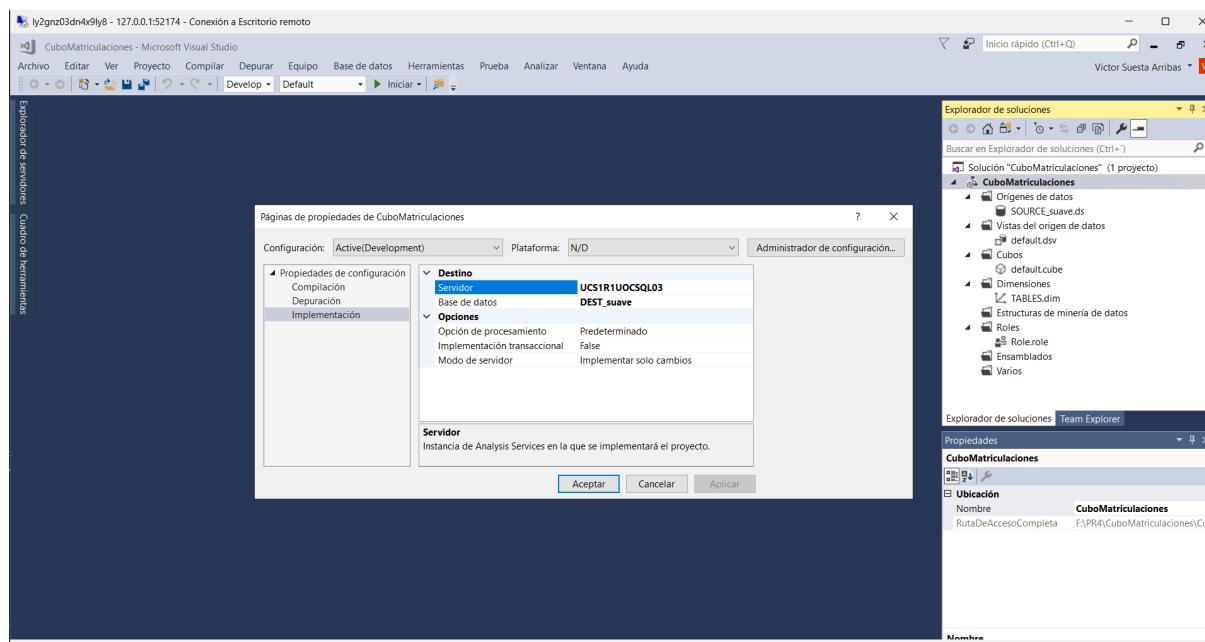
Tras crear/importar el proyecto comprobé en el **Explorador de soluciones** que aparecen los componentes esperados: origen de datos (.ds), vista de origen de datos (.dsv), cubos (.cube) y dimensiones (.dim). Esta verificación inicial confirma que la estructura del proyecto es coherente y que se puede continuar con la configuración sin perder el cubo **default** importado (que no debe eliminarse).



**Evidencia (Figura F2):** Explorador de soluciones mostrando el proyecto y sus artefactos principales.

### 2.1.2 F3 — Configuración del destino de implementación (Deployment)

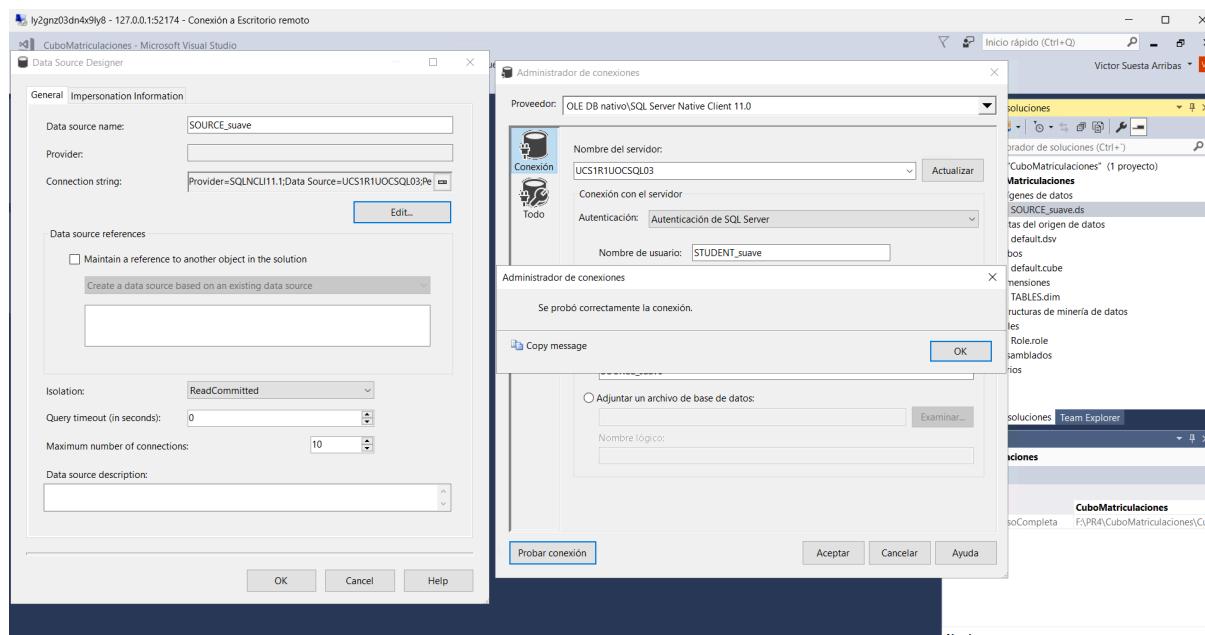
A continuación abrí las propiedades del proyecto y, en el apartado **Implementación**, configuré el servidor de Analysis Services proporcionado por el laboratorio y verifiqué que la base de datos de destino es la correspondiente (DEST\_loginuoc). Este ajuste es imprescindible para que el despliegue y el procesamiento se realicen contra el destino correcto.



**Evidencia (Figura F3):** propiedades del proyecto en Implementación (servidor y base de datos).

### 2.1.3 F4.a — Configuración del origen de datos y prueba de conexión

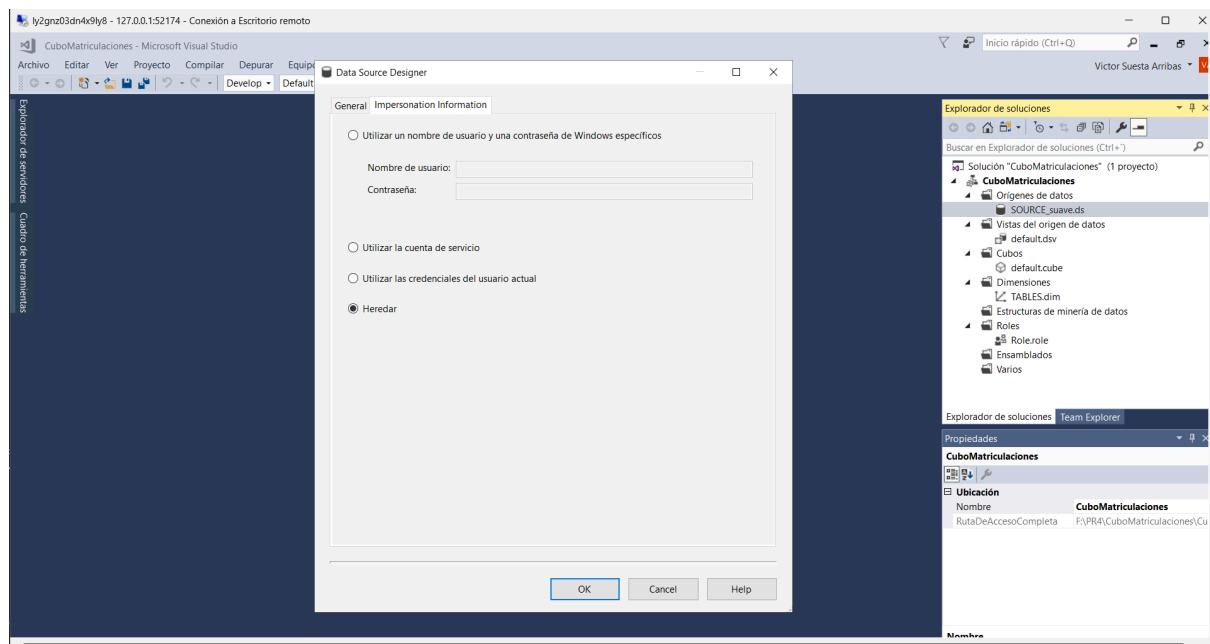
Después configuré el **Origen de datos** del proyecto, editando la conexión para asegurar que el servidor es correcto e introduciendo las credenciales requeridas. Finalmente ejecuté **Probar conexión** para confirmar que la conectividad es correcta antes de continuar con la creación de la DSV y del cubo.



**Evidencia (Figura F4.a):** edición del origen de datos y prueba de conexión satisfactoria.

## 2.1.4 F4.b — Impersonation Information (Heredar)

Para garantizar que el motor de Analysis Services puede acceder a los datos durante el procesamiento, revisé la pestaña **Impersonation Information** y dejé activada la opción **Heredar (Inherit)**, tal como solicita el enunciado. Esta configuración evita errores típicos de permisos durante el procesamiento de dimensiones y cubos.

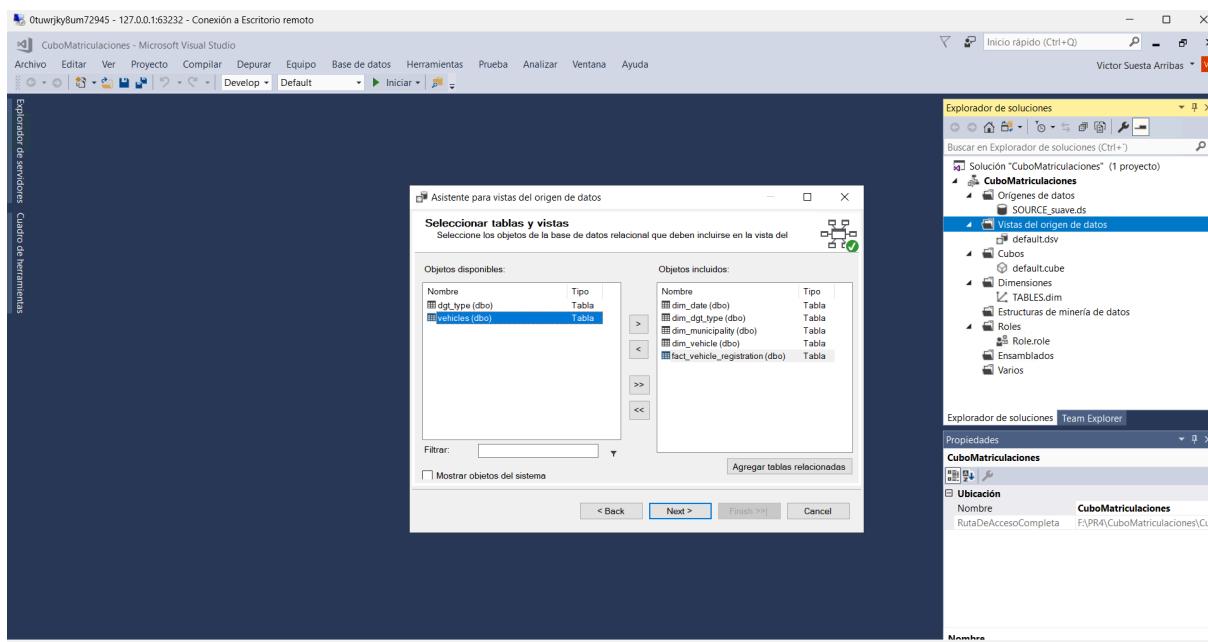


**Evidencia (Figura F4.b):** configuración de Impersonation Information = Heredar.

## 2.2 Crear una vista de origen de datos (DSV) y el cubo correspondiente

### 2.2.1 F5 — Asistente de DSV: selección de tablas y vistas (incidencia de tablas no disponibles)

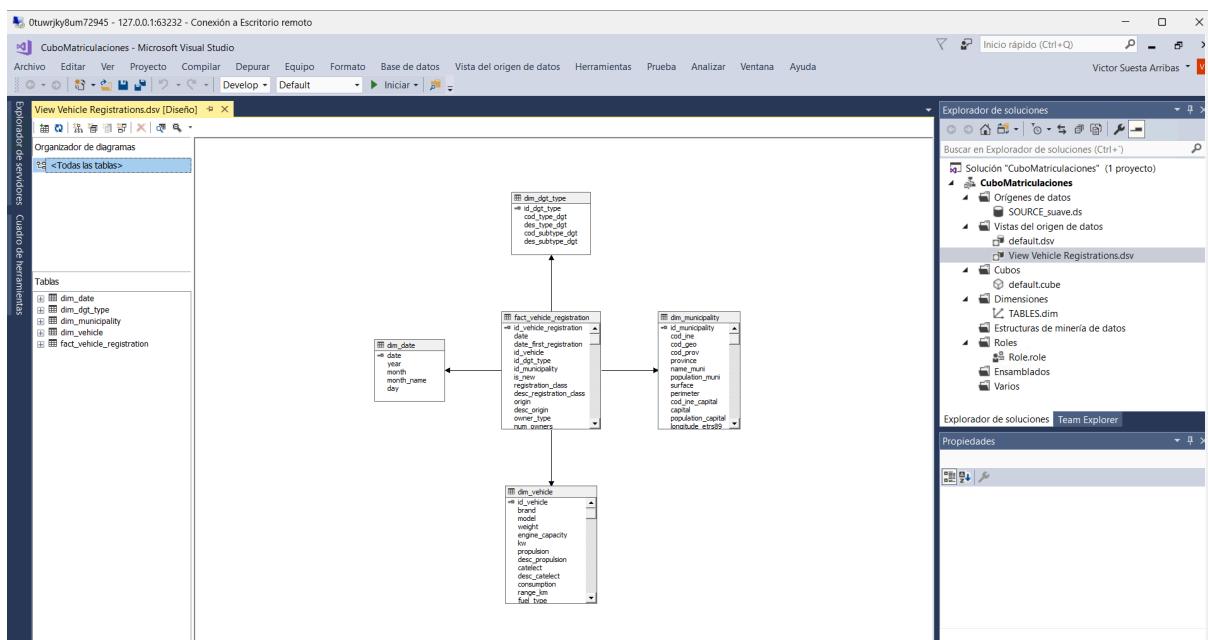
Inicié el asistente para crear una nueva **Vista del origen de datos** destinada a incorporar el modelo en estrella. En el paso **Seleccionar tablas y vistas**, el asistente quedó detenido debido a que no se encontraban disponibles las tablas esperadas en ese momento. Documenté esta incidencia porque forma parte de la trazabilidad del proceso seguido en el entorno.



**Evidencia (Figura F5):** asistente de creación de DSV en “Seleccionar tablas y vistas” con la incidencia detectada.

## 2.2.2 F6 — DSV creada: diagrama del modelo en estrella (View\_vehicle\_registrations)

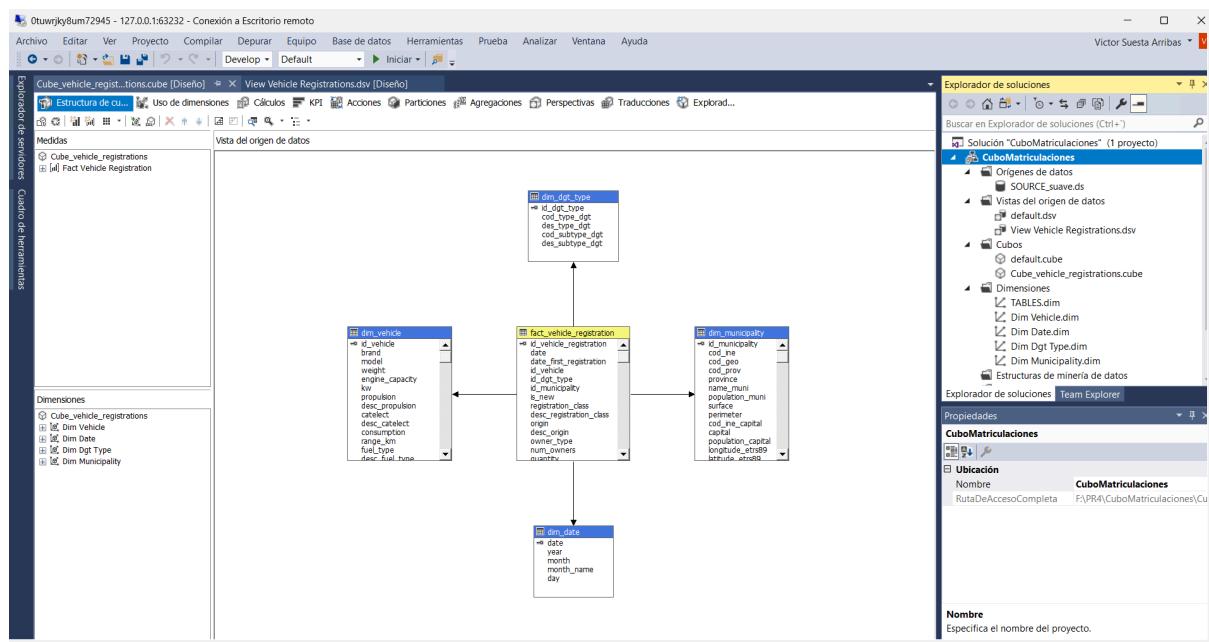
Una vez disponible el conjunto de tablas, completé la creación de la DSV y verifiqué el diagrama resultante. En el diagrama se observa claramente el esquema en estrella con **fact\_vehicle\_registration** en el centro y las relaciones hacia **dim\_date**, **dim\_dgt\_type**, **dim\_municipality** y **dim\_vehicle**. Esta comprobación es crítica para asegurar que las claves y uniones están correctamente definidas antes de crear el cubo.



**Evidencia (Figura F6):** diagrama de la DSV View\_vehicle\_registrations con el modelo estrella y sus relaciones.

## 2.2.3 F7 — Creación del cubo Cube\_vehicle\_registrations (tablas existentes + medida Quantity)

A continuación creé el cubo mediante el asistente **Nuevo cubo** → **Usar tablas existentes**. Seleccioné la DSV **View\_vehicle\_registrations**, definí **fact\_vehicle\_registration** como tabla del grupo de medidas y marqué únicamente la medida **Quantity**. Después seleccioné como dimensiones todas las tablas **DIM\_** (dim\_date, dim\_dgt\_type, dim\_municipality y dim\_vehicle), excluyendo la tabla de hechos, y asigné el nombre **Cube\_vehicle\_registrations**. Tras finalizar, comprobé que el diseñador del cubo muestra la medida y las dimensiones vinculadas.



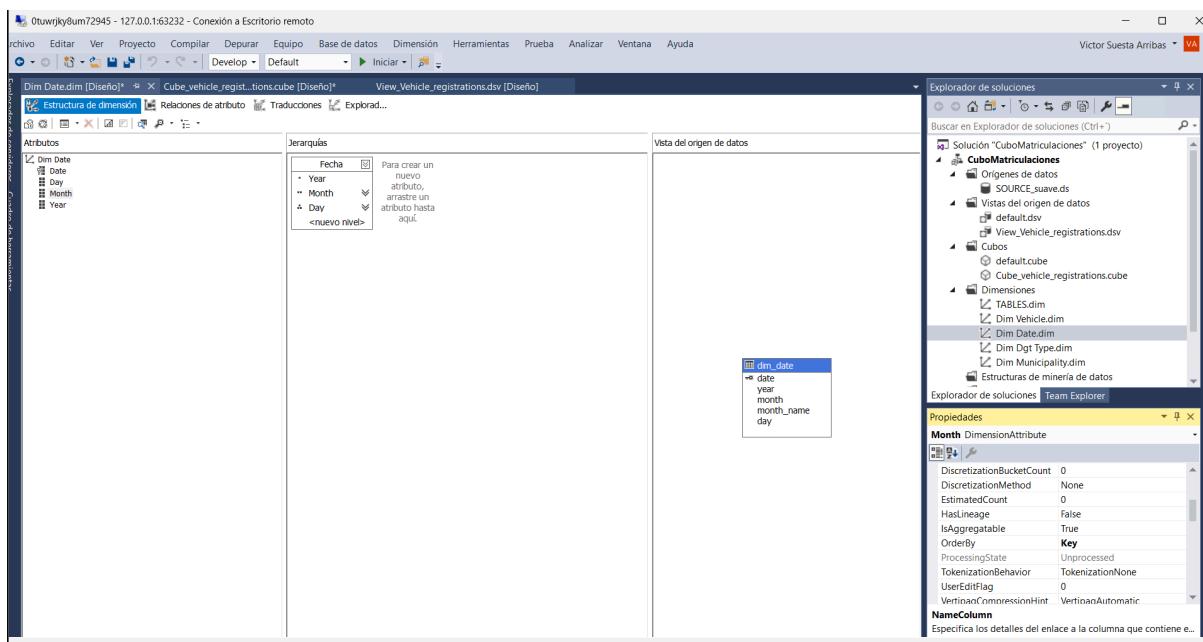
**Evidencia (Figura F7):** diagrama del cubo con medida Quantity y dimensiones asociadas.

## 2.3 Jerarquías, dimensiones y atributos

### 2.3.1 Configuración de la dimensión DIM\_DATE

#### 2.3.1.1 F10 — Estructura de DIM\_DATE y jerarquía “Fecha” (Year > Month > Day)

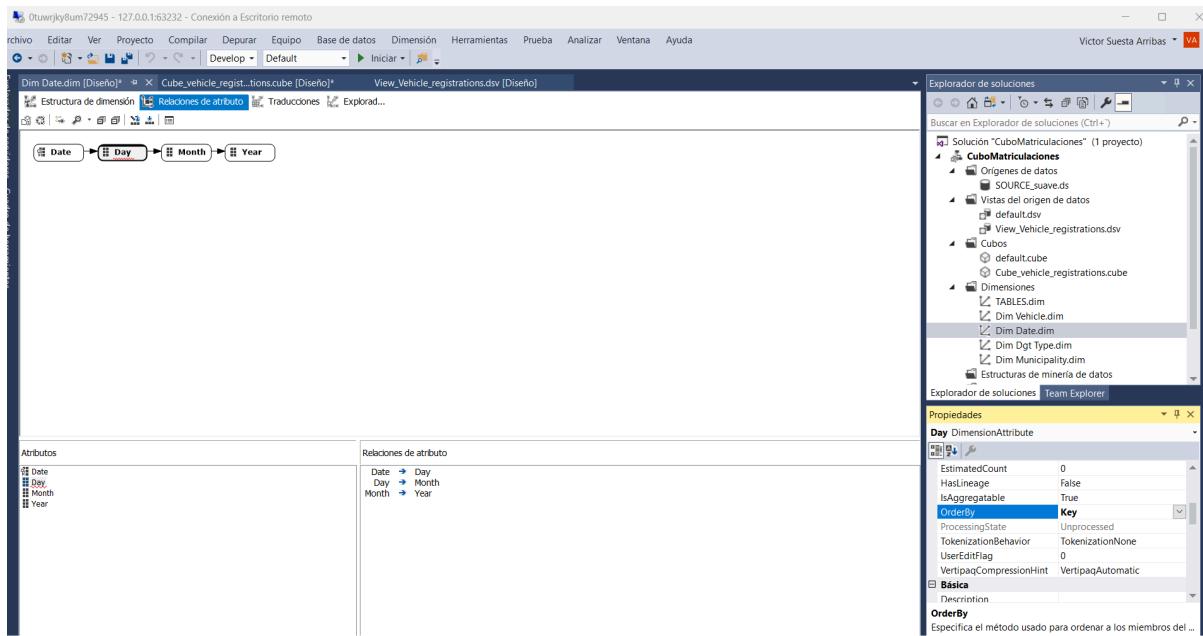
Abrí la dimensión **DIM\_DATE** y configuré los atributos necesarios para el análisis temporal. Construí una jerarquía de tres niveles **Year > Month > Day** y la renombré como **Fecha**, de modo que la navegación temporal del cubo se realice mediante esta jerarquía.



**Evidencia (Figura F10): jerarquía “Fecha” creada en DIM\_Date con niveles Year, Month y Day.**

### 2.3.1.2 F11 — Relaciones de atributo en DIM\_DATE (Date → Day → Month → Year)

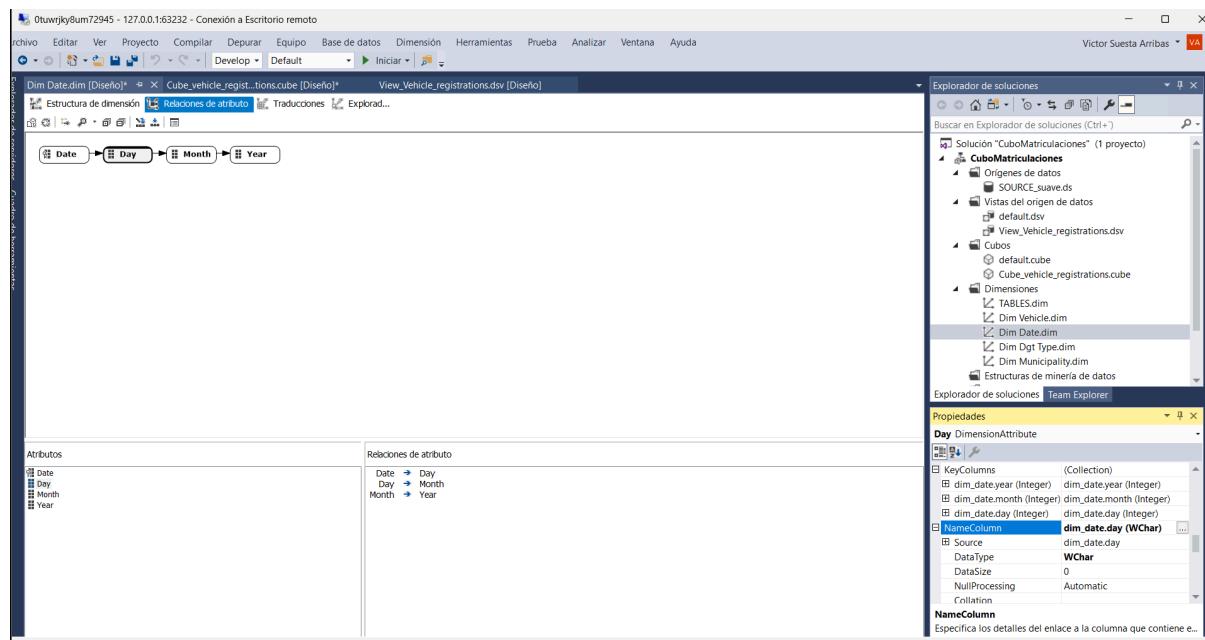
En la pestaña **Relaciones de atributo** verifiqué que la relación está definida desde el nivel más detallado al más general, quedando **Date → Day → Month → Year**. Esta estructura es importante para optimizar la agregación y la navegación jerárquica.



**Evidencia (Figura F11): relaciones de atributo definidas en DIM\_Date.**

### 2.3.1.3 F12.1.1 — Propiedades del atributo Day (KeyColumns / NameColumn / OrderBy)

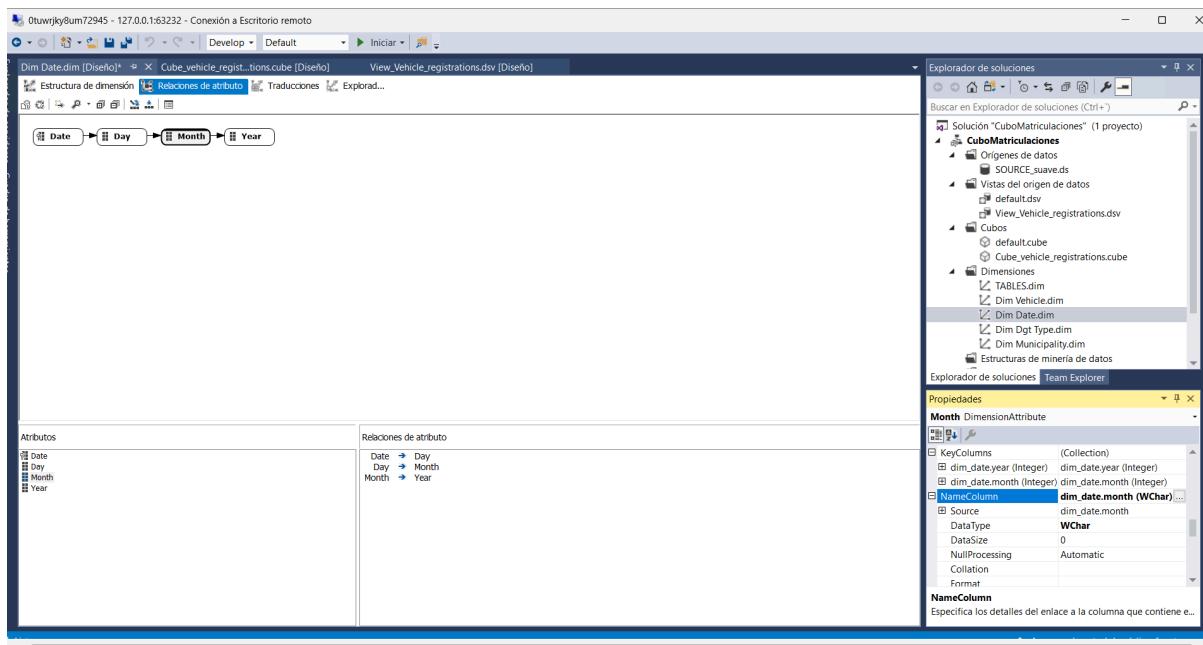
Para el atributo **Day** ajusté la configuración de claves para evitar ambigüedades (por ejemplo, “día 1” repetido en distintos meses). Configuré **KeyColumns** incluyendo **Year**, **Month** y **Day** (en ese orden), establecí **NameColumn = Day** y fijé **OrderBy = Key** para garantizar orden cronológico correcto.



**Evidencia (Figura F12.1.1):** propiedades de Day (KeyColumns, NameColumn y OrderBy).

### 2.3.1.4 F12.2.1 — Propiedades del atributo Month (KeyColumns / NameColumn / OrderBy)

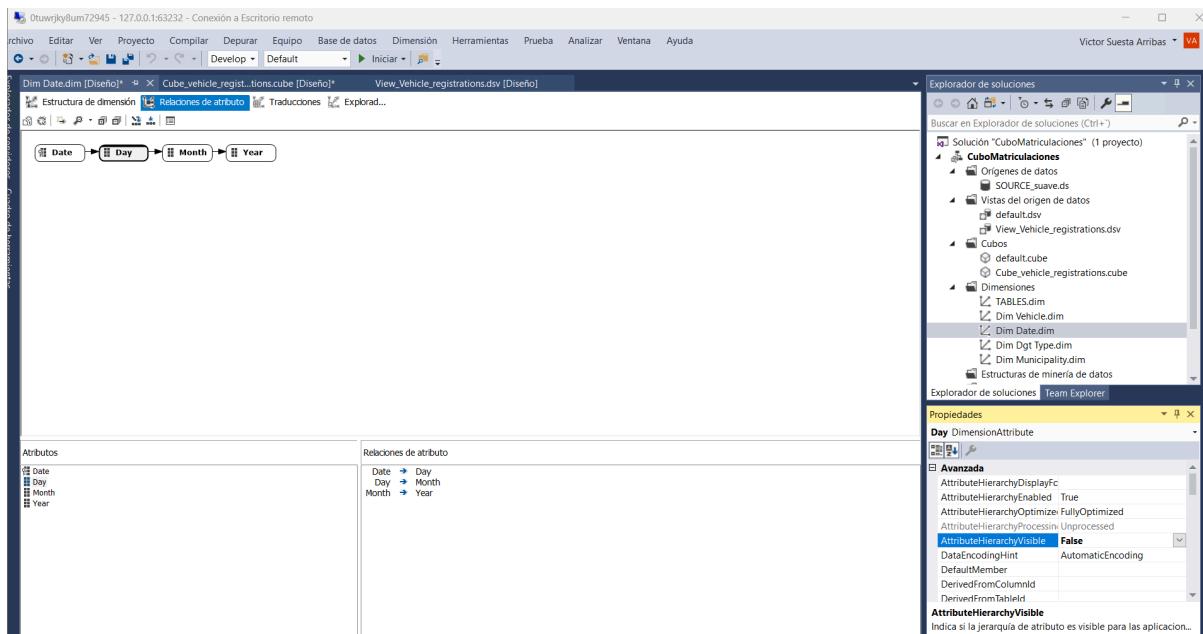
En el atributo **Month** configuré **KeyColumns** con **Year y Month** (en ese orden), establecí **NameColumn = Month** y fijé **OrderBy = Key**, asegurando que los meses se ordenen correctamente dentro de cada año.



**Evidencia (Figura F12.2.1):** propiedades de Month (KeyColumns, NameColumn y OrderBy).

### 2.3.1.5 F12.1.2 — AttributeHierarchyVisible en Day/Month/Year (ocultación de jerarquías individuales)

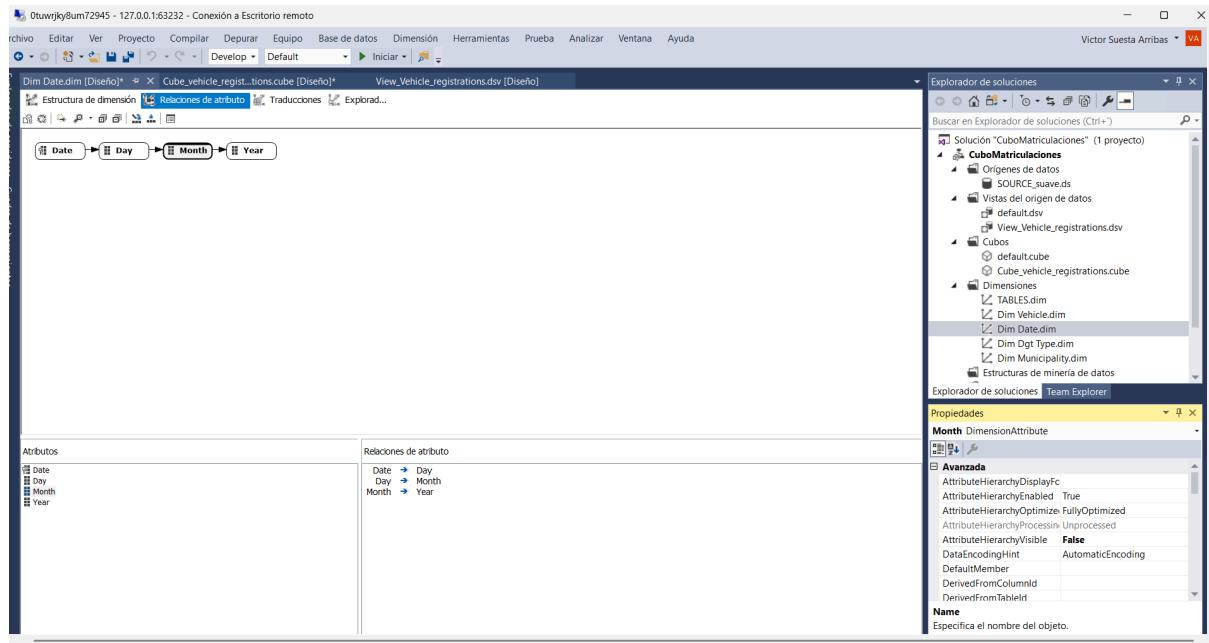
Para que el usuario trabaje con la jerarquía **Fecha** (Año > Mes > Día) y no con filtros redundantes por cada nivel, establecí **AttributeHierarchyVisible = False** en los atributos **Day, Month y Year**, de forma que únicamente se exponga la jerarquía construida.



**Evidencia (Figura F12.1.2):** AttributeHierarchyVisible configurado para ocultar niveles individuales.

### 2.3.1.6 F12.2.2 — Confirmación de visibilidad/ajustes en la jerarquía temporal

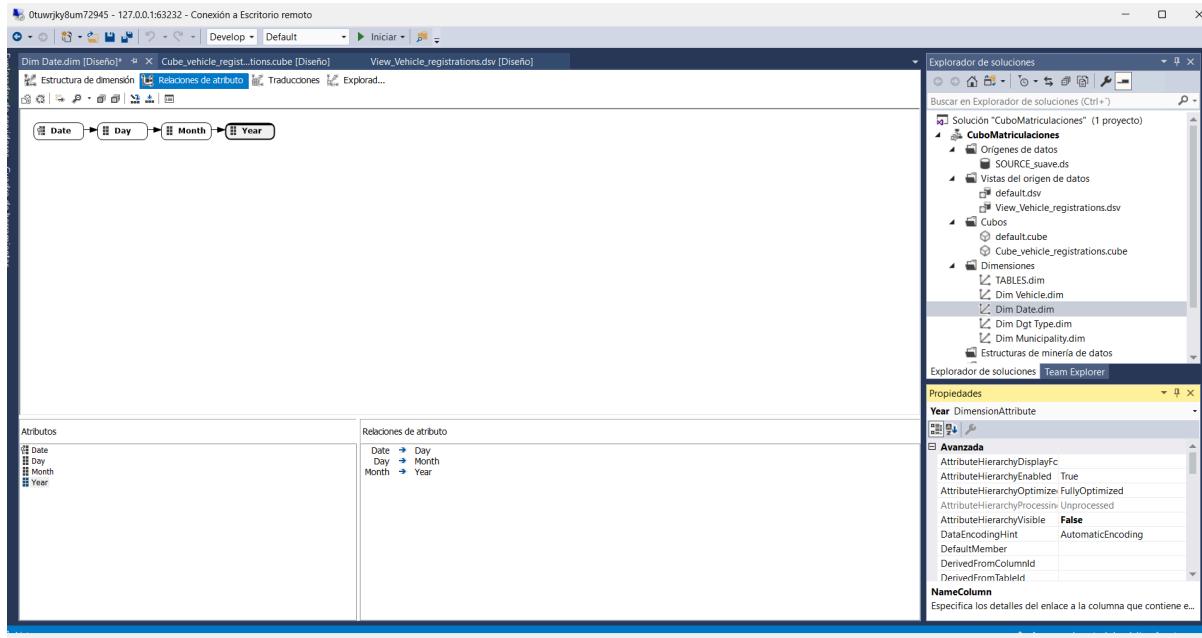
Revisé la configuración final de la dimensión para confirmar que los atributos implicados en la jerarquía quedan ocultos de forma individual y que el comportamiento de la jerarquía temporal es el esperado.



**Evidencia (Figura F12.2.2):** confirmación de configuración de visibilidad/propiedades en DIM\_DATE.

### 2.3.1.7 F12.3 — Estado final de configuración de DIM\_DATE (validación adicional)

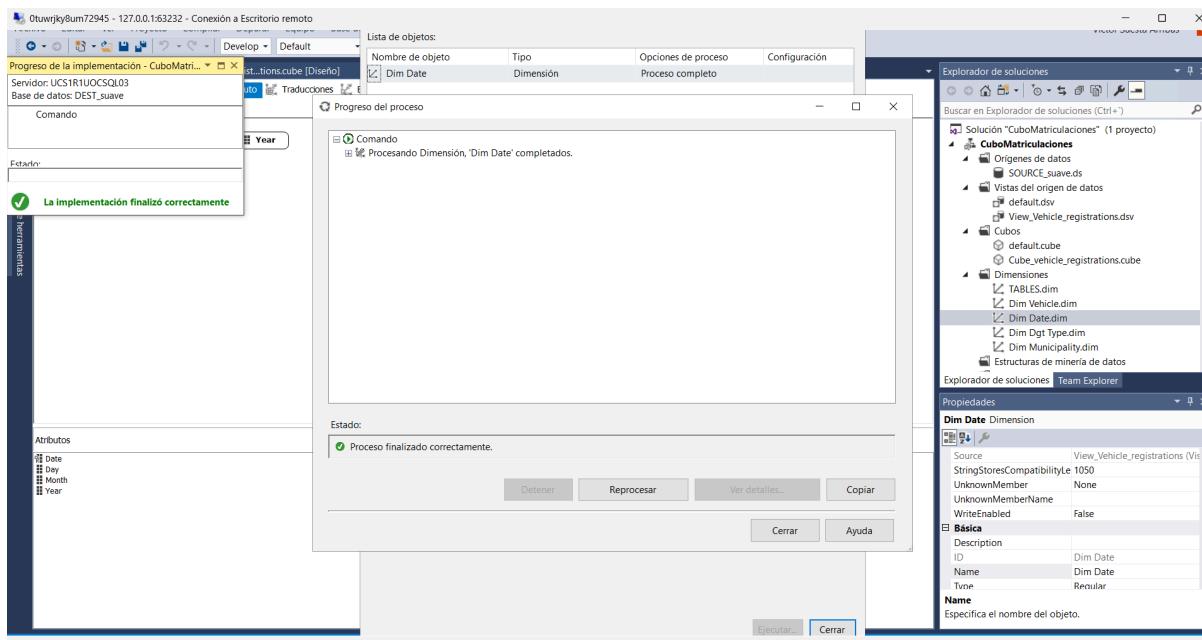
Finalmente validé el estado final de la dimensión tras aplicar todos los cambios de jerarquía, relaciones y visibilidad, asegurando que la dimensión queda consistente antes de proceder al procesamiento.



**Evidencia (Figura F12.3):** validación final de la configuración de DIM\_DATE.

### 2.3.1.8 F13 — Procesamiento de DIM\_DATE finalizado correctamente

Una vez configurada la dimensión temporal ejecuté su **procesamiento** para materializar miembros y jerarquías en el servidor. Confirmé que la implementación/procesamiento finaliza correctamente, lo cual indica que las claves y relaciones están bien definidas y no generan errores.

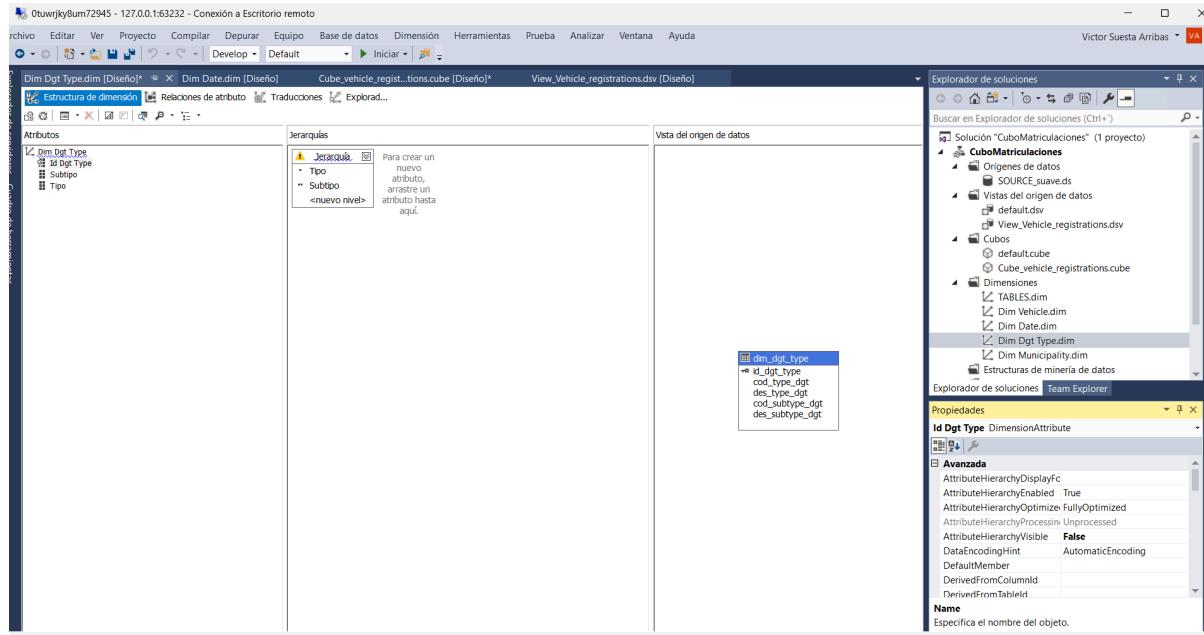


**Evidencia (Figura F13):** “Proceso finalizado correctamente” / “La implementación finalizó correctamente” para DIM\_DATE.

## 2.3.2 Configuración de la dimensión DIM\_DGT\_TYPE

### 2.3.2.1 F14 — Jerarquía Tipo → Subtipo en DIM\_DGT\_TYPE

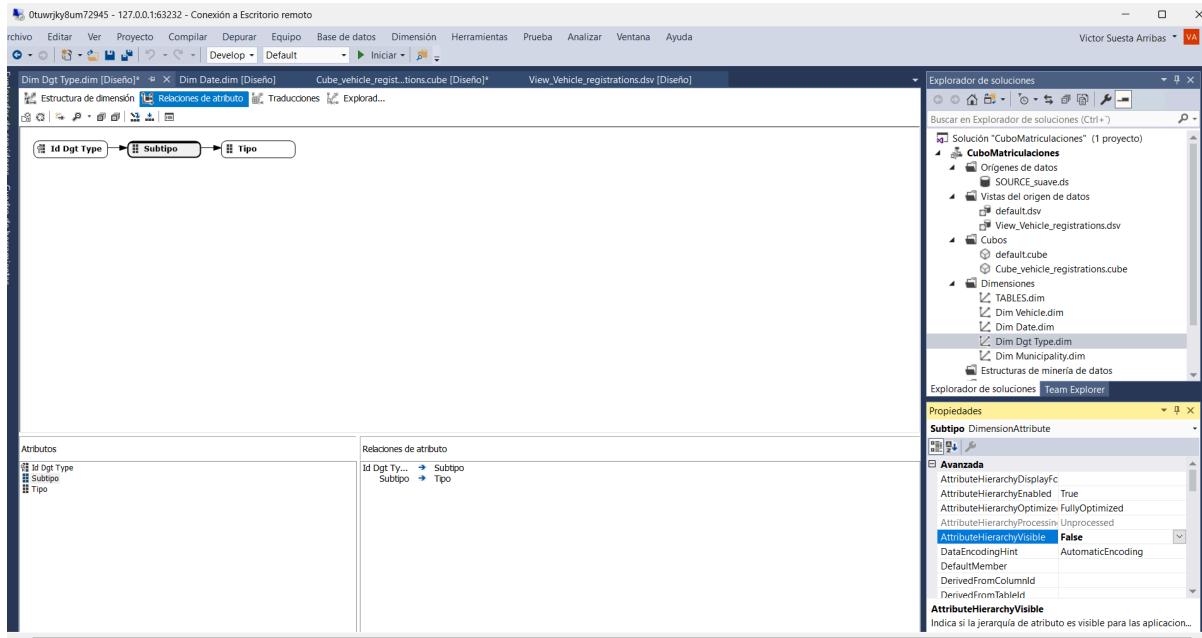
A continuación abrí la dimensión **DIM\_dgt\_type** y construí la jerarquía requerida con dos niveles **Tipo** → **Subtipo**, renombrando los atributos para que la dimensión sea interpretable durante el análisis.



**Evidencia (Figura F14):** jerarquía Tipo–Subtipo definida en DIM\_dgt\_type.

### 2.3.2.2 F15.1 — Relaciones de atributo en DIM\_DGT\_TYPE (Id Dgt Type → Subtipo; Subtipo → Tipo)

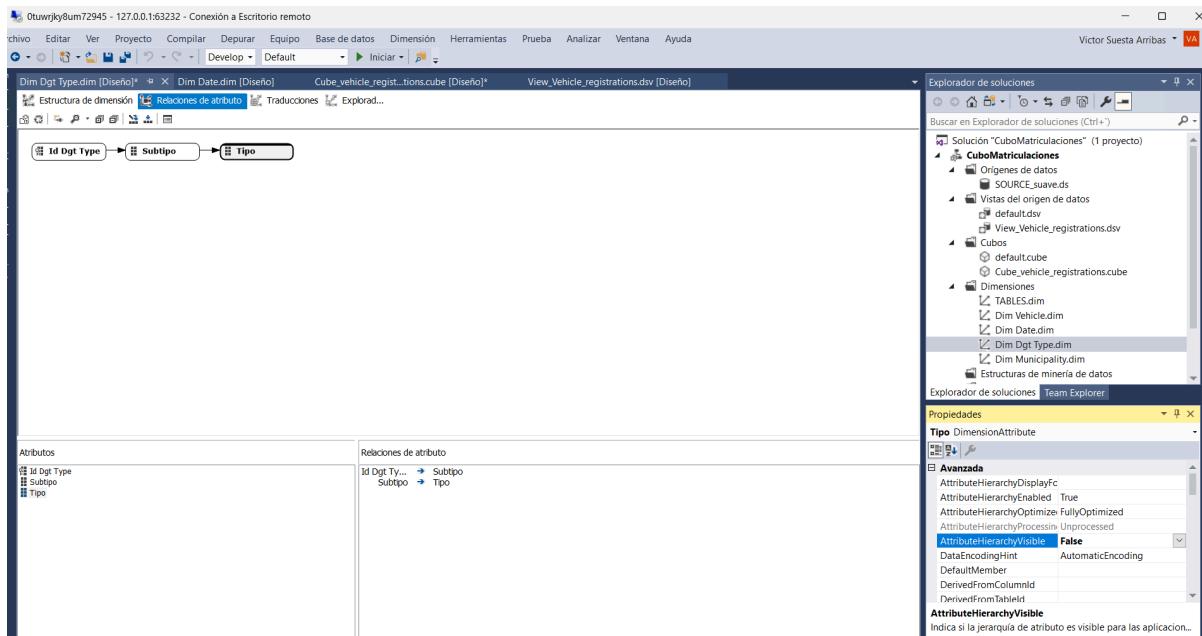
En la pestaña de **Relaciones de atributo** verifiqué que la granularidad queda correctamente reflejada: **Id Dgt Type** → **Subtipo** y **Subtipo** → **Tipo**. Esta configuración es coherente con la jerarquía definida y optimiza la navegación.



**Evidencia (Figura F15.1): relaciones de atributo configuradas en DIM\_dgt\_type.**

### 2.3.2.3 F15.2 — AttributeHierarchyVisible = False en atributos (mostrar solo la jerarquía)

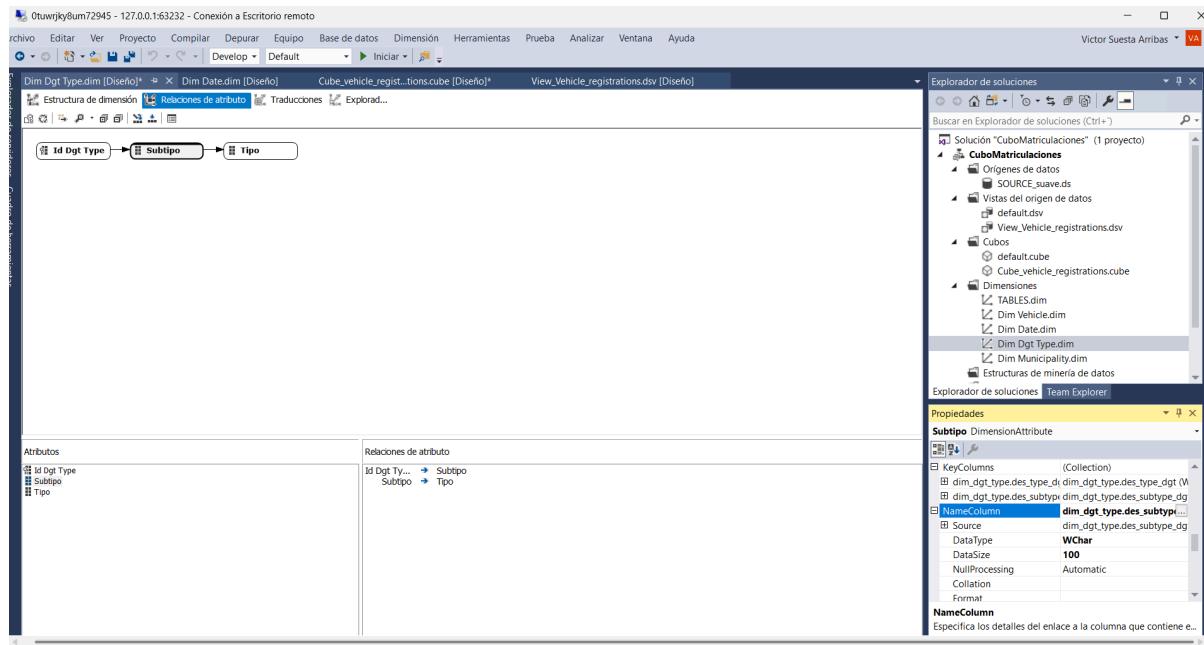
Para que el cubo muestre principalmente la jerarquía construida y no atributos individuales como jerarquías independientes, configuro **AttributeHierarchyVisible = False** en los atributos relevantes, dejando el análisis orientado a la jerarquía Tipo–Subtipo.



**Evidencia (Figura F15.2): AttributeHierarchyVisible desactivado en atributos de DIM\_dgt\_type.**

### 2.3.2.4 F16.1 — Subtipo: KeyColumns compuesta y NameColumn

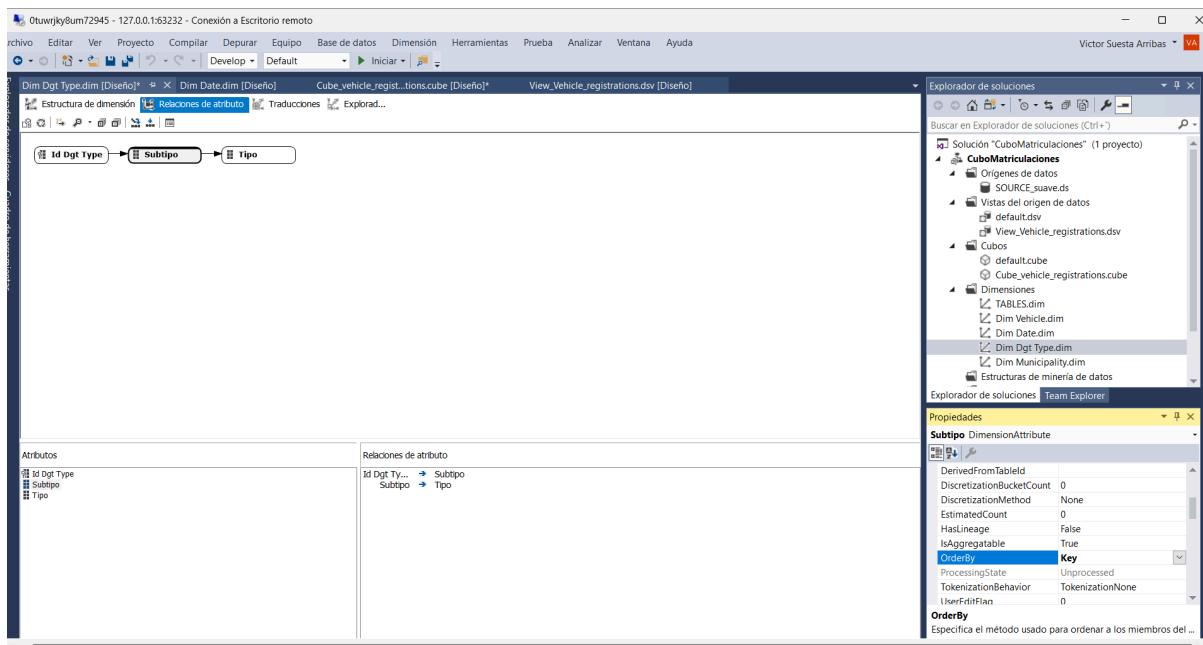
Configuré el atributo **Subtipo** con una clave compuesta para garantizar unicidad por combinación: **KeyColumns = des\_type\_dgt** y **des\_subtype\_dgt**. Además, establecí **NameColumn = des\_subtype\_dgt** para que el valor mostrado en la jerarquía sea el subtipo textual.



**Evidencia (Figura F16.1):** propiedades de Subtipo (KeyColumns y NameColumn).

### 2.3.2.5 F16.2 — Subtipo: OrderBy = Key (ordenación consistente)

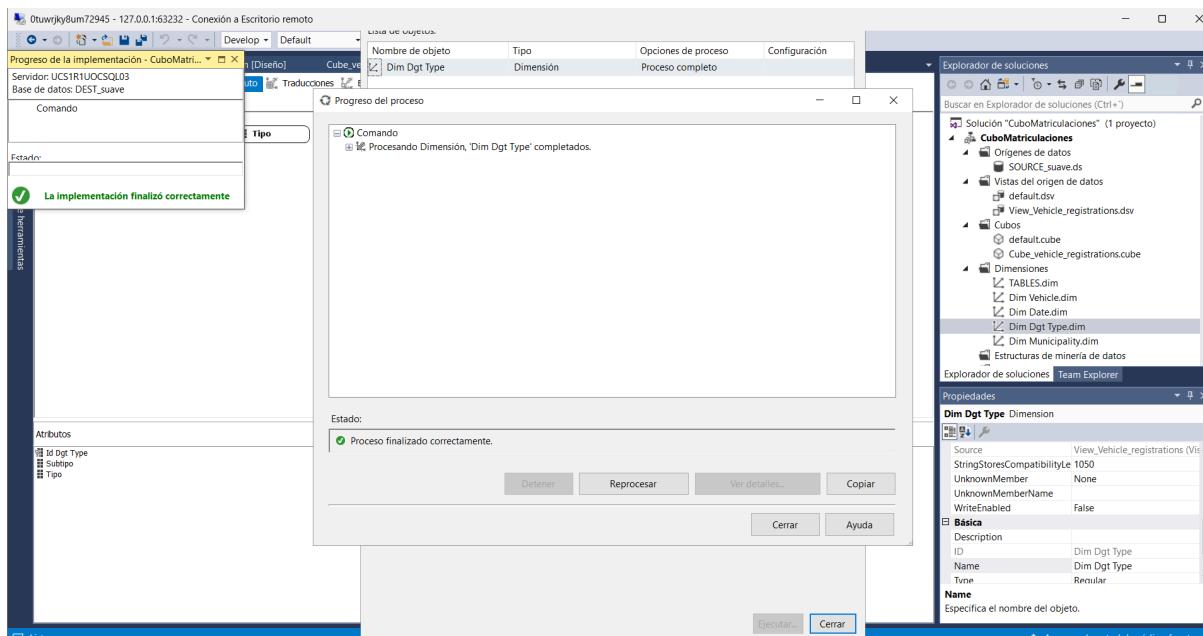
Ajusté la propiedad **OrderBy = Key** en el atributo **Subtipo** para mantener una ordenación consistente basada en la clave, evitando ordenaciones lexicográficas no deseadas.



**Evidencia (Figura F16.2):** propiedad OrderBy = Key aplicada en Subtipo.

### 2.3.2.6 F17.1 — Procesamiento/despliegue de DIM\_DGT\_TYPE finalizado correctamente

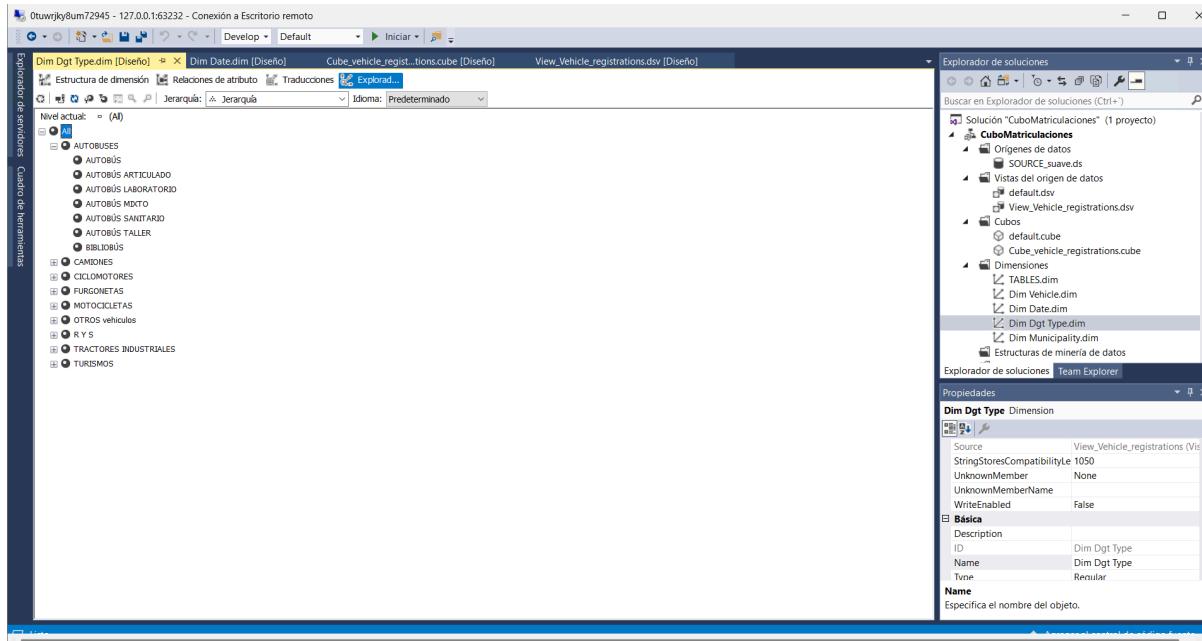
Una vez aplicada la configuración, procesé la dimensión **DIM\_dgt\_type** y verifiqué que el proceso finaliza correctamente. Esto confirma que la jerarquía, relaciones y claves están definidas de forma válida para SSAS.



**Evidencia (Figura F17.1):** “Proceso finalizado correctamente” en el procesamiento de **DIM\_dgt\_type**.

### 2.3.2.7 F17.2 — Exploración de DIM\_DGT\_TYPE (jerarquía navegable)

Tras el procesamiento, accedí a la pestaña de exploración de la dimensión y comprobé que la jerarquía **Tipo** → **Subtipo** es navegable, mostrando categorías y subcategorías de forma coherente.

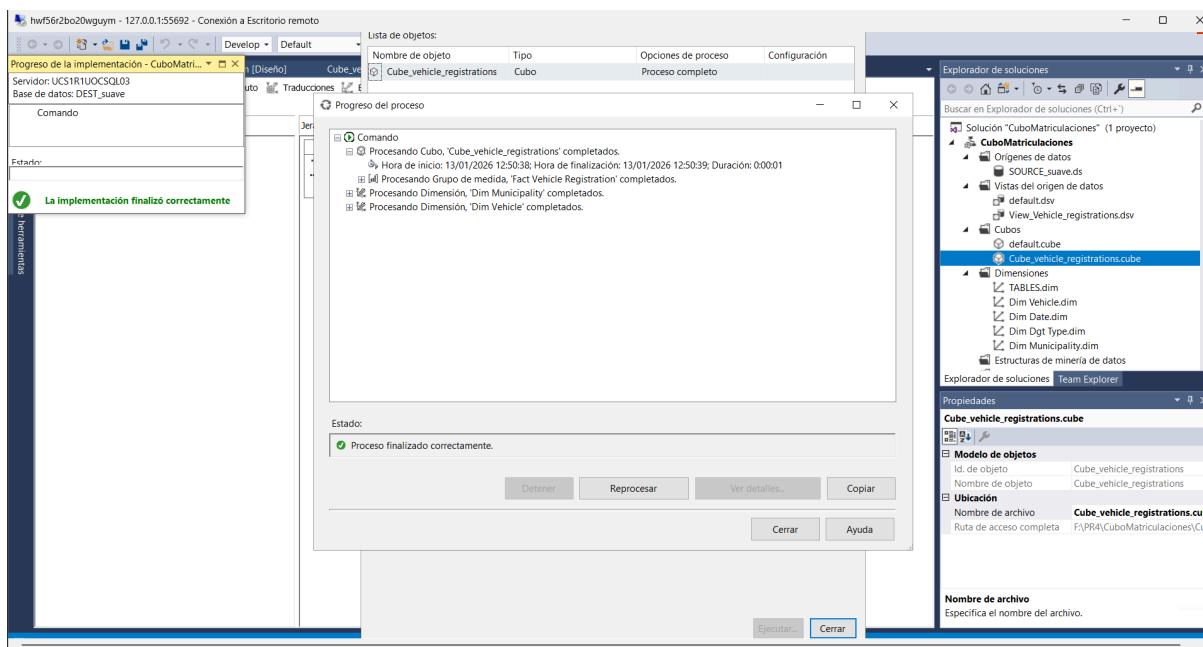


**Evidencia (Figura F17.2):** exploración de DIM\_dgt\_type con jerarquía y miembros visibles.

## 2.4 Procesamiento y despliegue del cubo

### 2.4.1 F18 — Procesamiento/despliegue del cubo Cube\_vehicle\_registrations

Con las dimensiones configuradas, ejecuté el **procesamiento del cubo** completo. Este paso es imprescindible porque es el que carga datos, resuelve relaciones, materializa miembros y genera agregaciones para que el cubo pueda responder consultas multidimensionales.



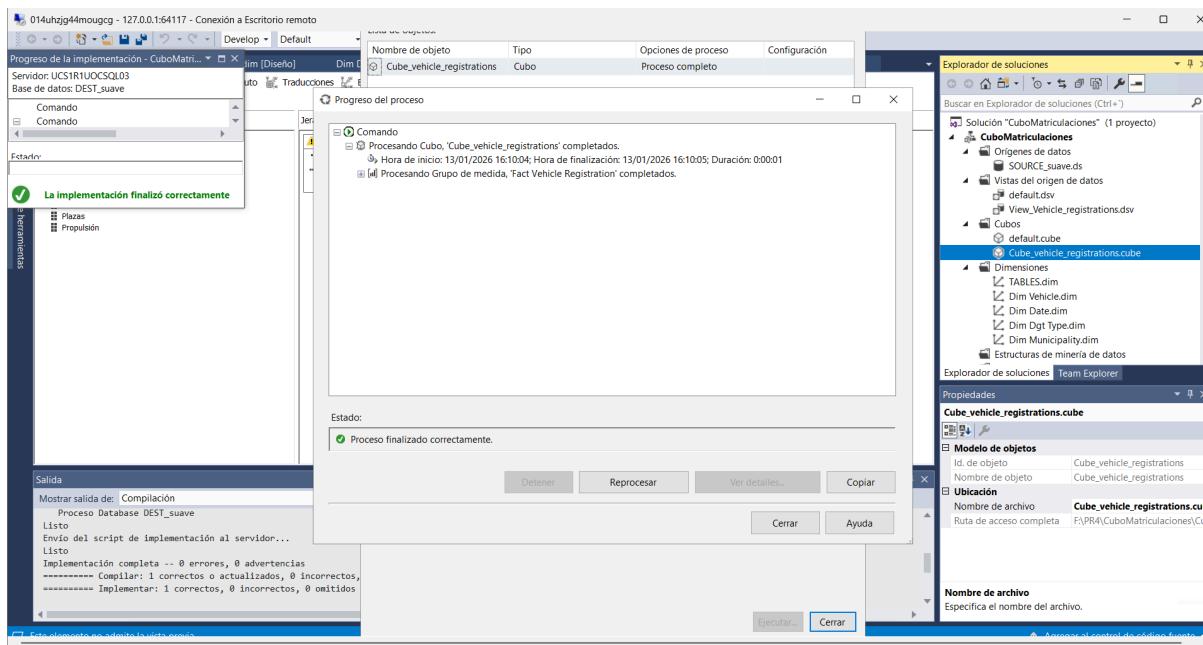
**Evidencia (Figura F18): procesamiento del cubo con detalle de tareas completadas.**

#### 2.4.2 Explicación: por qué es necesario procesar

El procesamiento es necesario porque el cubo no es solo una definición lógica: Analysis Services debe **leer los datos del origen, construir las jerarquías y miembros, validar relaciones, y calcular/agregar** la medida (Quantity) a lo largo de las dimensiones. Sin este paso, el cubo no ofrece resultados fiables o directamente no devuelve datos al explorar.

#### 2.4.3 F27 — Proceso de implementación del cubo finalizado correctamente (validación)

Tras ejecutar el procesamiento verifiqué que el sistema indica explícitamente que la implementación y el proceso han finalizado correctamente para el cubo **Cube\_vehicle\_registrations**. Esto confirma que el cubo queda desplegado y disponible para análisis.

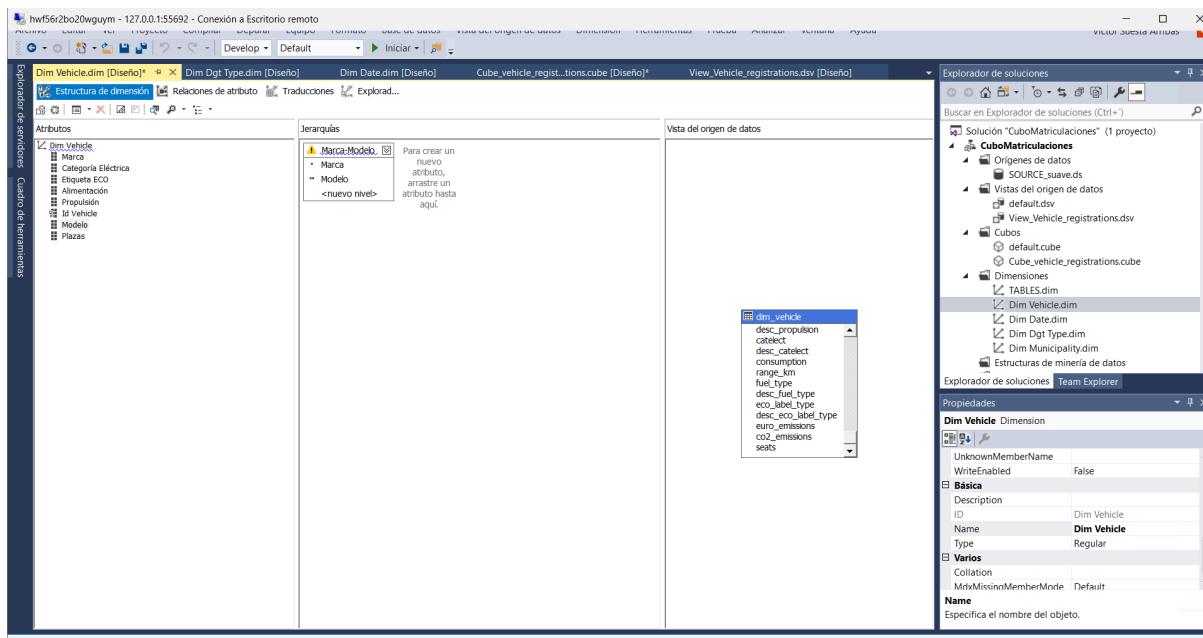


**Evidencia (Figura F27):** “La implementación finalizó correctamente” y “Proceso finalizado correctamente” para Cube\_vehicle\_registrations.

## 2.5 Ejercicios de configuración adicional del cubo (DIM\_Vehicle y DIM\_Municipality)

### 2.5.1 F25 — Configuración de DIM\_Vehicle (atributos renombrados + jerarquía Marca–Modelo)

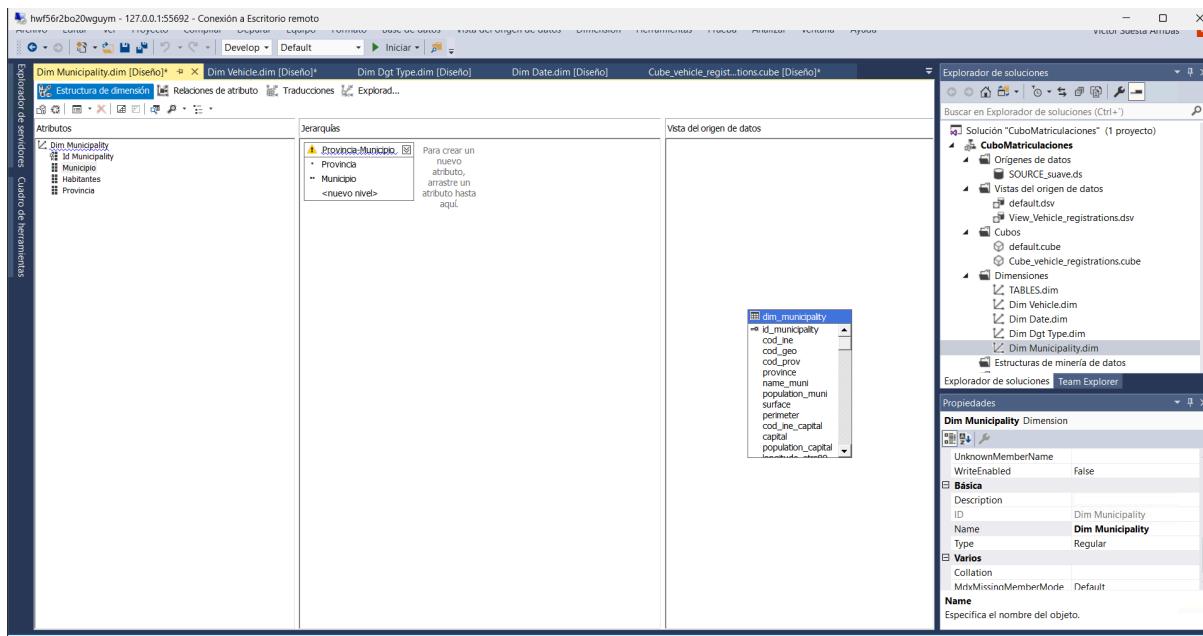
Configuré la dimensión **DIM\_Vehicle** incorporando los atributos solicitados y renombrándolos para mejorar la legibilidad del análisis: **Marca, Modelo, Categoría Eléctrica, Alimentación, Propulsión, Etiqueta ECO y Plazas**. Además construí la jerarquía de dos niveles **Marca → Modelo**. También comprobé que **Id Vehicle** no se presenta como jerarquía, manteniéndose como identificador interno.



**Evidencia (Figura F25): DIM\_Vehicle con atributos renombrados y jerarquía Marca–Modelo.**

## 2.5.2 F26 — Configuración de DIM\_Municipality (atributos renombrados + jerarquía Provincia–Municipio)

Después configuré la dimensión **DIM\_Municipality**, incluyendo y renombrando los atributos requeridos: **Provincia, Municipio y Habitantes**. Construí la jerarquía **Provincia → Municipio** y verifiqué que no se exponen jerarquías adicionales innecesarias.

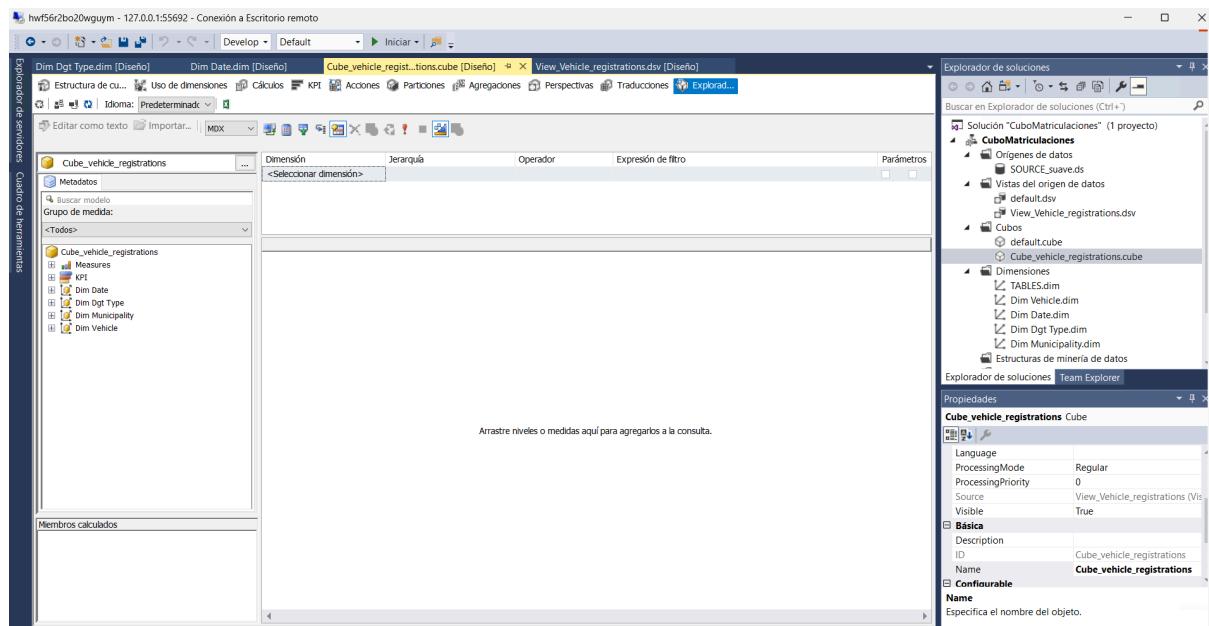


**Evidencia (Figura F26):** DIM\_Municipality con atributos renombrados y jerarquía Provincia–Municipio.

## 2.6 Explotación del cubo OLAP (Explorador de cubos)

### 2.6.1 F19 — Apertura del Explorador del cubo (estado inicial)

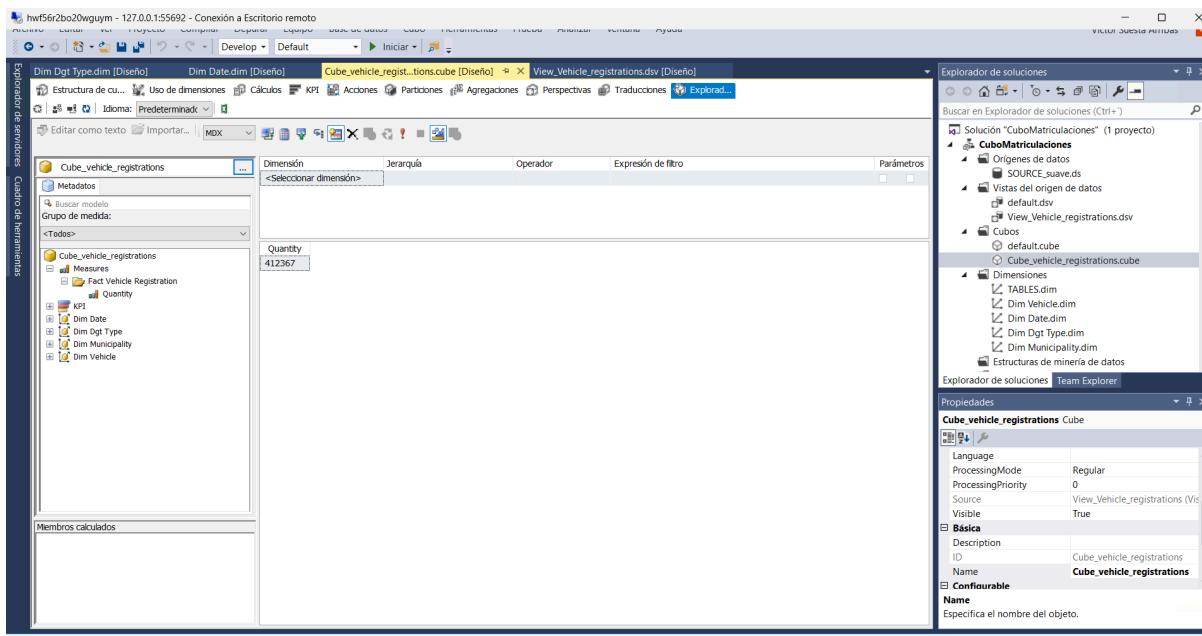
Tras el despliegue abrí el diseñador del cubo y accedí a la pestaña **Explorador**, preparando el entorno para realizar consultas multidimensionales arrastrando medidas, niveles y filtros.



**Evidencia (Figura F19):** vista inicial del Explorador del cubo Cube\_vehicles\_registration.

### 2.6.2 F20 — Prueba de navegación: medida Quantity

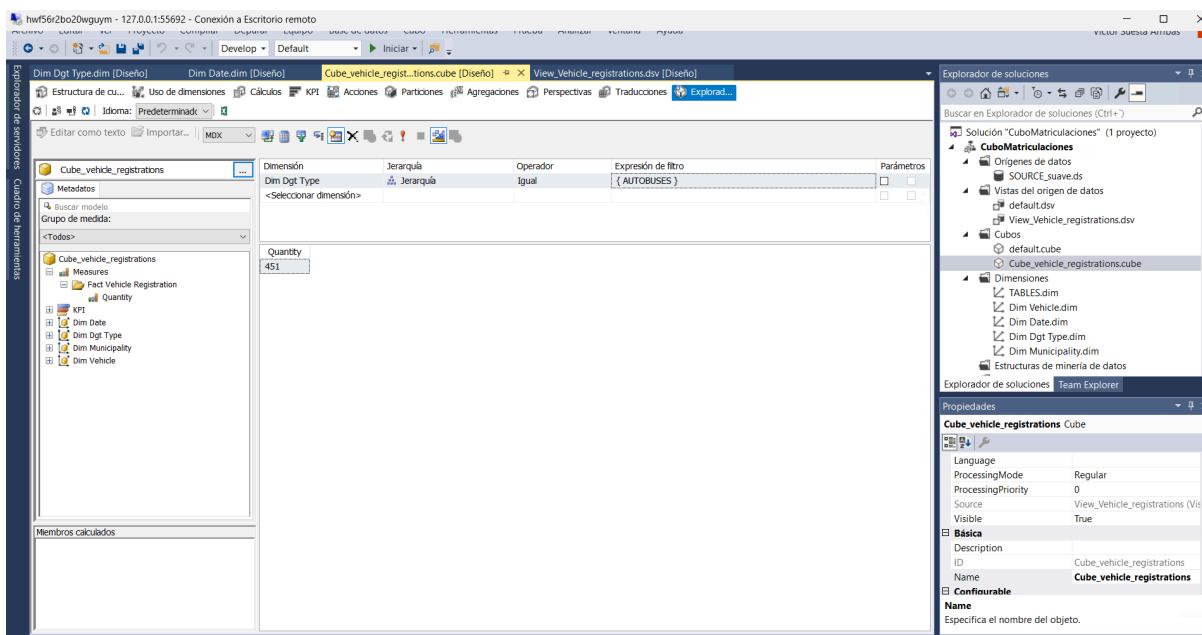
Realicé una primera comprobación funcional arrastrando la medida **Quantity** al área de resultados y ejecutando la consulta. El cubo devolvió un valor agregado, confirmando que la medida está disponible y que el cubo responde correctamente.



**Evidencia (Figura F20):** resultado de Quantity en el explorador del cubo.

### 2.6.3 F21 — Verificación de filtros: Dim Dgt Type = {AUTOBUSES}

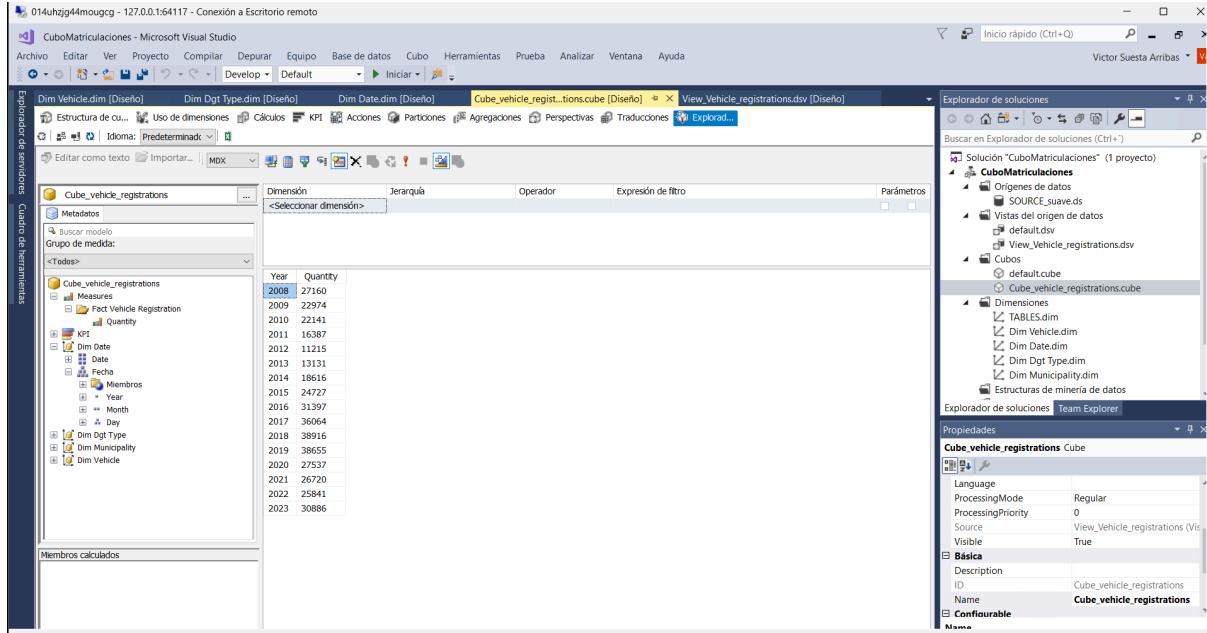
Para comprobar el funcionamiento de los filtros, arrastré la dimensión **Dim Dgt Type** al área de filtros y seleccioné el valor **AUTOBUSES**. El resultado se actualizó automáticamente, mostrando el agregado de **Quantity** condicionado por el filtro aplicado.



**Evidencia (Figura F21):** filtro {AUTOBUSES} aplicado y Quantity actualizado.

## 2.6.4 F28 — Consulta 1: total de matriculaciones por año

Finalmente ejecuté la consulta solicitada de listado por año. Para ello desglosé la medida **Quantity** por el nivel **Year** de la dimensión temporal, obteniendo el total de matriculaciones por cada año disponible en el cubo.



**Evidencia (Figura F28):** tabla de Quantity por Year (matriculaciones por año).

## 2.7 Limitación del entorno en la fase final

A partir de este punto no fue posible continuar con el resto de consultas solicitadas (consultas 2–5 del explorador y las consultas 6–8 en Excel) debido a un **fallo del entorno/laboratorio** que impidió proseguir la explotación del cubo con normalidad. Por este motivo, he incluido todas las evidencias disponibles hasta el punto en el que el entorno permitió trabajar de forma consistente, dejando documentada la configuración completa del cubo, su procesamiento y las verificaciones de navegación realizadas.

## Ejercicio 3. Análisis de datos mediante Dashboards