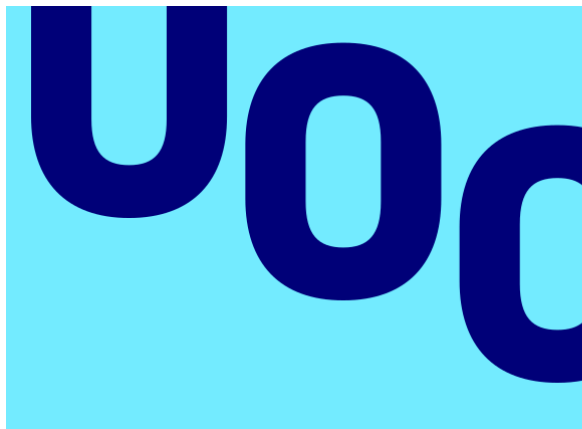


# BASES DE DATOS ANALÍTICAS

## Práctica 2

Noviembre 2025



Universitat  
Oberta  
de Catalunya

Víctor Suesta Arribas

# ÍNDICE

## ÍNDICE

### Ejercicio 1 — Diseño conceptual

#### 1.1 Revisión del modelo conceptual

#### 1.2 Evolución del modelo conceptual

### Ejercicio 2 — Diseño lógico

#### 2.1 Esquema lógico final (PK/AK/FK)

#### 2.2 Justificación de cambios

### Ejercicio 3 — Modelo físico SQL

#### 3.1 Carga de datos y verificación (con evidencias)

##### Evidencia 1

##### Evidencia 2

##### Evidencia 3

##### Evidencia 4

##### Evidencia 5

##### Evidencia 6

#### 3.2 Consultas solicitadas: SQL + explicación

#### 3.3 Corrección de sentencias SQL

##### Caso A — JOIN implícito con coma

##### Caso B — Precedencia AND/OR sin paréntesis

##### Caso C — LEFT JOIN “anulado” por condición en WHERE

### Conclusión breve

### Apéndice

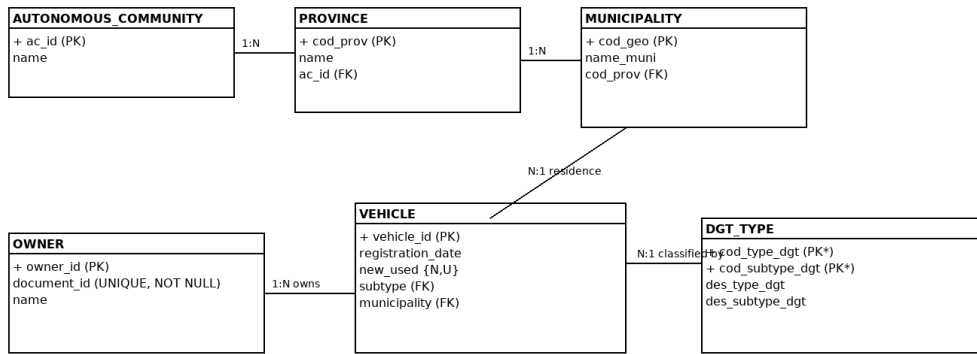
# Ejercicio 1 — Diseño conceptual

En 1.1 se corrigen solo inconsistencias del modelo base; la multipropiedad y las sanciones se tratan en 1.2.

## 1.1 Revisión del modelo conceptual

En esta sección se revisa el modelo conceptual propuesto, identificando errores y omisiones en entidades, atributos y relaciones. Para cada incidencia se justifica la corrección con base en los requisitos del enunciado y en principios del modelo relacional (identificación unívoca, integridad referencial, dominios y cardinalidades consistentes). A continuación se presenta una tabla “Problema–Justificación–Corrección”.

Problema detectado	Justificación	Corrección propuesta
VEHICLE referencia PROVINCE en lugar de MUNICIPALITY	La residencia del vehículo se define por municipio; la relación con provincia no refleja el nivel real de localización. Teoría: la FK debe apuntar a la entidad que materializa el requisito.	Cambiar la relación a VEHICLE → MUNICIPALITY y eliminar la dependencia directa con PROVINCE.
OWNER sin identificador único ni documento oficial	Requisito: propietario unívocamente identificable; teoría: toda entidad debe tener PK y un atributo identificador con unicidad.	Añadir owner_id (PK) y document_id (UNIQUE, NOT NULL) que admita DNI/NIE/Pasaporte.
DGT_TYPE sin clave primaria definida (tipo/subtipo)	Toda entidad de referencia necesita PK. En los datos de carga, un mismo cod_subtype_dgt puede repetirse bajo distintos cod_type_dgt, por lo que no es único por sí solo.	Definir PK compuesta (cod_type_dgt, cod_subtype_dgt) y referenciar desde VEHICLE por ese subtipo (o por la PK compuesta).
VEHICLE no vincula explícitamente su categoría DGT por clave	Teoría: la clasificación DGT debe ser referencial, no un texto suelto, para garantizar consistencia y permitir consultas por tipo/subtipo.	Incluir FK VEHICLE → DGT_TYPE por el código de subtipo (o por la PK definida en DGT_TYPE).
Dominios y restricciones no documentados (p. ej., new_used)	Buenas prácticas: explicitar dominios para integridad semántica (evitar valores fuera de catálogo).	Documentar new_used ∈ {'N','U'} y demás dominios relevantes (fechas válidas, longitudes de códigos).

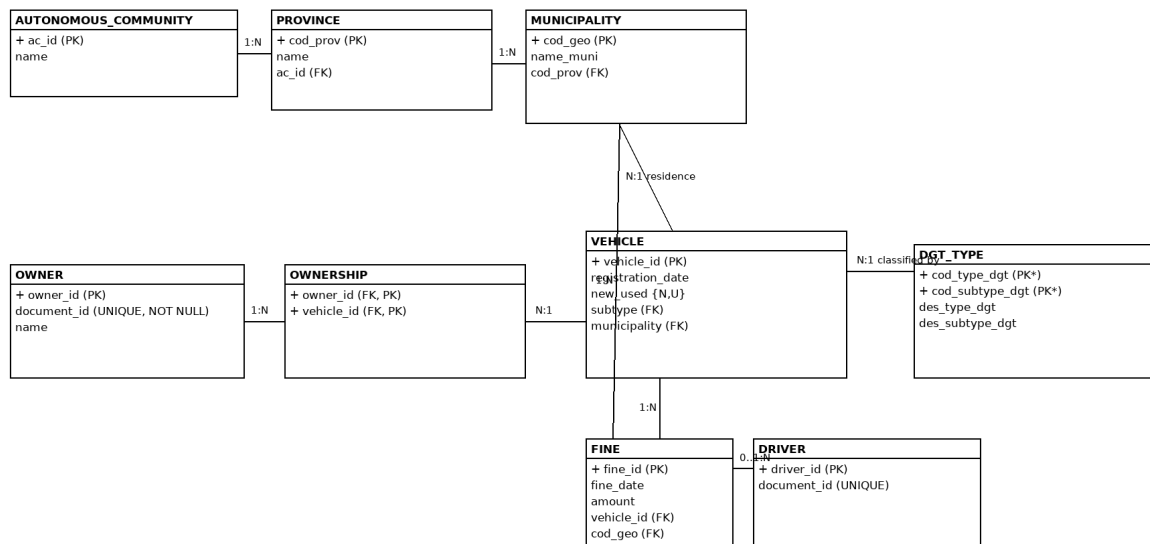


Nota: DGT\_TYPE con PK compuesta (cod\_type\_dgt, cod\_subtype\_dgt).

Figura 1. Diagrama E/R corregido. DGT\_TYPE con PK compuesta (cod\_type\_dgt, cod\_subtype\_dgt).

## 1.2 Evolución del modelo conceptual

- Multipropiedad. Se solicita permitir varios propietarios por vehículo. Para ello se introduce la entidad intermedia OWNERSHIP que materializa una relación N:M entre OWNER y VEHICLE. Su PK compuesta es (owner\_id, vehicle\_id).
- Sanciones y conductor opcional. Se solicita registrar sanciones por vehículo y municipio del hecho, pudiendo identificar el conductor si se conoce. Se añaden las entidades DRIVER (con driver\_id y document\_id único) y FINE con fine\_id, fine\_date, amount, vehicle\_id (FK), cod\_geo de MUNICIPALITY (FK) y driver\_id (FK nullable).



Notas: (i) DGT\_TYPE con PK compuesta (cod\_type\_dgt, cod\_subtype\_dgt). (ii) driver\_id en FINE es opcional (FK nullable).

Figura 2. Diagrama E/R evolucionado. DGT\_TYPE con PK compuesta.

## Ejercicio 2 — Diseño lógico

### 2.1 Esquema lógico final (PK/AK/FK)

DGT\_TYPE (dbo.dgt\_type)

- PK: (cod\_type\_dgt, cod\_subtype\_dgt)
- Atributos: des\_type\_dgt, des\_subtype\_dgt

MUNICIPALITY (dbo.municipalities)

- PK: (cod\_geo)
- Atributos: name\_muni, province, population\_muni, altitude, ...

OWNER (dbo.owner)

- PK: (owner\_id)
- AK: (document\_id) [UNIQUE, NOT NULL]
- Atributos: name, ...

VEHICLE (dbo.vehicles)

- PK: (id)
- FK1: (municipality) → MUNICIPALITY(cod\_geo)
- FK2: (type, subtype) → DGT\_TYPE(cod\_type\_dgt, cod\_subtype\_dgt)
- Dominios: new\_used  $\in \{ 'N', 'U' \}$
- Atributos: type, subtype, brand, model, registration\_date

#### OWNERSHIP (dbo.ownership)

- PK: (owner\_id, vehicle\_id)
- FK1: (owner\_id) → OWNER(owner\_id)
- FK2: (vehicle\_id) → VEHICLE(id)
- Atributos opcionales: ownership\_start\_date, ownership\_end\_date

#### DRIVER (dbo.driver)

- PK: (driver\_id)
- AK: (document\_id) [UNIQUE]
- Atributos: license\_issue, license\_expiry, ...

#### FINE (dbo.fine)

- PK: (fine\_id)
- FK1: (vehicle\_id) → VEHICLE(id)
- FK2: (cod\_geo) → MUNICIPALITY(cod\_geo)
- FK3: (driver\_id) → DRIVER(driver\_id) [NULLABLE]
- Atributos: fine\_date, amount

## 2.2 Justificación de cambios

- VEHICLE → MUNICIPALITY (FK a cod\_geo): la residencia se define por municipio; garantiza integridad y consultas correctas por localidad.
- VEHICLE → DGT\_TYPE (FK a (type, subtype)): la clasificación DGT es referencial; permite filtrar por tipo/subtipo sin incoherencias.
- Multipropiedad: se modela N:M mediante OWNERSHIP con PK compuesta; evita duplicidad y admite trazabilidad temporal si se usan fechas.
- Sanciones: FINE depende de VEHICLE y del MUNICIPALITY del hecho; DRIVER es opcional (FK nullable) para casos sin conductor identificado.
- Unidades civiles: document\_id en OWNER y DRIVER como AK para identificación unívoca.
- Dominios: new\_used  $\in$  {'N','U'} para integridad semántica.

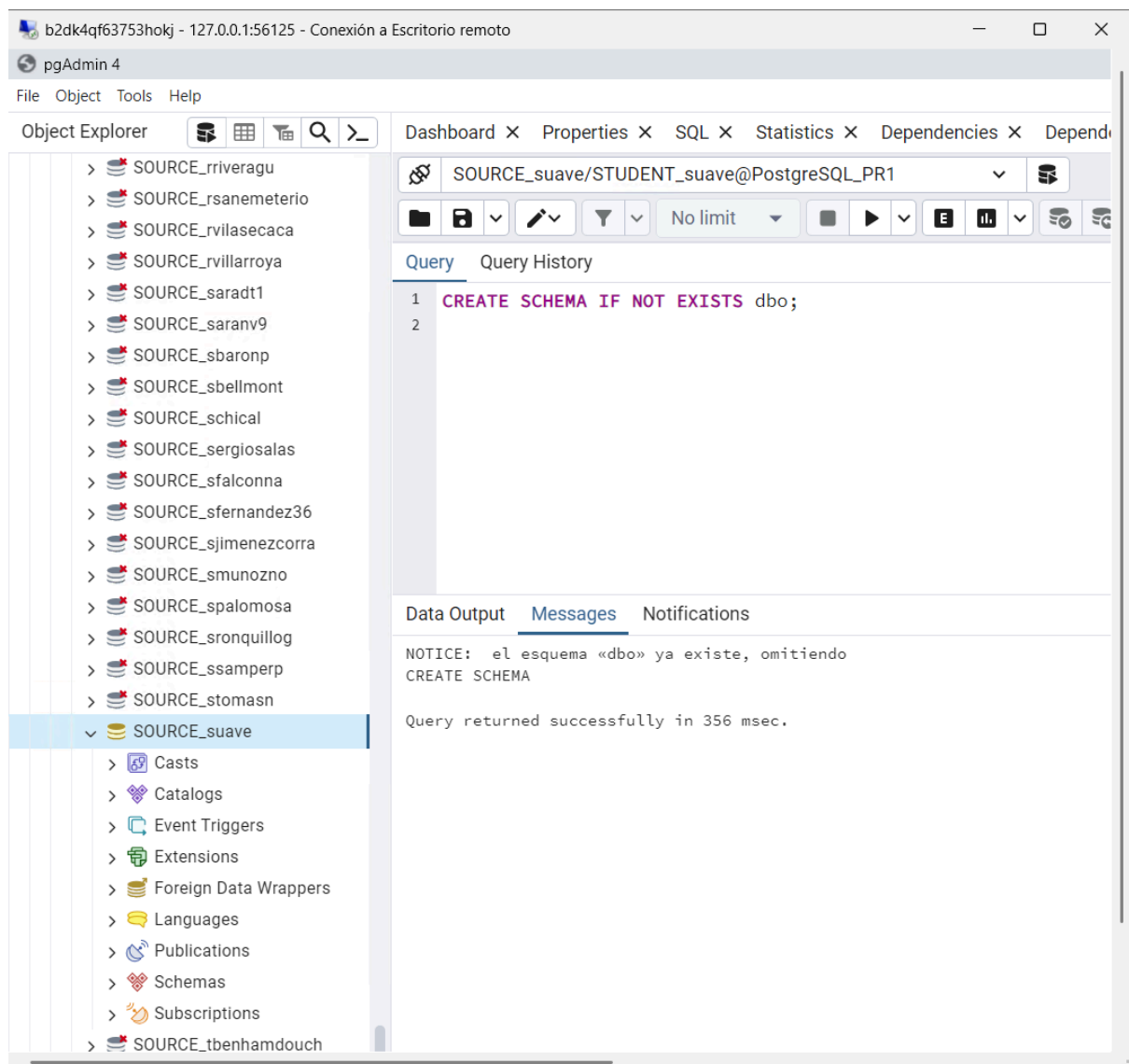
La PK de DGT\_TYPE es compuesta (cod\_type\_dgt,cod\_subtype\_dgt); por ello, en el lógico la FK de VEHICLE se modela sobre (type, subtype). En el físico proporcionado por los scripts, el atributo type no existe y las uniones operativas se realizan por subtype, pero el diseño lógico correcto exige la clave compuesta.

## Ejercicio 3 — Modelo físico SQL

### 3.1 Carga de datos y verificación (con evidencias)

Entorno: PostgreSQL (pgAdmin 4). Se ha creado el esquema dbo y ejecutados los scripts dgt\_type.sql, municipalities.sql y vehicles.sql en este orden. En esta subsección se documentan las evidencias de creación del esquema y de carga/verificación de datos. Para cada evidencia se incluye la captura de pantalla y un pie con el identificador (E1–E6).

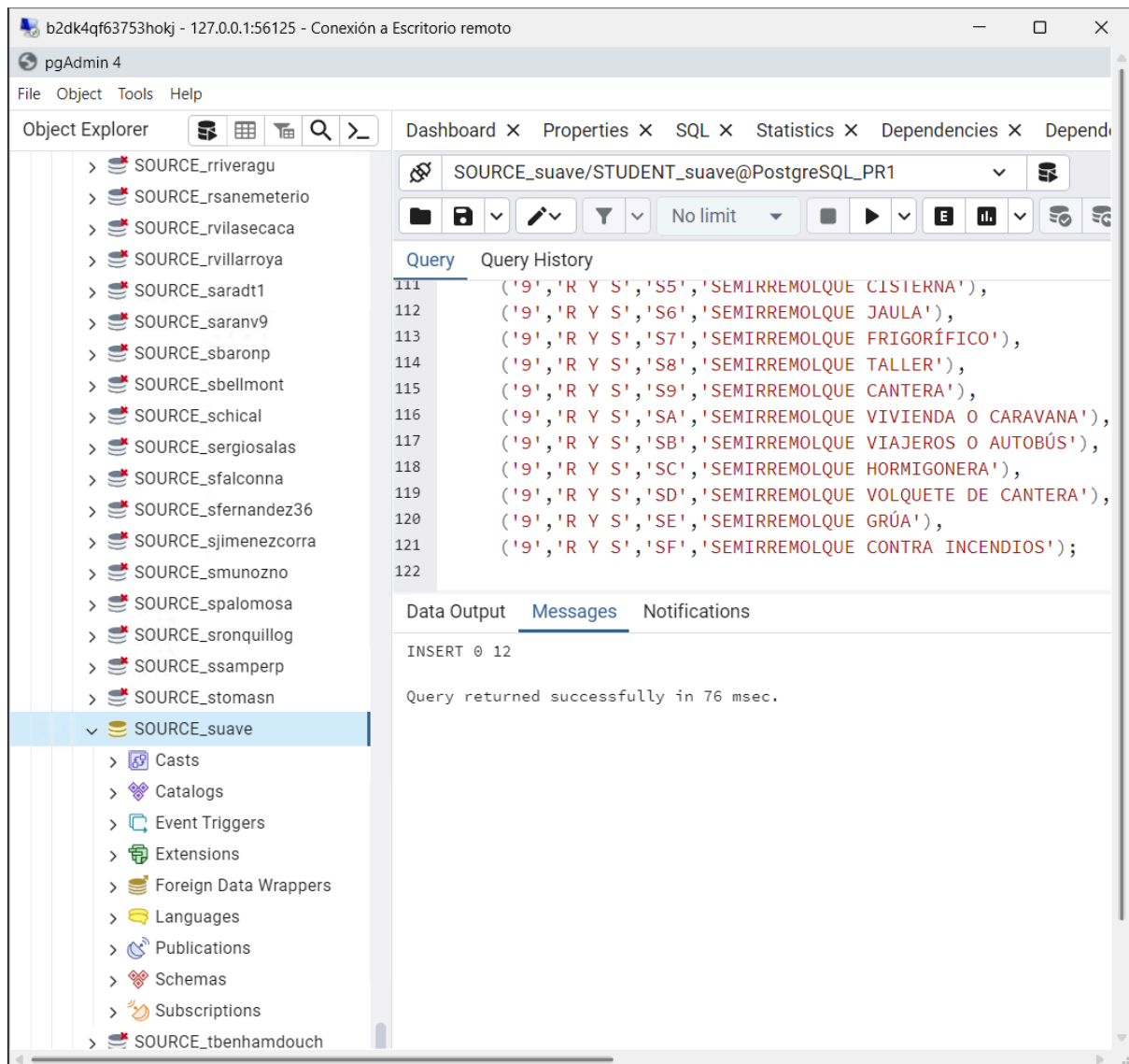
# Evidencia 1



Creación de esquema dbo: se ejecutó CREATE SCHEMA IF NOT EXISTS dbo; y el entorno confirmó la ejecución correcta. (E1\_dbo\_schema\_ok.png)

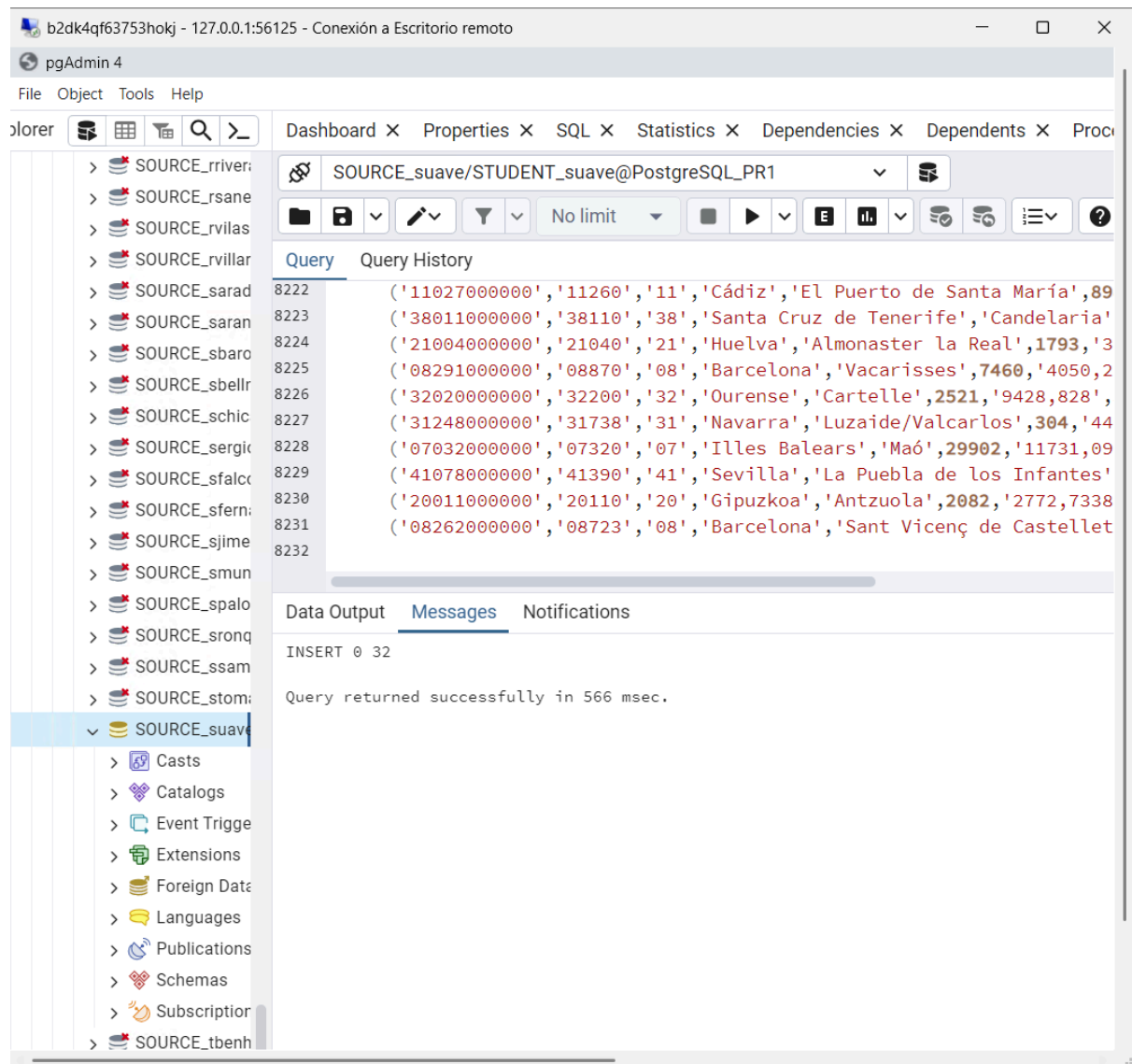


## Evidencia 2



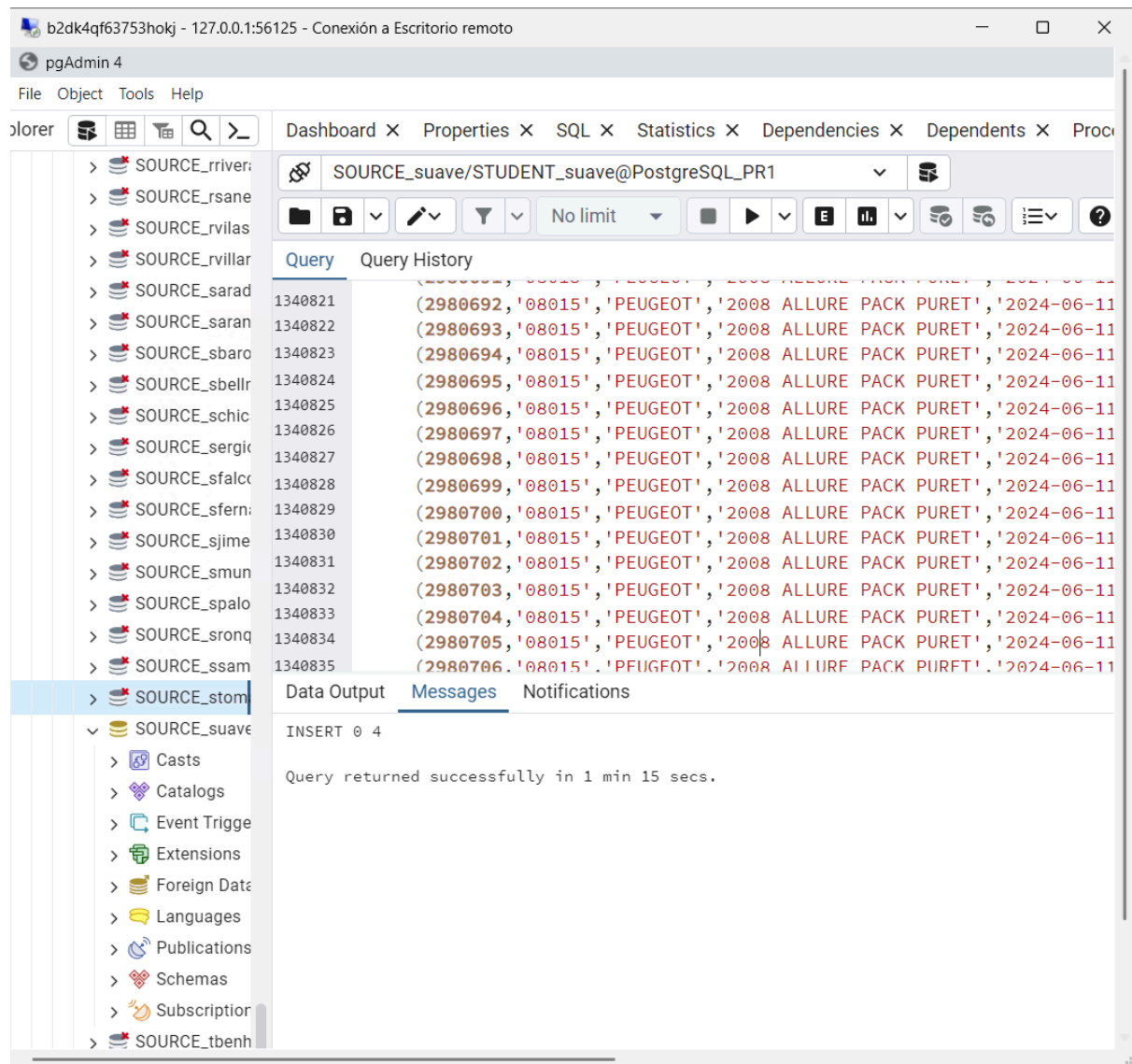
dgt\_type.sql: creación y carga correctas de dbo.dgt\_type. (E2\_dgt\_type\_success.png)

## Evidencia 3



municipalities.sql: creación y carga correctas de dbo.municipalities.  
(E3\_municipalities\_success.png)

## Evidencia 4



vehicles.sql: creación y carga correctas de dbo.vehicles. (E4\_vehicles\_success.png)

## Evidencia 5

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure for 'SOURCE\_suave'. The 'Schemas (2)' folder is expanded, showing 'dbo'. Under 'dbo', the 'Tables (3)' folder is selected, listing 'dgt\_type', 'municipalities', and 'vehicles'. The main pane shows a SQL query in the 'Query' tab:

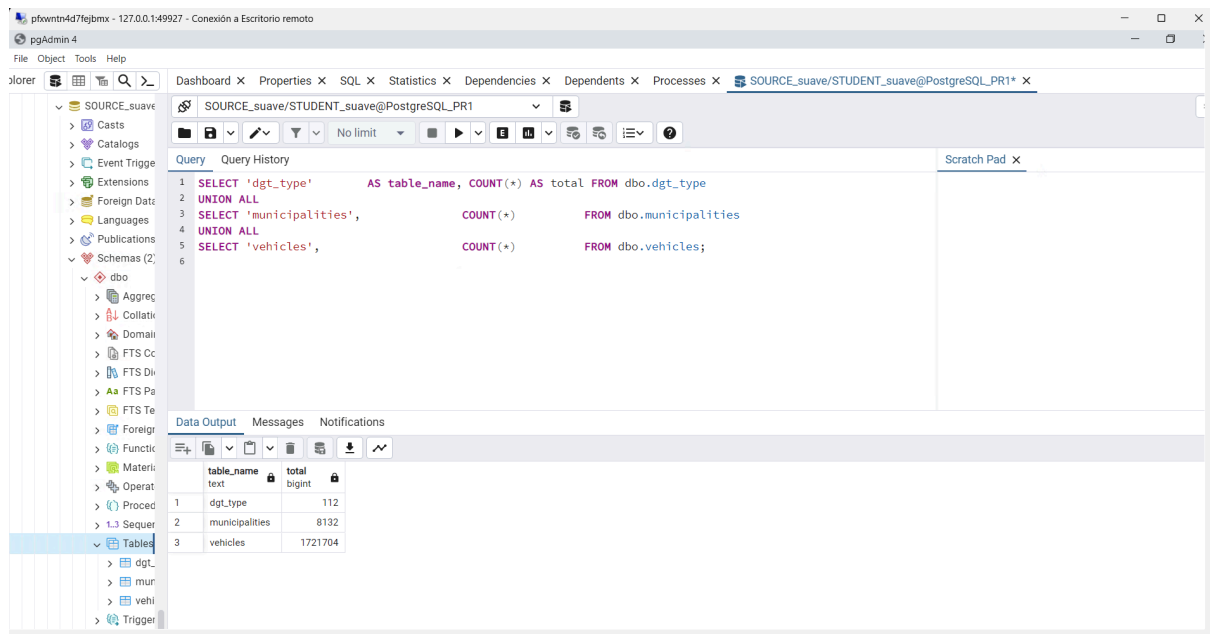
```
1 SELECT table_schema, table_name
2 FROM information_schema.tables
3 WHERE table_schema = 'dbo'
4 ORDER BY table_name;
5
```

Below the query, the 'Data Output' tab displays the results of the query in a table format:

	table_schema name	table_name name
1	dbo	dgt_type
2	dbo	municipalities
3	dbo	vehicles

Tablas en dbo: aparecen dgt\_type, municipalities, vehicles. (E5\_tables\_in\_dbo.png)

## Evidencia 6



dgt\_type = 112 subtipos; municipalities = 8 132 registros; vehicles = 1 721 704 registros. La carga coincide con los scripts y confirma que las tablas están pobladas correctamente. (E6\_counts.png)

### 3.2 Consultas solicitadas: SQL + explicación

- Q1 — Subtipos DGT (ordenados):

```
<<
SELECT
    d.cod_subtype_dgt,
    d.des_subtype_dgt
FROM dbo.dgt_type AS d
ORDER BY d.cod_subtype_dgt, d.des_subtype_dgt;
>>
```

Listado del catálogo DGT (tabla dbo.dgt\_type): se seleccionan los códigos de subtipo y sus descripciones y se ordena de forma estable por código y descripción. Sirve como verificación de referencia de que el diccionario está cargado y consistente antes de hacer consultas o modificaciones. Véase Captura Q1 en el apéndice.

- Q2 — N° municipios “Sevilla”:

```
<<
SELECT COUNT(1) AS total_municipios_sevilla
FROM dbo.municipalities AS m
WHERE m.province = 'Sevilla';
>>
```

Cómputo de municipios de la provincia de Sevilla (tabla dbo.municipalities). El filtro por province = 'Sevilla' y COUNT(1) devuelve 106, confirmando el volumen esperado de registros para esa provincia. Véase Captura Q2 en el apéndice.

- Q3 — Municipios de “Sevilla” que empiezan por A o U (orden DESC):

```
<<
SELECT
    m.name_muni,
    m.province
FROM dbo.municipalities AS m
WHERE m.province = 'Sevilla'
    AND (m.name_muni ILIKE 'A%' OR m.name_muni ILIKE 'U%')
ORDER BY m.name_muni DESC;
>>
```

Filtro por provincia = 'Sevilla' y por nombre que comience por ‘A’ o ‘U’ (ILIKE para mayúsculas/minúsculas). Se muestran los municipios resultantes ordenados en orden descendente por nombre; aparecen 14 filas, lo que valida la condición de prefijo. Véase Captura Q3 en el apéndice.

- Q4 — INSERT controlado de un nuevo subtipo (Z9) + verificación:

```
<<
BEGIN;

INSERT INTO dbo.dgt_type (cod_type_dgt, des_type_dgt, cod_subtype_dgt,
des_subtype_dgt)
VALUES ('4', 'TURISMOS', 'Z9', 'TURISMO PR2');

-- Verificación
SELECT cod_type_dgt, des_type_dgt, cod_subtype_dgt, des_subtype_dgt
FROM dbo.dgt_type
WHERE cod_subtype_dgt = 'Z9';
>>
```

Inserción controlada de un nuevo subtipo DGT con código ‘Z9’ y verificación inmediata: el SELECT posterior muestra exactamente 1 fila con cod\_subtype\_dgt = 'Z9'. Esta evidencia confirma que la operación INSERT fue correcta. Véase Captura Q4 en el apéndice.

- Q5 — UPDATE del subtipo Z9 + verificación:

```
<<
UPDATE dbo.dgt_type
SET des_subtype_dgt = 'TURISMO PR2 ACTUALIZADO'
WHERE cod_subtype_dgt = 'Z9';

-- Verificación
SELECT cod_type_dgt, des_type_dgt, cod_subtype_dgt, des_subtype_dgt
FROM dbo.dgt_type
```

```
WHERE cod_subtype_dgt = 'Z9';  
>>
```

Actualización del texto del subtipo recién insertado ('Z9') y verificación: tras el UPDATE, el SELECT refleja la descripción modificada ("TURISMO PR2 ACTUALIZADO"), evidenciando que el cambio persiste en la tabla. Véase Captura Q5 en el apéndice.

- Q6 — DELETE del subtipo Z9 + comprobación + ROLLBACK:

```
<<  
DELETE FROM dbo.dgt_type  
WHERE cod_subtype_dgt = 'Z9';
```

-- Verificación: no debe devolver filas

```
SELECT 1  
FROM dbo.dgt_type  
WHERE cod_subtype_dgt = 'Z9';  
>>
```

Eliminación del subtipo de prueba 'Z9' y verificación de ausencia de filas (el SELECT ya no devuelve resultados). La transacción iniciada en Q4 (BEGIN) se mantiene abierta hasta Q6 y se cierra con ROLLBACK, dejando la BD como antes de Q4–Q6. Véase Captura Q6 en el apéndice.

- Q7 — Turismos con altitud del municipio > 950 (join por marca/modelo):

```
<<  
SELECT  
    v.brand,  
    v.model,  
    m.name_muni AS municipality,  
    m.altitude,  
    d.des_type_dgt AS dgt_type  
FROM dbo.vehicles AS v  
JOIN dbo.municipalities AS m ON v.municipality = m.cod_geo  
JOIN dbo.dgt_type AS d ON v.subtype = d.cod_subtype_dgt  
WHERE d.des_type_dgt = 'TURISMOS'  
    AND m.altitude > 950  
ORDER BY v.brand ASC, v.model ASC;  
>>
```

Join entre vehicles, municipalities y dgt\_type para listar turismos (des\_type\_dgt = 'TURISMOS') con altitud del municipio > 950 m. Se muestran marca y modelo, municipio y tipo DGT; el orden por marca y modelo facilita detectar duplicidades/consistencias. Véase Captura Q7 en el apéndice.

- Q8 — N° de vehículos por municipio (conteo simple):

```
<<  
SELECT
```

```

    m.name_muni AS municipality,
    COUNT(*) AS total_vehicles
FROM dbo.vehicles AS v
JOIN dbo.municipalities AS m ON v.municipality = m.cod_geo
GROUP BY m.name_muni
ORDER BY m.name_muni ASC;
>>

```

Agregación por municipio: COUNT(\*) de vehículos tras unir vehicles con municipalities por el código geográfico. El resultado devuelve un recuento por municipio y se ordena alfabéticamente, permitiendo comparar tamaños relativos. Véase Captura Q8 en el apéndice.

- Q9 — Municipios con < 15 000 vehículos (HAVING):  
<<  
SELECT  
 m.name\_muni AS municipality,  
 COUNT(\*) AS total\_vehicles  
FROM dbo.vehicles AS v  
JOIN dbo.municipalities AS m ON v.municipality = m.cod\_geo  
GROUP BY m.name\_muni  
HAVING COUNT(\*) < 15000  
ORDER BY m.name\_muni ASC;  
>>

Misma agregación que Q8 pero con cláusula HAVING COUNT(\*) < 15000 para filtrar municipios con menos de 15 000 vehículos. El resultado devuelve 2 municipios, mostrando el uso correcto de HAVING sobre la agregación. Véase Captura Q9 en el apéndice.

- Q10 — N° de vehículos por tipo DGT × nuevo/usado (tabla cruzada básica):  
<<  
SELECT  
 d.des\_type\_dgt AS dgt\_type,  
 v.new\_used AS new\_or\_used, -- 'N' nuevo, 'U' usado  
 COUNT(\*) AS total\_vehicles  
FROM dbo.vehicles AS v  
JOIN dbo.dgt\_type AS d ON v.subtype = d.cod\_subtype\_dgt  
GROUP BY d.des\_type\_dgt, v.new\_used  
ORDER BY d.des\_type\_dgt ASC, v.new\_used ASC;  
>>

Cruce de vehicles con el catálogo DGT por subtipo para obtener el tipo DGT (des\_type\_dgt) y distribución por estado del vehículo ('N' nuevo / 'U' usado). El GROUP BY (tipo, new\_or\_used) y el ORDER BY muestran un resumen claro de totales por categoría y condición. Véase Captura Q10 en el apéndice.



### 3.3 Corrección de sentencias SQL

A continuación se muestran tres ejemplos típicos de errores y su versión corregida, indicando por qué la segunda cumple los principios de SQL y del modelo.

#### Caso A — JOIN implícito con coma

Errónea:

```
<<
SELECT m.name_muni, v.brand
FROM dbo.municipalities m, dbo.vehicles v
WHERE v.municipality = m.cod_geo
      AND m.province = 'Sevilla';
>>
```

Corregida (JOIN explícito):

```
<<
SELECT m.name_muni, v.brand
FROM dbo.vehicles AS v
JOIN dbo.municipalities AS m
      ON v.municipality = m.cod_geo
WHERE m.province = 'Sevilla';
>>
```

Por qué ahora cumple:

Se emplea JOIN ... ON con la condición de unión separada del filtro. Mejora legibilidad, evita cruces accidentales y respeta la semántica de la relación FK  
vehicles.municipality → municipalities.cod\_geo.

#### Caso B — Precedencia AND/OR sin paréntesis

Errónea:

```
<<
SELECT d.cod_subtype_dgt, d.des_subtype_dgt
FROM dbo.dgt_type AS d
WHERE d.des_type_dgt = 'TURISMOS' OR d.des_subtype_dgt ILIKE 'A%'
      AND d.cod_subtype_dgt <> 'Z9';
>>
```

Corregida (paréntesis claros):

```
<<
SELECT d.cod_subtype_dgt, d.des_subtype_dgt
FROM dbo.dgt_type AS d
WHERE (d.des_type_dgt = 'TURISMOS' OR d.des_subtype_dgt ILIKE 'A%')
```

```
    AND d.cod_subtype_dgt <> 'Z9';  
>>
```

Por qué ahora cumple:

La precedencia en SQL evalúa AND antes que OR. Los paréntesis fuerzan el criterio pretendido: primero “TURISMOS o subtipo empieza por A”, y después excluir Z9.

## Caso C — LEFT JOIN “anulado” por condición en WHERE

Errónea:

```
<<  
SELECT m.name_muni, COUNT(v.id) AS total  
FROM dbo.municipalities AS m  
LEFT JOIN dbo.vehicles AS v  
    ON v.municipality = m.cod_geo  
WHERE v.new_used = 'N'  
GROUP BY m.name_muni;  
>>
```

Corregida (condición en la cláusula ON):

```
<<  
SELECT m.name_muni, COUNT(v.id) AS total  
FROM dbo.municipalities AS m  
LEFT JOIN dbo.vehicles AS v  
    ON v.municipality = m.cod_geo  
    AND v.new_used = 'N'  
GROUP BY m.name_muni;  
>>
```

Por qué ahora cumple:

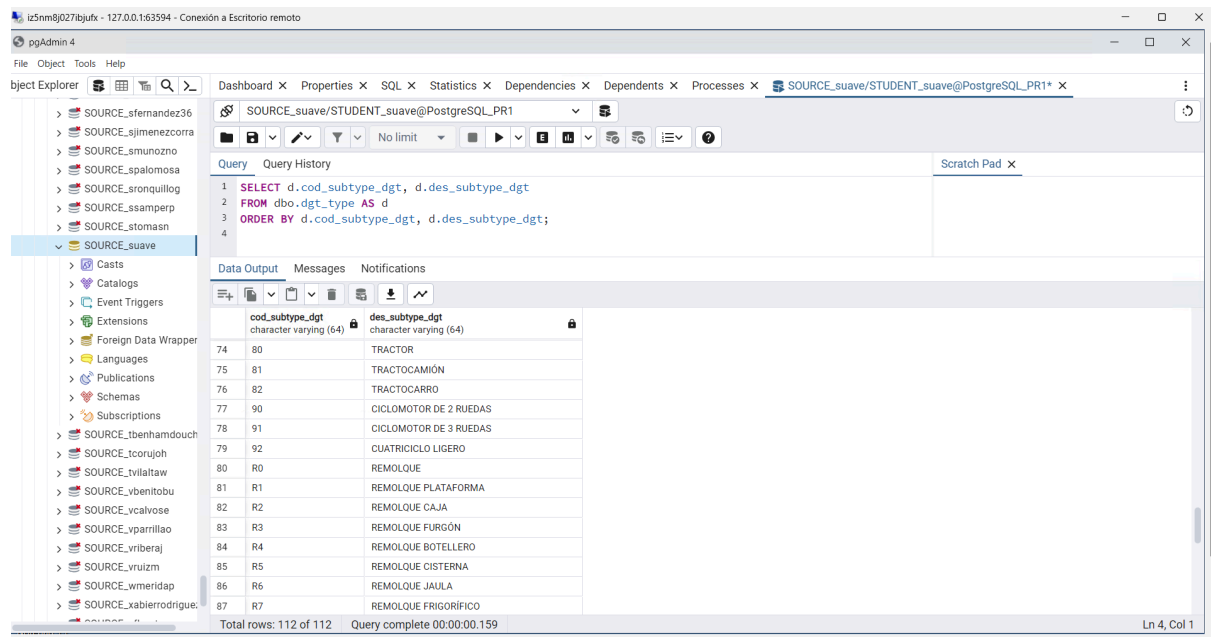
La condición sobre v.new\_used se aplica en el ON, manteniendo los municipios sin vehículos nuevos (aparecen con COUNT=0). En la versión errónea, el WHERE v.new\_used='N' elimina las filas nulas del LEFT, comportándose como INNER.

## Conclusión breve

Con la práctica se recorre el ciclo completo de diseño relacional. En el Ejercicio 1, he identificado y corregido errores del modelo conceptual (identificación unívoca, cardinalidades, dominios) y he actualizado el E/R para cubrir multipropiedad (OWNER-OWNERSHIP-VEHICLE) y sanciones con conductor opcional (FINE y DRIVER). En el Ejercicio 2, se ha traducido al modelo lógico con PK/AK/FK coherentes —incluida la PK compuesta de DGT\_TYPE (cod\_type\_dgt, cod\_subtype\_dgt) y la FK (type, subtype) en VEHICLE. En el Ejercicio 3, he verificado el modelo físico en PostgreSQL: creación del esquema, ejecución de scripts y validaciones por conteo (dgt\_type = 112, municipalities = 8

132, vehicles = 1 721 704). Con las consultas 1–10 se demuestra selección, filtrado, ordenación, JOIN, agregación y HAVING, y corrigiendo las sentencias típicas (JOIN implícito, precedencia AND/OR, LEFT JOIN anulado) se justifica la semántica correcta. El resultado es un diseño coherente y reproducible, con integridad y correcta interpretación de los datos.

## Apéndice



pgAdmin 4 - iz5nm8j027bjufx - 127.0.0.1:63594 - Conexión a Escritorio remoto

File Object Tools Help

Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1\*

Query Query History Scratch Pad x

```

1 SELECT d.cod_subtype_dgt, d.des_subtype_dgt
2 FROM dbo.dgt_type AS d
3 ORDER BY d.cod_subtype_dgt, d.des_subtype_dgt;
4

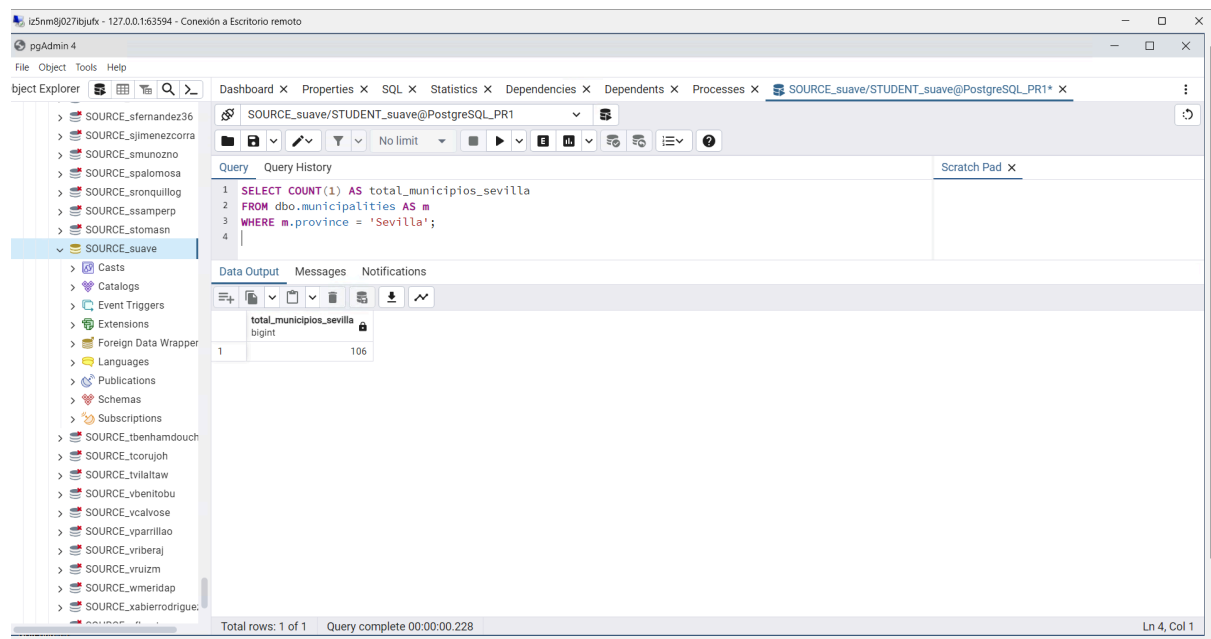
```

Data Output Messages Notifications

	cod_subtype_dgt	des_subtype_dgt
	character varying (64)	character varying (64)
74	80	TRACTOR
75	81	TRACTOCAMIÓN
76	82	TRACTOCARRO
77	90	CICLOMOTOR DE 2 RUEDAS
78	91	CICLOMOTOR DE 3 RUEDAS
79	92	CUATRICICLO LIGERO
80	R0	REMOLQUE
81	R1	REMOLQUE PLATAFORMA
82	R2	REMOLQUE CAJA
83	R3	REMOLQUE FURGÓN
84	R4	REMOLQUE BOTELLERO
85	R5	REMOLQUE CISTERNA
86	R6	REMOLQUE JAULA
87	R7	REMOLQUE FRIGORÍFICO

Total rows: 112 of 112 Query complete 00:00:00.159 Ln 4, Col 1

Captura Q1 — E7\_Q1\_dgt\_subtypes.png



pgAdmin 4 - iz5nm8j027bjufx - 127.0.0.1:63594 - Conexión a Escritorio remoto

File Object Tools Help

Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1\*

Query Query History Scratch Pad x

```

1 SELECT COUNT(1) AS total_municipios_sevilla
2 FROM dbo.municipalities AS m
3 WHERE m.province = 'Sevilla';
4

```

Data Output Messages Notifications

	total_municipios_sevilla
	bigint
1	106

Total rows: 1 of 1 Query complete 00:00:00.228 Ln 4, Col 1

Captura Q2 — E8\_Q2\_count\_sevilla.png

pgAdmin 4 - 127.0.0.1:53594 - Conexión a Escritorio remoto

Dashboard | Properties | SQL | Statistics | Dependencies | Dependents | Processes | SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1\*

Object Explorer: SOURCE\_suave

Query: SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1

```

2 FROM dbo.municipalities AS m
3 WHERE m.province = 'Sevilla'
4 AND (m.name_muni ILIKE 'A%' OR m.name_muni ILIKE 'U%')
5 ORDER BY m.name_muni DESC;
6

```

Data Output: Messages | Notifications

	name_muni	province
1	Utrera	Sevilla
2	Umbrete	Sevilla
3	Aznalcóllar	Sevilla
4	Aznalcázar	Sevilla
5	Arahal	Sevilla
6	Almensilla	Sevilla
7	Almadén de la Plata	Sevilla
8	Algámitas	Sevilla
9	Alcolea del Río	Sevilla
10	Alcalá del Río	Sevilla
11	Alcalá de Guadaira	Sevilla
12	Albaida del Aljarafe	Sevilla
13	Alanís	Sevilla
14	Aguadulce	Sevilla

Total rows: 14 of 14 | Query complete 00:00:00.212 | Ln 6, Col 1

Captura Q3 — E9\_Q3\_sevilla\_prefijos.png

pgAdmin 4 - 127.0.0.1:53594 - Conexión a Escritorio remoto

Dashboard | Properties | SQL | Statistics | Dependencies | Dependents | Processes | SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1\*

Object Explorer: SOURCE\_suave

Query: SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1

```

1 BEGIN;
2
3 INSERT INTO dbo.dgt_type (cod_type_dgt, des_type_dgt, cod_subtype_dgt, des_subtype_dgt)
4 VALUES ('4', 'TURISMOS', 'Z9', 'TURISMO PRUEBA PR2');
5
6 -- Verificación: debe aparecer 1 fila con el subtipo Z9
7 SELECT cod_type_dgt, des_type_dgt, cod_subtype_dgt, des_subtype_dgt
8 FROM dbo.dgt_type
9 WHERE cod_subtype_dgt = 'Z9';
10

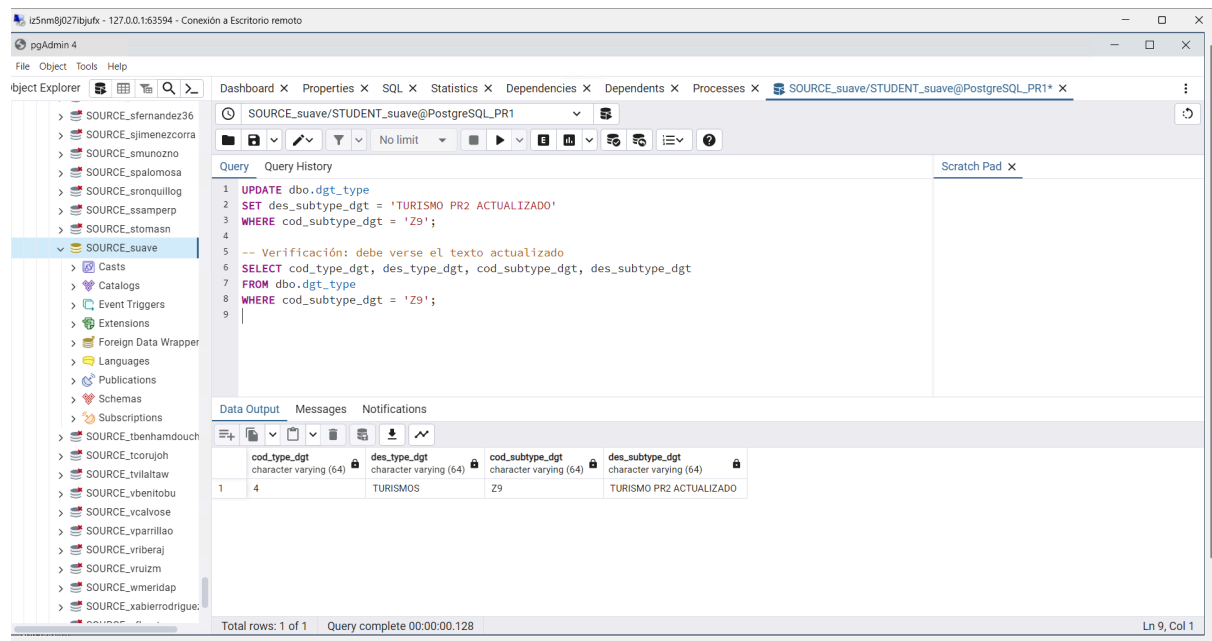
```

Data Output: Messages | Notifications

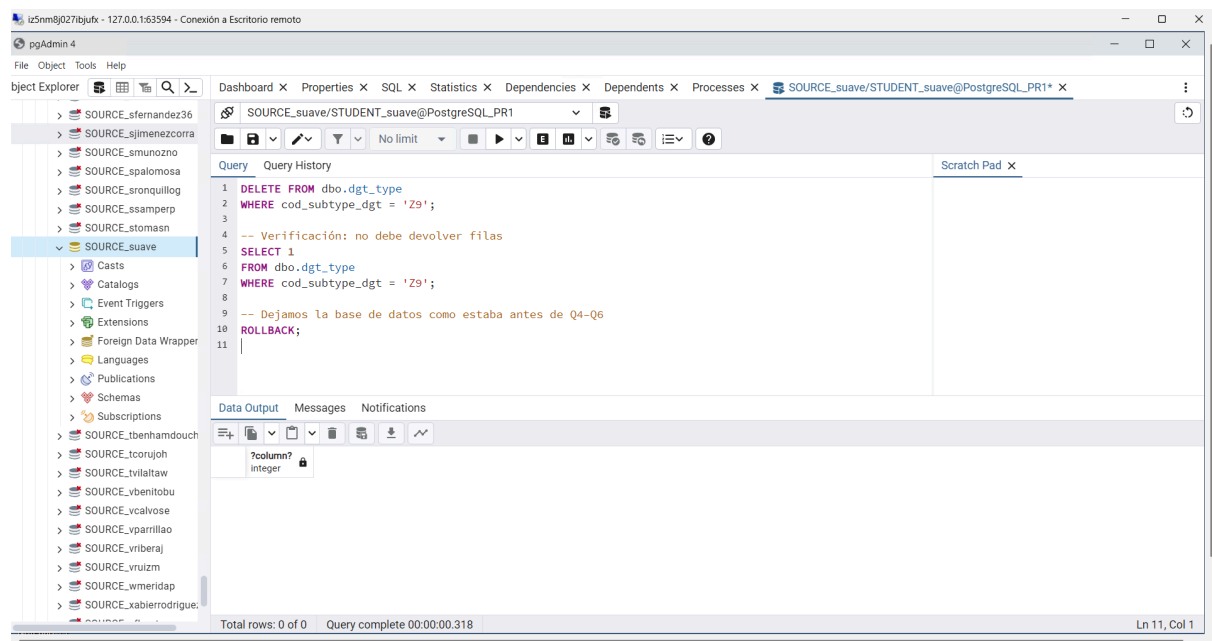
	cod_type_dgt	des_type_dgt	cod_subtype_dgt	des_subtype_dgt
1	4	TURISMOS	Z9	TURISMO PRUEBA PR2

Total rows: 1 of 1 | Query complete 00:00:00.166 | Ln 10, Col 1

Captura Q4 — E10\_Q4\_insert\_verify.png



Captura Q5 — E11\_Q5\_update\_verify.png



Captura Q6 — E12\_Q6\_delete\_verify.png

pgAdmin 4

Object Explorer

- SOURCE\_sfernandez36
- SOURCE\_sjimenezcorra
- SOURCE\_smunozno
- SOURCE\_spalomosa
- SOURCE\_sronquillo
- SOURCE\_ssamperp
- SOURCE\_stomasn
- SOURCE\_suave**
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrapper
  - Languages
  - Publications
  - Schemas
  - Subscriptions
  - SOURCE\_tbenhamdouch
  - SOURCE\_tcorujoh
  - SOURCE\_tvillataw
  - SOURCE\_vbenitobu
  - SOURCE\_vcalvose
  - SOURCE\_vparillao
  - SOURCE\_vriberaj
  - SOURCE\_vruizm
  - SOURCE\_vmeridap
  - SOURCE\_xabierrodrigue

Query Editor: SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1

```

1 SELECT
2   v.brand,
3   v.model,
4   m.name_muni AS municipality,
5   m.altitude,
6   d.des_type_dgt AS dgt_type
7 FROM dbo.vehicles AS v
8 JOIN dbo.municipalities AS m ON v.municipality = m.cod_geo
9 JOIN dbo.dgt_type AS d ON v.subtype = d.cod_subtype_dgt
10 WHERE d.des_type_dgt = 'TURISMOS'
11 AND m.altitude > 950
12 ORDER BY v.brand ASC, v.model ASC;

```

Data Output

	brand	model	municipality	altitude	dgt_type
1	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
2	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
3	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
4	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
5	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
6	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS
7	AUDI	A1 SPORTBACK	Castell de l'Areny	954	TURISMOS

Total rows: 1000 of 26828 Query complete 00:00:00.544 Ln 13, Col 1

Captura Q7 — E13\_Q7\_turismos\_alt950.png

pgAdmin 4

Object Explorer

- SOURCE\_sfernandez36
- SOURCE\_sjimenezcorra
- SOURCE\_smunozno
- SOURCE\_spalomosa
- SOURCE\_sronquillo
- SOURCE\_ssamperp
- SOURCE\_stomasn
- SOURCE\_suave**
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrapper
  - Languages
  - Publications
  - Schemas
  - Subscriptions
  - SOURCE\_tbenhamdouch
  - SOURCE\_tcorujoh
  - SOURCE\_tvillataw
  - SOURCE\_vbenitobu
  - SOURCE\_vcalvose
  - SOURCE\_vparillao
  - SOURCE\_vriberaj
  - SOURCE\_vruizm
  - SOURCE\_vmeridap
  - SOURCE\_xabierrodrigue

Query Editor: SOURCE\_suave/STUDENT\_suave@PostgreSQL\_PR1

```

1 SELECT
2   m.name_muni AS municipality,
3   COUNT(*) AS total_vehicles
4 FROM dbo.vehicles AS v
5 JOIN dbo.municipalities AS m ON v.municipality = m.cod_geo
6 GROUP BY m.name_muni
7 ORDER BY m.name_muni ASC;

```

Data Output

	municipality	total_vehicles
1	Abrera	15770
2	Aguilar de Segarra	19456
3	Alella	16739
4	Avià	28025
5	Badia del Vallès	24529
6	Balençà	40394
7	Balsareny	25327
8	Barcelona	22458
9	Caldes de Montbui	102410
10	Calella	59698

Total rows: 27 of 27 Query complete 00:00:00.707 Ln 8, Col 1

Captura Q8 — E14\_Q8\_count\_por\_municipio.png

pgAdmin 4

Object Explorer

- SOURCE\_sfernandez36
- SOURCE\_sjimenezcorra
- SOURCE\_smunozno
- SOURCE\_spalomosa
- SOURCE\_sronquillo
- SOURCE\_ssamperp
- SOURCE\_stomasn
- SOURCE\_suave**
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrapper
  - Languages
  - Publications
  - Schemas
  - Subscriptions
- SOURCE\_tbenhamdouch
- SOURCE\_tcorujoh
- SOURCE\_tvallataw
- SOURCE\_ybentobu
- SOURCE\_ycalvose
- SOURCE\_yparillao
- SOURCE\_yvriberaj
- SOURCE\_yruizm
- SOURCE\_ywmeridap
- SOURCE\_xabierrodrigue

Query

```

1 SELECT
2   m.name_muni AS municipality,
3   COUNT(*) AS total_vehicles
4 FROM dbo.vehicles AS v
5 JOIN dbo.municipalities AS m ON v.municipality = m.cod_geo
6 GROUP BY m.name_muni
7 HAVING COUNT(*) < 15000
8 ORDER BY m.name_muni ASC;

```

Data Output

	municipality character varying (50)	total_vehicles bigint
1	Calonge de Segarra	13908
2	Fogars de Montclús	14915

Total rows: 2 of 2 Query complete 00:00:00.716 Ln 9, Col 1

Captura Q9 — E15\_Q9\_municipios\_lt\_15000.png

pgAdmin 4

Object Explorer

- SOURCE\_sfernandez36
- SOURCE\_sjimenezcorra
- SOURCE\_smunozno
- SOURCE\_spalomosa
- SOURCE\_sronquillo
- SOURCE\_ssamperp
- SOURCE\_stomasn
- SOURCE\_suave**
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrapper
  - Languages
  - Publications
  - Schemas
  - Subscriptions
- SOURCE\_tbenhamdouch
- SOURCE\_tcorujoh
- SOURCE\_tvallataw
- SOURCE\_ybentobu
- SOURCE\_ycalvose
- SOURCE\_yparillao
- SOURCE\_yvriberaj
- SOURCE\_yruizm
- SOURCE\_ywmeridap
- SOURCE\_xabierrodrigue

Query

```

1 SELECT
2   d.des_type_dgt AS dgt_type,
3   v.new_used AS new_or_used, -- 'N' nuevo, 'U' usado
4   COUNT(*) AS total_vehicles
5 FROM dbo.vehicles AS v
6 JOIN dbo.dgt_type AS d
7   ON v.subtype = d.cod_subtype_dgt
8 GROUP BY d.des_type_dgt, v.new_used
9 ORDER BY d.des_type_dgt ASC, v.new_used ASC;

```

Data Output

	dgt_type character varying (64)	new_or_used character varying (1)	total_vehicles bigint
1	AUTOBUSES	N	4161
2	AUTOBUSES	U	247
3	CAMIONES	N	30951
4	CAMIONES	U	456
5	CICLOMOTORES	N	7828
6	FURGONETAS	N	61598
7	FURGONETAS	U	2717
8	MOTOCICLETAS	N	215384
9	MOTOCICLETAS	U	342
10	OTROS vehiculos	N	11305

Total rows: 17 of 17 Query complete 00:00:00.843 Ln 10, Col 1

Captura Q10 — E16\_Q10\_tipo\_y\_nuevo\_usado.png