Desarrollo de Aplicaciones con Bases de Datos

Licenciatura en Informática

Trabajo Práctico 4

**Prof. Titular Disciplinar: Silvia Laura Castelli**

**Prof. Titular Experto: Ricardo Ramón Daubrowsky**

**Alumno: Pablo Alejandro Hamann**

**Legajo: VINF010782**

**Año: 2025**



Tabla de contenido

[Introducción 1](#_Toc1)

[Repositorio en GitHub 1](#_Toc2)

[Consigna 1](#_Toc3)

[Actividad 1: Crear una base de datos denominada DWPedidos 1](#_Toc4)

[Actividad 2: Crear las siguientes tablas en la base de datos DWPedidos 2](#_Toc5)

[Actividad 3: Cargar de las tablas… 2](#_Toc6)

[Sección 0: Tareas previas a la creación del esquema y configuración del entorno 2](#_Toc7)

[Borrar esquema (usuario) si existiera 2](#_Toc8)

[Crear el esquema y darle privilegios 3](#_Toc9)

[Definición de funciones personalizadas 3](#_Toc10)

[Asignación de permisos 4](#_Toc11)

[Sección 1: Sentencias de creación de la base de datos DWPedidos 4](#_Toc12)

[Modelado dimensional y el modelo estrella 4](#_Toc13)

[Dimensión de Fechas 4](#_Toc14)

[Dimensión de Productos 5](#_Toc15)

[Dimensión de Clientes 5](#_Toc16)

[Tabla de hechos de Pedidos 5](#_Toc17)

[Sección 2: Detalle de proced. almacenados p/carga de datos en DWPedidos (Proceso ETL) 5](#_Toc18)

[Procedimiento para cargar DIMFechas 6](#_Toc19)

[Procedimiento para cargar DIMClientes 6](#_Toc20)

[Procedimiento para cargar DIMProductos 7](#_Toc21)

[Procedimiento para cargar FACTPedidos 7](#_Toc22)

[Procedimiento ejecutar\_ETLMaestroDW 8](#_Toc23)

[Ejecución del procedimiento ETL Maestro 8](#_Toc24)

[Sección 3: Extra 8](#_Toc25)

[Captura de pantalla primera ejecución de sentencias SQL 8](#_Toc26)

[Captura de pantalla segunda ejecución de sentencias SQL 9](#_Toc27)

[Captura de pantalla de los datos en DWPEDIDOS.DIMClientes 10](#_Toc28)

[Captura de pantalla de los datos en DWPEDIDOS.DIMFechas 10](#_Toc29)

[Captura de pantalla de los datos en DWPEDIDOS.DIMProductos 10](#_Toc30)

[Captura de pantalla de los datos en DWPEDIDOS.FACTPedidos 10](#_Toc31)

# Introducción

Este documento corresponde al desarrollo de las consignas planteadas en el Trabajo Práctico 4, y retoma a partir de las tareas realizadas en todos los TPs anteriores.

## Repositorio en GitHub

Todo lo producido, tanto para este presente TP, com para el anterior, se encuentra en un repositorio en GitHub creado para el cursado de esta materia. Allí, se mantienen actualizadas tanto las actividades prácticas como los TPs y cualquier otro tipo de actividad que implique desarrollo (de documentación, programación, etc.), que se dé durante el cursado de la materia. El repositorio se puede acceder mediante el siguiente enlace:

<https://github.com/linkstat/dabd/tree/main>

Archivos principales del proyecto ***dabd*** (este proyecto):

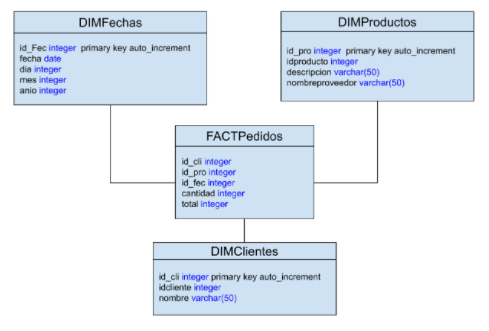
* Este documento en formato PDF:
  + <https://github.com/linkstat/dabd/raw/refs/heads/main/docs/HAMANN-PABLO-ALEJANDRO-TP4.pdf>
* Este documento en formato DOCX de Word:
  + <https://github.com/linkstat/dabd/raw/refs/heads/main/docs/HAMANN-PABLO-ALEJANDRO-TP4.docx>
* Archivo de script SQL que debe ejecutar el usuario SYSTEM (DBO):
  + [https://raw.githubusercontent.com/linkstat/dabd/refs/heads/main/sql/HAMANN-PABLO-ALEJANDRO-TP4-[SYSTEM].sql](o https://raw.githubusercontent.com/linkstat/dabd/refs/heads/main/sql/HAMANN-PABLO-ALEJANDRO-TP4-[SYSTEM].sql)
* Archivo de script SQL que debe ejecutar el usuario DWPEDIDOS (DW):
  + [https://raw.githubusercontent.com/linkstat/dabd/refs/heads/main/sql/HAMANN-PABLO-ALEJANDRO-TP4-[DWPEDIDOS].sql](o https://raw.githubusercontent.com/linkstat/dabd/refs/heads/main/sql/HAMANN-PABLO-ALEJANDRO-TP4-[DWPEDIDOS].sql)

# Consigna

El objetivo de este trabajo individual consiste en desarrollar un proceso ETL (extracción, transformación y carga) utilizando lenguaje SQL, para completar la base de datos del datawarehouse de la organización.

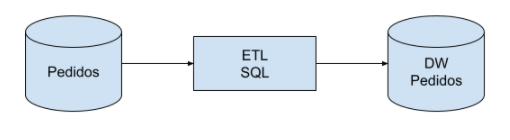
## Actividad 1: Crear una base de datos denominada DWPedidos

## Actividad 2: Crear las siguientes tablas en la base de datos DWPedidos



## Actividad 3: Cargar de las tablas…

Cargar las tablas de la base de datos DWPedidos con los datos de la base de datos de PEDIDOS, mediante procedimientos almacenados desarrollados en SQL



# Sección 0: Tareas previas a la creación del esquema y configuración del entorno

## Borrar esquema (usuario) si existiera

|  |
| --- |
| BEGIN  EXECUTE IMMEDIATE 'DROP USER DWPEDIDOS CASCADE';  EXCEPTION  WHEN OTHERS THEN  IF SQLCODE != -1918 THEN RAISE; END IF; -- ORA-01918: usuario no existe  END;  / |

## Crear el esquema y darle privilegios

|  |
| --- |
| BEGIN  EXECUTE IMMEDIATE q'[  CREATE USER DWPEDIDOS IDENTIFIED BY dabdTP4  DEFAULT TABLESPACE USERS  TEMPORARY TABLESPACE TEMP  QUOTA UNLIMITED ON USERS  ]';  EXECUTE IMMEDIATE 'GRANT CONNECT, RESOURCE TO DWPEDIDOS';  END;  / |

## Definición de funciones personalizadas

|  |
| --- |
| -- Función para convertir (y almacenar) UUID en RAW(16)  CREATE OR REPLACE FUNCTION uuid\_to\_raw(p\_uuid IN VARCHAR2)  RETURN RAW DETERMINISTIC AS  BEGIN  RETURN HEXTORAW(REPLACE(p\_uuid,'-',''));  END;  /    -- Función para recuperar y convertir de nuevo a UUID  CREATE OR REPLACE FUNCTION raw\_to\_uuid(p\_raw IN RAW)  RETURN VARCHAR2 DETERMINISTIC AS  v\_hex VARCHAR2(32) := RAWTOHEX(p\_raw);  BEGIN  RETURN LOWER(  SUBSTR(v\_hex,1,8)||'-'||  SUBSTR(v\_hex,9,4)||'-'||  SUBSTR(v\_hex,13,4)||'-'||  SUBSTR(v\_hex,17,4)||'-'||  SUBSTR(v\_hex,21,12)  );  END;  / |

## Asignación de permisos

Para que el usuario DWPEDIDOS consulte datos de las tablas del esquema PEDIDOS, necesitamos:

1. Permisos de SELECT en las tablas de PEDIDOS
2. Permisos de ejecución de funciones.
3. Permisos de creación de sinónimos para comodidad (así no escribimos PEDIDOS.<tabla> siempre)

|  |
| --- |
| -- A. Dar permisos de SELECT:  GRANT SELECT ON PEDIDOS.Clientes TO DWPEDIDOS;  GRANT SELECT ON PEDIDOS.Proveedores TO DWPEDIDOS;  GRANT SELECT ON PEDIDOS.Productos TO DWPEDIDOS;  GRANT SELECT ON PEDIDOS.Pedidos TO DWPEDIDOS;  GRANT SELECT ON PEDIDOS.DetallePedidos TO DWPEDIDOS;    -- B. Dar permisos de EXECUTE:  GRANT EXECUTE ON uuid\_to\_raw TO DWPEDIDOS;  GRANT EXECUTE ON raw\_to\_uuid TO DWPEDIDOS;    -- C. Dar permiso para crear sinónimos  GRANT CREATE SYNONYM TO DWPEDIDOS; |

# Sección 1: Sentencias de creación de la base de datos DWPedidos

Sentencias de creación de las tablas del DW (modelo estrella, PK RAW(16)/UUID).

## Modelado dimensional y el modelo estrella

En los data warehouses, **no todas las tablas son iguales**:

* **Tablas de hechos:**
  + Registran los eventos principales del negocio, con métricas cuantificables.
  + Ejemplo: ventas, pedidos, transacciones.
  + Suele tener muchas filas, y claves foráneas a dimensiones.
* **Tablas de dimensiones:**
  + Describen los contextos de los hechos, o los “ejes” de análisis.
  + Ejemplo: productos, clientes, fechas, sucursales, empleados.
  + Suele tener pocos registros y campos descriptivos.

En el **modelo estrella**, la tabla central es la **tabla de hechos** y las otras son **dimensiones** que se conectan por claves foráneas.

## Dimensión de Fechas

|  |
| --- |
| -- Dimensión de Fechas  CREATE TABLE DIMFechas (  id\_Fec RAW(16) PRIMARY KEY,  fecha DATE,  dia NUMBER,  mes NUMBER,  anio NUMBER  ); |

## Dimensión de Productos

|  |
| --- |
| -- Dimensión de Productos  CREATE TABLE DIMProductos (  id\_pro RAW(16) PRIMARY KEY,  idproducto RAW(16),  descripcion VARCHAR2(255),  nombreproveedor VARCHAR2(100)  ); |

## Dimensión de Clientes

|  |
| --- |
| -- Dimensión de Clientes  CREATE TABLE DIMClientes (  id\_cli RAW(16) PRIMARY KEY,  idcliente RAW(16),  nombre VARCHAR2(255)  ); |

## Tabla de hechos de Pedidos

|  |
| --- |
| -- Tabla de hechos de Pedidos (FACTORIAL)  CREATE TABLE FACTPedidos (  id\_cli RAW(16),  id\_pro RAW(16),  id\_fec RAW(16),  cantidad NUMBER,  total NUMBER(10,2),  FOREIGN KEY (id\_cli) REFERENCES dim\_clientes(id\_cli),  FOREIGN KEY (id\_pro) REFERENCES dim\_productos(id\_pro),  FOREIGN KEY (id\_fec) REFERENCES dim\_fechas(id\_fec)  ); |

# Sección 2: Detalle de proced. almacenados p/carga de datos en DWPedidos (Proceso ETL)

Conviene encapsular cada carga en un procedimiento almacenado.

Aquí la idea es que en vez de ejecutar directamente el INSERT INTO ... SELECT ..., se desarrolle un procedimiento almacenado (por ejemplo, cargar\_FACTPedidos).

Así, cada vez que se quiera refrescar la tabla, solo ejecutaremos el procedimiento y listo.

Esto se aplica tanto a la **tabla de hechos** como a cada **dimensión**.

## Procedimiento para cargar DIMFechas

|  |
| --- |
| -- A. Procedimiento para cargar DIMFechas  CREATE OR REPLACE PROCEDURE cargar\_DIMFechas IS  BEGIN  -- Primero eliminamos datos existentes, para evitar duplicados en recargas  DELETE FROM DIMFechas;    -- Cargamos fechas únicas desde Pedidos  INSERT INTO DIMFechas (id\_fec, fecha, dia, mes, anio)  SELECT  SYS\_GUID(),  fecha,  EXTRACT(DAY FROM fecha),  EXTRACT(MONTH FROM fecha),  EXTRACT(YEAR FROM fecha)  FROM (SELECT DISTINCT fecha FROM PEDIDOS.Pedidos);    COMMIT;  END cargar\_DIMFechas;  / |

## Procedimiento para cargar DIMClientes

|  |
| --- |
| -- B. Procedimiento para cargar DIMClientes  CREATE OR REPLACE PROCEDURE cargar\_DIMClientes IS  BEGIN  DELETE FROM DIMClientes;    INSERT INTO DIMClientes (id\_cli, idcliente, nombre)  SELECT  SYS\_GUID(),  idcliente,  apellido || ', ' || nombres  FROM PEDIDOS.Clientes;    COMMIT;  END cargar\_DIMClientes;  / |

## Procedimiento para cargar DIMProductos

|  |
| --- |
| -- C. Procedimiento para cargar DIMProductos  CREATE OR REPLACE PROCEDURE cargar\_DIMProductos IS  BEGIN  DELETE FROM DIMProductos;    INSERT INTO DIMProductos (id\_pro, idproducto, descripcion, nombreproveedor)  SELECT  SYS\_GUID(),  p.idproducto,  p.descripcion,  prov.nombreproveedor  FROM PEDIDOS.Productos p  JOIN PEDIDOS.Proveedores prov ON p.idproveedor = prov.idproveedor;    COMMIT;  END cargar\_DIMProductos;  / |

## Procedimiento para cargar FACTPedidos

|  |
| --- |
| -- D. Procedimiento para cargar FACTPedidos  CREATE OR REPLACE PROCEDURE cargar\_FACTPedidos IS  BEGIN  DELETE FROM FACTPedidos;    INSERT INTO FACTPedidos (id\_cli, id\_pro, id\_fec, cantidad, total)  SELECT  dc.id\_cli,  dp.id\_pro,  df.id\_fec,  det.cantidad,  det.cantidad \* det.preciounitario  FROM  PEDIDOS.DetallePedidos det  JOIN PEDIDOS.Pedidos ped ON det.numeropedido = ped.numeropedido  JOIN DIMFechas df ON ped.fecha = df.fecha  JOIN DIMClientes dc ON ped.idcliente = dc.idcliente  JOIN DIMProductos dp ON det.idproducto = dp.idproducto;    COMMIT;  END cargar\_FACTPedidos;  / |

## Procedimiento ejecutar\_ETLMaestroDW

Es un procedimiento que invoca a todos los anteriores, en orden.

|  |
| --- |
| -- Procedimiento para cargar todos los procedimientos ordenadamente  CREATE OR REPLACE PROCEDURE ejecutar\_ETLMaestroDW IS  BEGIN  cargar\_DIMFechas;  cargar\_DIMClientes;  cargar\_DIMProductos;  cargar\_FACTPedidos;  END;  / |

## Ejecución del procedimiento ETL Maestro

|  |
| --- |
| -- Ejecutar el ETL Maestro  EXEC ejecutar\_ETLMaestroDW; |

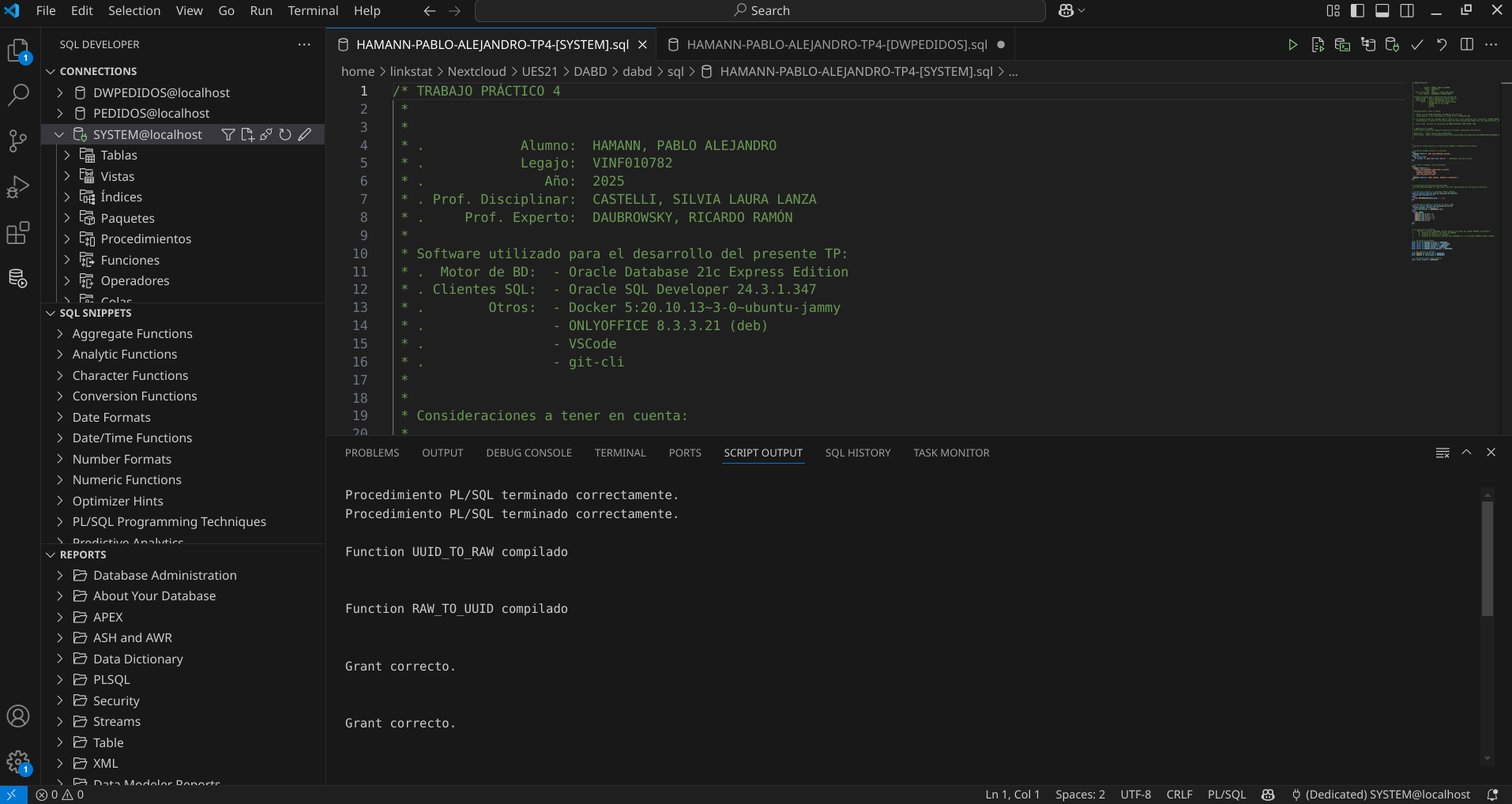
# Sección 3: Extra

Usando la instancia Oracle XE (instalada usando docker, como se explicó en el TP2), ahora configuré la extensión de Oracle SQL Developer en Visual Studio Code, lo que convirtió al editor de texto en una interfaz moderna, amigable y cómoda para trabajar sobre el motor de base de datos de Oracle.

Primero, me conecté como usuario SYSTEM (DBO), probé los comandos que guardé en el primer archivo de scipt (HAMANN-PABLO-ALEJANDRO-TP4-[SYSTEM].sql); allí verificamos que la ejecución de las sentencias fue exitosa.

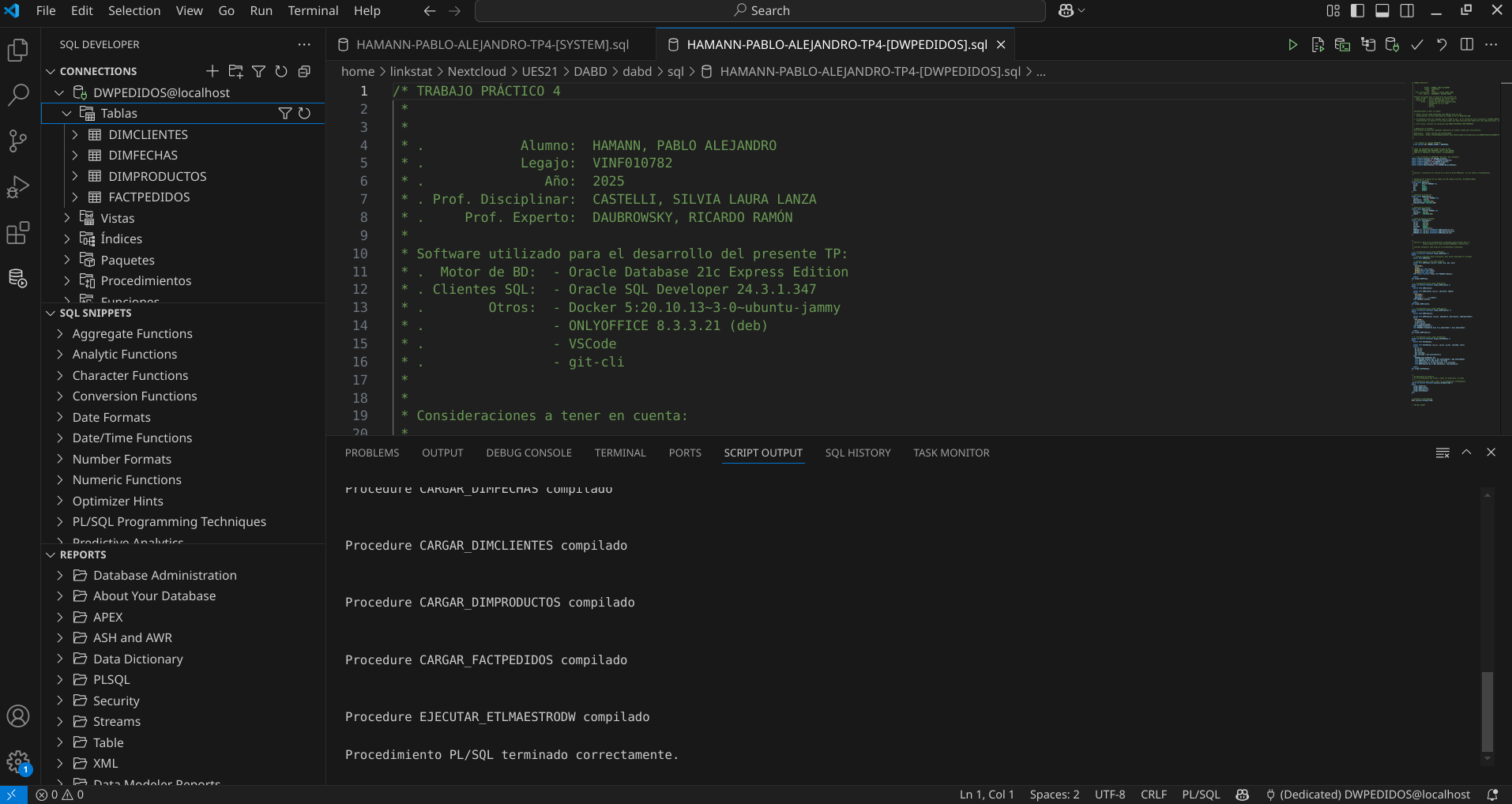
## Captura de pantalla primera ejecución de sentencias SQL

Ejecutadas como usuario SYSTEM@localhost:

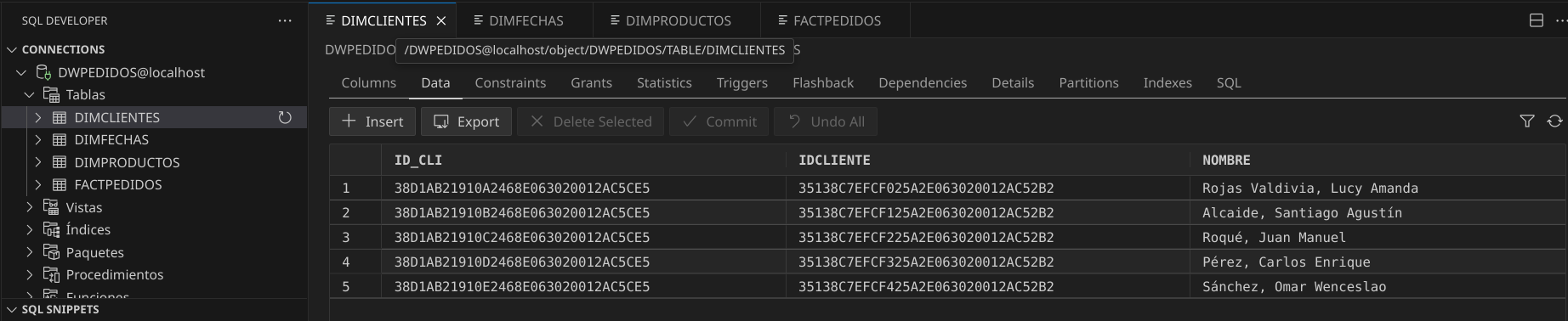


## Captura de pantalla segunda ejecución de sentencias SQL

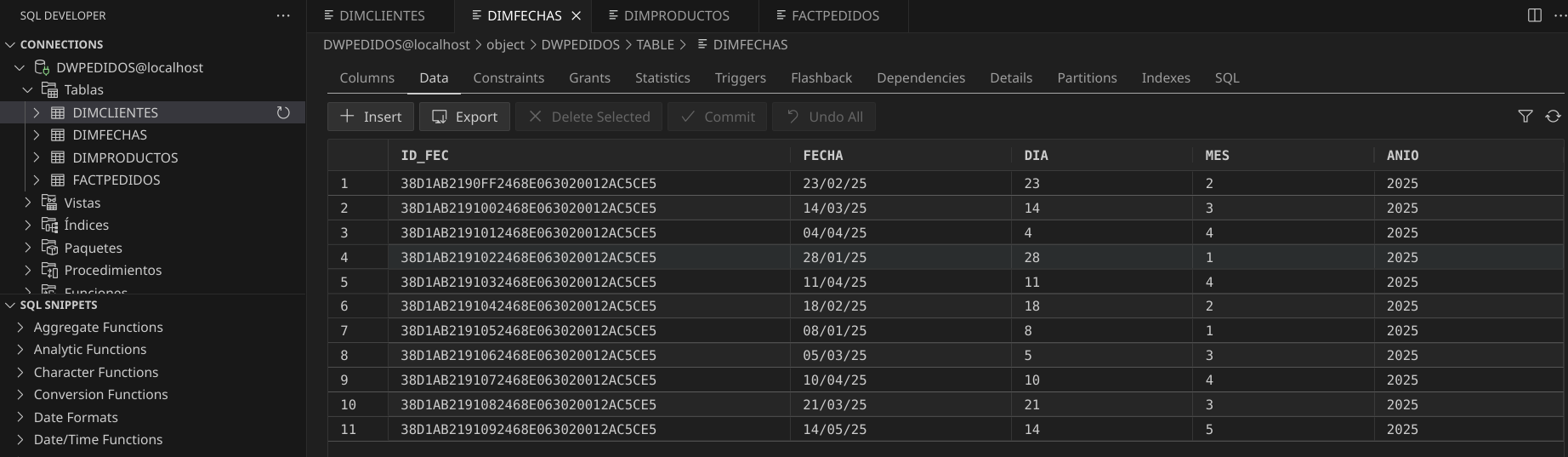
Ejecutadas como usuario DWPEDIDOS@localhost:



## Captura de pantalla de los datos en DWPEDIDOS.DIMClientes



## Captura de pantalla de los datos en DWPEDIDOS.DIMFechas



## Captura de pantalla de los datos en DWPEDIDOS.DIMProductos



## Captura de pantalla de los datos en DWPEDIDOS.FACTPedidos

