Desarrollo de Aplicaciones con Bases de Datos

Licenciatura en Informática

Trabajo Práctico 2

**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](#_Toc197977771)

[Repositorio en GitHub 1](#_Toc197977772)

[Tareas previas para preparar el entorno de trabajo 1](#_Toc197977773)

[Creación de un container docker para Oracle Server en Linux 1](#_Toc197977774)

[Instalación del cliente Oracle SQL Developer en Linux Debian y dervados 5](#_Toc197977775)

[Sentencias de creación de la estructura de la base de datos 5](#_Toc197977776)

[Sentencias DDL para la creación del esquema y objetos (tablas, secuencias, funciones) 6](#_Toc197977777)

[Creación de Tablas 7](#_Toc197977778)

[Creación de *Triggers* 7](#_Toc197977779)

[Conjunto de sentencias SQL para poblar la base de datos 8](#_Toc197977780)

[Sentencias DML para la inserción de datos iniciales en el esquema dentro de un bloque PL/SQL 8](#_Toc197977781)

[Consignas específicas del TP2 13](#_Toc197977782)

[1. Bloque PL/SQL para realización de un pedido 13](#_Toc197977783)

[Resultado de la consulta 1 17](#_Toc197977784)

[Solicitud del ID de cliente: 18](#_Toc197977785)

[Solicitud del ID de Vendedor 18](#_Toc197977786)

[Solicitud de ID de 1er Producto 18](#_Toc197977787)

[Solicitud de la cantidad de unidades del 1er producto 19](#_Toc197977788)

[Solicitud del ID de 2do producto (opcional) 19](#_Toc197977789)

[Cantidad de la cantidad de unidades del 2do producto 19](#_Toc197977790)

[Solicitud de ID del 3er producto 20](#_Toc197977791)

[Solicitud de la cantidad de unidades del 3er producto 20](#_Toc197977792)

[Resultado: 21](#_Toc197977793)

[2. Procedimiento almacenado para anular un pedido confirmado 21](#_Toc197977794)

[Listado de pedidos y sus estados 23](#_Toc197977795)

[Primera ejecución del procedimiento almacenado 23](#_Toc197977796)

[Resultado de la primera ejección del procedimiento almacenado 24](#_Toc197977797)

[Segunda ejecución del procedimiento almacenado 24](#_Toc197977798)

[Resultado de la segunda ejección del procedimiento almacenado 24](#_Toc197977799)

[3. Creación de una tabla log 24](#_Toc197977800)

[4. Creación de un *trigger* para registros en la tabla log 25](#_Toc197977801)

[Prueba de funcionamiento del trigger 25](#_Toc197977802)

[Consulta a la tabla de log 26](#_Toc197977803)

[Resultado de la consulta a la tabla log 26](#_Toc197977804)

[5. Procedimiento almacenado para actualización de precios 26](#_Toc197977805)

[Consulta de artículos nacionales antes de su modificación de precios 27](#_Toc197977806)

[Consulta de artículos nacionales posterior a su modificación de precios 27](#_Toc197977807)

[Consulta de artículos importados antes de su modificación de precios 29](#_Toc197977808)

[Consulta de artículos importados posterior a su modificación de precios 29](#_Toc197977809)

# Introducción

Este documento corresponde al desarrollo de las consignas planteadas en el Trabajo Práctico 2, y retoma a partir de las tareas realizadas en el TP1. Al igual que en el TP anterior, complementan este documento, el archivo SQL correspondiente, que contiene todas las sentencias necesarias para crear la base de datos (esta vez en Oracle), su estructura, poblarla con algunos datos modelo, y realizar las consultas solicitadas.

## 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-TP2.pdf>
* Este documento en formato DOCX de Word:
  + <https://github.com/linkstat/dabd/raw/refs/heads/main/docs/HAMANN-PABLO-ALEJANDRO-TP2.docx>
* Archivo de script SQL:
  + <https://raw.githubusercontent.com/linkstat/dabd/refs/heads/main/sql/HAMANN-PABLO-ALEJANDRO-TP2.sql>

También es posible ver el historial de *commits* realizado a los archivos (y a toda la estructura del proyecto), ya que se trata de un repositorio público y se actualizando de forma regular, sobre todo cuando se aplican muchos cambios.

# Tareas previas para preparar el entorno de trabajo

## Creación de un container docker para Oracle Server en Linux

En lo personal, no tuve éxito con la versión del servidor *Oracle Database XE* para Windows. Aprovechando que dispongo (además) de un sistema Linux, intenté allí. Pero los instaladores de Oracle para Linux no son aptos / compatibles para sistemas basados en Debian/Ubuntu (sistema de paquetería deb). Por lo cual finalmente opté por construir un contenedor docker, a partir de una imagen oficial. Los pasos:

1. Clonación el repositorio oficial de Oracle

~$ git clone <https://github.com/oracle/docker-images.git>

~$ cd docker-images/OracleDatabase/SingleInstance/dockerfiles

1. Descarga el RPM manualmente desde el sitio de Oracle (21c XE para OL8)

~/docker-images/OracleDatabase$ curl -OL https://download.oracle.com/otn-pub/otn\_software/db-express/oracle-database-xe-21c-1.0-1.ol8.x86\_64.rpm?AuthParam=1747014285\_e33ae7cf7f5723d3a4cdc77282ea97ff

1. Construye la imagen (demora bastante)

~/docker-images/OracleDatabase$ ./buildContainerImage.sh -v 21.3.0 -x

1. Crear carpeta (y establecer permisos) para la persistencia del contenedor del servidor Oracle en /opt/

~/docker-images/OracleDatabase$ sudo mkdir -p /opt/oracle/oradata

~/docker-images/OracleDatabase$ sudo chgrp -R docker /opt/oracle/oradata

~/docker-images/OracleDatabase$ sudo chmod -R g+rw /opt/oracle/oradata

1. Establecer una contraseña (mínimo 8 caracteres, al menos una mayuscula, una minúscula y un número)

~/docker-images/OracleDatabase$ export DBXEPass="dabdTP2"

1. Crear contenedor con persistencia y contraseña definida en $OracleXEPass (demora bastantes minutos)

~/docker-images/OracleDatabase$ docker run -d --name oracle-xe \

-p 1521:1521 -p 5500:5500 \

-e ORACLE\_PWD=$DBXEPass \

-e ORACLE\_CHARACTERSET=AL32UTF8 \

-v ~/oracle-data:/opt/oracle/oradata \

oracle/database:21.3.0-xe

1. Verificar logs (para asegurar que el contenedor se construye correctamente)

~/docker-images/OracleDatabase$ docker logs -f oracle-xe

La salida debería ser similar a esta:

Specify a password to be used for database accounts. Oracle recommends that the password entered should be at least 8 characters in length, contain at least 1 uppercase character, 1 lower case character and 1 digit [0-9]. Note that the same password will be used for SYS, SYSTEM and PDBADMIN accounts:

Confirm the password:

Configuring Oracle Listener.

Listener configuration succeeded.

Configuring Oracle Database XE.

Enter SYS user password:

\*\*\*\*\*\*\*\*\*\*\*

Enter SYSTEM user password:

\*\*\*\*\*\*\*\*\*\*\*

Enter PDBADMIN User Password:

\*\*\*\*\*\*\*\*\*\*\*\*\*

Prepare for db operation

7% complete

Copying database files

29% complete

Creating and starting Oracle instance

30% complete

Completing Database Creation

50% complete

Creating Pluggable Databases

71% complete

Executing Post Configuration Actions

93% complete

Running Custom Scripts

100% complete

Database creation complete. For details check the logfiles at:

/opt/oracle/cfgtoollogs/dbca/XE.

Database Information:

Global Database Name:XE

System Identifier(SID):XE

Look at the log file "/opt/oracle/cfgtoollogs/dbca/XE/XE.log" for further details.

Connect to Oracle Database using one of the connect strings:

Pluggable database: 101691f20601/XEPDB1

Multitenant container database: 101691f20601

Use https://localhost:5500/em to access Oracle Enterprise Manager for Oracle Database XE

SQL\*Plus: Release 21.0.0.0.0 - Production on Mon May 12 04:11:39 2025

Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:

Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

SQL>

System altered.

SQL>

Pluggable database altered.

SQL>

PL/SQL procedure successfully completed.

SQL>

User created.

SQL>

Grant succeeded.

SQL>

User altered.

SQL> SQL> Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

SQL\*Plus: Release 21.0.0.0.0 - Production on Mon May 12 04:11:40 2025

Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:

Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

SQL>

PL/SQL procedure successfully completed.

SQL> Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

The Oracle base remains unchanged with value /opt/oracle

The Oracle base remains unchanged with value /opt/oracle

**#########################**

**DATABASE IS READY TO USE!**

**#########################**

The following output is now a tail of the alert.log:

XEPDB1(3):Completed: ALTER DATABASE DEFAULT TABLESPACE "USERS"

2025-05-12T04:11:39.236609+00:00

ALTER PLUGGABLE DATABASE XEPDB1 SAVE STATE

Completed: ALTER PLUGGABLE DATABASE XEPDB1 SAVE STATE

2025-05-12T04:11:39.771304+00:00

ALTER SYSTEM SET control\_files='/opt/oracle/oradata/XE/control01.ctl' SCOPE=SPFILE;

2025-05-12T04:11:39.834437+00:00

ALTER SYSTEM SET local\_listener='' SCOPE=BOTH;

ALTER PLUGGABLE DATABASE XEPDB1 SAVE STATE

Completed: ALTER PLUGGABLE DATABASE XEPDB1 SAVE STATE

Si la salida terminó con el mensaje "DATABASE IS READY TO USE!", podemos abandonar la vista del log (con Ctrl+C). El entorno ya está listo para conectarse desde un cliente SQL (por ejemplo: *Oracle SQL Developer*, u otro). El servidor *Oracle Database XE* (ejecutándose dentro del contenedor) está escuchando en los puertos del host anfitrión gracias a los parámetros: -p 1521:1521 y -p 5500:5500). A partir de ese momento, ya nos podemos conectar al servidor, con estos datos de conexión:

|  |  |
| --- | --- |
| **DATOS DE CONEXIÓN** | |
| **Parámetro** | **Valor** |
| Host | **localhost** / 127.0.0.1 |
| Puerto | **1521** |
| SID | **XE** |
| Servicio | ***XEPDB1*** *(recomendado)* |
| Usuario | ***SYSTEM*** *o cualquier otro* |
| Contraseña | ***dabdTP2*** (o la antes definida) |

**Nota:** se prefiere "Servicio" (XEPDB1) en lugar de "SID" si el cliente lo permite. Es la forma moderna de acceder a la *Pluggable Database*.

1. Al reiniciar el sistema, el contendor docker no autoinicia; por lo cual para iniciar (y/o detener) manualmente el contenedor, disponemos de los comandos:

~$ docker start oracle-xe # inicia

~$ docker stop oracle-xe # detiene

1. Opcionalmente, se puede configurar para que el contenedor con el servidor Oracle inicie automáticamente con el sistema:

~$ docker update --restart=unless-stopped oracle-xe

## Instalación del cliente Oracle SQL Developer en Linux Debian y dervados

Al igual que con el servidor de Oracle Database XE, no hay una versión para Debian y derivados. La opción más conveniente, consiste en 'alienizar' el paquete RPM (de distribuciones *RedHat* y derivadas), a un paquete apt-compatible, del tipo deb. Para esto:

1. Descargar el cliente *Oracle SQL Developer* (en formato RPM):

~$ curl -OL https://download.oracle.com/otn-pub/otn\_software/db-express/oracle-database-xe-21c-1.0-1.ol8.x86\_64.rpm

1. *“Alienizar*“ (convertir) el paquete RPM en un paquete DEB, compatible con *Debian* y derivados (*Ubuntu* / *Mint* / *Zorin OS* / etc.):

~$ sudo alien -c ./oracle-database-xe-21c-1.0-1.ol8.x86\_64.rpm

*(la conversión de RPM a DEB puede llevar bastantes minutos, dependiendo del equipo)*

1. Finalmente, instalar el paquete deb obtenido:

~$ sudo dpkg -i ./oracle-database-xe-21c-1.0-1.ol8.x86\_64.rpm

1. De forma.

# Sentencias de creación de la estructura de la base de datos

La base de datos (que en el TP1 se presentó en formato MySQL / MariaDB), se migró completamente al motor de bases de datos de Oracle XE.

Se siguió la misma lógica que antes, esto es: uso de UUID (almacenado antes como BINARY, ahora como RAW), y funciones para convertir desde UUID hacia RAW y viceversa.

|  |
| --- |
| -- Borrar esquema (usuario) si existiera  BEGIN  EXECUTE IMMEDIATE 'DROP USER PEDIDOS 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 PEDIDOS IDENTIFIED BY dabdTP2  DEFAULT TABLESPACE USERS  TEMPORARY TABLESPACE TEMP  QUOTA UNLIMITED ON USERS  ]';  EXECUTE IMMEDIATE 'GRANT CONNECT, RESOURCE TO PEDIDOS';  END;  /  -- Cambiar al esquema PEDIDOS  ALTER SESSION SET CURRENT\_SCHEMA = PEDIDOS;  /\* Definición de funciones personalizadas  \* De forma análoga a lo antes realizado en MySQL, utilizo UUID almacenado  \* en crudo (raw); para esto, desarrollamos dos funciones de conversión.  \*/  -- 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;  / |

# Sentencias DDL para la creación del esquema y objetos (tablas, secuencias, funciones)

Solo mostraremos parcialmente en este documento y a modo de ejemplo algunas pocas tablas, ya que si bien las sentencias se modificaron para el nuevo motor de base de datos, se desea no ser repetitivo respecto de lo ya realizado en el TP1. Por supuesto, que en el archivo SQL adjunto, se desarrollan todas las sentencias para la totalidad de las tareas solicitadas (tanto del TP1, como del TP2).

## Creación de Tablas

Las tablas son exactamente las mismas que en el TP1 (exceptuando la tabla log solicitada en el punto 4 del TP2, muy similar a la tabla LogAnulaciones que ya tenía, pero con sutiles diferencias, que se ajustaron acorde al enunciado).

Por ejemplo, la creación de la tabla **Pedidos** se ve ahora así:

Misma estructura que la presentada en el TP1, pero reescrita para *Oracle Database XE*:

|  |
| --- |
| CREATE TABLE Pedidos (  idpedido RAW(16) NOT NULL CONSTRAINT pk\_pedidos PRIMARY KEY,  numeropedido NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL,  idcliente RAW(16) NOT NULL,  idvendedor RAW(16) NOT NULL,  fecha DATE NOT NULL,  estado VARCHAR2(15) DEFAULT 'pendiente' NOT NULL,  CONSTRAINT uq\_pedidos\_numeropedido UNIQUE (numeropedido),  CONSTRAINT fk\_pedido\_cliente FOREIGN KEY (idcliente)  REFERENCES Clientes(idcliente) ON DELETE CASCADE,  CONSTRAINT fk\_pedido\_vendedor FOREIGN KEY (idvendedor)  REFERENCES Vendedor(idvendedor)  ); |

## Creación de *Triggers*

Los *trggers* presentados en el TP1, se mantienen y adaptan para que cumplan la misma funcionalidad en el nuevo motor de base de datos. Además, se agregarán los solicitados en el TP2.

Por ejemplo, el trigger **trg\_before\_insert\_detalle** ahora se ve así:

Misma responsabilidad que el presentado en el TP1, pero reescrito para *Oracle Database XE*:

|  |
| --- |
| CREATE OR REPLACE TRIGGER trg\_before\_insert\_detalle  BEFORE INSERT ON DetallePedidos  FOR EACH ROW  DECLARE  v\_stock Productos.stock%TYPE;  v\_precio Productos.preciounitario%TYPE;  v\_desc VARCHAR2(255);  v\_msg VARCHAR2(512);  BEGIN  -- Consultar stock, precio y descripción  SELECT stock, preciounitario, descripcion  INTO v\_stock, v\_precio, v\_desc  FROM Productos  WHERE idproducto = :NEW.idproducto;    -- Si no hay stock suficiente, error con detalle  IF v\_stock < :NEW.cantidad THEN  v\_msg := 'Stock insuficiente para el producto ' || v\_desc ||  '. Stock disponible: ' || v\_stock ||  '. Cantidad requerida: ' || :NEW.cantidad;  RAISE\_APPLICATION\_ERROR(-20001, v\_msg);  END IF;    -- Asignar precio unitario actual al detalle  :NEW.preciounitario := v\_precio;  END;  / |

# Conjunto de sentencias SQL para poblar la base de datos

Dado que desde elcomienzo del desarrollo TP1, decidí definir los ID como UUID almacenados en binario (aora RAW), antes almacenaba losmismosen variables, que luegoreutilizaba. Dadoque en *Oracle*, la ariabl puede ser reutilizada dentro del bloque PL/SQL, decidí poblar la abse de datos, dentor de un gran bloque PL/SQL.

## Sentencias DML para la inserción de datos iniciales en el esquema dentro de un bloque PL/SQL

|  |
| --- |
| DECLARE  -- UUIDs para Clientes  v\_uuid\_cliente1 RAW(16) := SYS\_GUID();  v\_uuid\_cliente2 RAW(16) := SYS\_GUID();  v\_uuid\_cliente3 RAW(16) := SYS\_GUID();  v\_uuid\_cliente4 RAW(16) := SYS\_GUID();  v\_uuid\_cliente5 RAW(16) := SYS\_GUID();  -- UUIDs para Proveedores  v\_uuid\_proveedor1 RAW(16) := SYS\_GUID();  v\_uuid\_proveedor2 RAW(16) := SYS\_GUID();  v\_uuid\_proveedor3 RAW(16) := SYS\_GUID();  -- UUIDs para Productos  v\_uuid\_prod01 RAW(16) := SYS\_GUID();  v\_uuid\_prod02 RAW(16) := SYS\_GUID();  v\_uuid\_prod03 RAW(16) := SYS\_GUID();  v\_uuid\_prod04 RAW(16) := SYS\_GUID();  v\_uuid\_prod05 RAW(16) := SYS\_GUID();  v\_uuid\_prod06 RAW(16) := SYS\_GUID();  v\_uuid\_prod07 RAW(16) := SYS\_GUID();  v\_uuid\_prod08 RAW(16) := SYS\_GUID();  v\_uuid\_prod09 RAW(16) := SYS\_GUID();  v\_uuid\_prod10 RAW(16) := SYS\_GUID();  -- UUIDs para Vendedores  v\_uuid\_vendedor1 RAW(16) := SYS\_GUID();  v\_uuid\_vendedor2 RAW(16) := SYS\_GUID();  v\_uuid\_vendedor3 RAW(16) := SYS\_GUID();  -- UUIDs para Pedidos  v\_uuid\_pedido01 RAW(16) := SYS\_GUID();  v\_uuid\_pedido02 RAW(16) := SYS\_GUID();  v\_uuid\_pedido03 RAW(16) := SYS\_GUID();  v\_uuid\_pedido04 RAW(16) := SYS\_GUID();  v\_uuid\_pedido05 RAW(16) := SYS\_GUID();  v\_uuid\_pedido06 RAW(16) := SYS\_GUID();  v\_uuid\_pedido07 RAW(16) := SYS\_GUID();  v\_uuid\_pedido08 RAW(16) := SYS\_GUID();  v\_uuid\_pedido09 RAW(16) := SYS\_GUID();  v\_uuid\_pedido10 RAW(16) := SYS\_GUID();  -- Variables para capturar numeropedido tras INSERT  v\_numPedido1 NUMBER;  v\_numPedido2 NUMBER;  v\_numPedido3 NUMBER;  v\_numPedido4 NUMBER;  v\_numPedido5 NUMBER;  v\_numPedido6 NUMBER;  v\_numPedido7 NUMBER;  v\_numPedido8 NUMBER;  v\_numPedido9 NUMBER;  v\_numPedido10 NUMBER;  BEGIN  -- Insertar Clientes  INSERT INTO Clientes(idcliente,dni,apellido,nombres,direccion,mail)  VALUES(v\_uuid\_cliente1,'18465781','Rojas Valdivia','Lucy Amanda','Av. Sabatini 3288','lucyamanda23@latinmail.com');  INSERT INTO Clientes(idcliente,dni,apellido,nombres,direccion,mail)  VALUES(v\_uuid\_cliente2,'39512723','Alcaide','Santiago Agustín','Yrigoyen 733 5 C, La Plata, Buenos Aires','santialcaide@mineral.ru');  INSERT INTO Clientes(idcliente,dni,apellido,nombres,direccion,mail)  VALUES(v\_uuid\_cliente3,'22101645','Roqué','Juan Manuel','Avellaneda 935, La Banda, Santiago del Estero','jmroque@yustech.com.ar');  INSERT INTO Clientes(idcliente,dni,apellido,nombres,direccion,mail)  VALUES(v\_uuid\_cliente4,'42013728','Pérez','Carlos Enrique','Bedoya 724, Córdoba, Córdoba','carlitosperez@gmail.com');  INSERT INTO Clientes(idcliente,dni,apellido,nombres,direccion,mail)  VALUES(v\_uuid\_cliente5,'12309421','Sánchez','Omar Wenceslao','Rivadavia, 724 3 C, Rosario, Santa Fe','wen733@mail.ru');  -- Insertar Proveedores  INSERT INTO Proveedores(idproveedor,nombreproveedor,direccion,email)  VALUES(v\_uuid\_proveedor1,'Marolio','Corrientes 2350, Gral. Rodríguez, Buenos Aires','info@marolio.com.ar');  INSERT INTO Proveedores(idproveedor,nombreproveedor,direccion,email)  VALUES(v\_uuid\_proveedor2,'Arcor','Av. Chacabuco 1160, Córdoba, Córdoba','arcor@arcor.com');  INSERT INTO Proveedores(idproveedor,nombreproveedor,direccion,email)  VALUES(v\_uuid\_proveedor3,'Dos Hermanos','Av. Pres. Juan Domingo Perón y Scalabrini Ortiz, Concordia, Entre Ríos','info@doshermanos.com.ar');  -- Insertar Productos  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod01,'Arroz Parboil 1kg Dos Hnos Libre Gluten Sin Tacc',20865.0,1518,5000,500,v\_uuid\_proveedor3,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod02,'Huevo de pascuas Arcor Milk unicornio chocolate 140g',18999.0,12497,15000,0,v\_uuid\_proveedor2,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod03,'Yerba Mate Marolio Con Menta - Bolsa 500g',1487.5,1213,12000,1050,v\_uuid\_proveedor1,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod04,'Turron Arcor 25 Gramos Display De 50 Unidades',11999.4,870,1942,200,v\_uuid\_proveedor2,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod05,'Arroz Yamani 500g Dos Hermanos Integral Sin Tacc Libre Gluten',6017.0,1803,7500,780,v\_uuid\_proveedor3,'importado');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod06,'Picadillo Marolio 90g',1648.98,680,3800,230,v\_uuid\_proveedor1,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod07,'Mermelada Marolio Damasco Frasco 454 Gr',2240.0,213,1300,25,v\_uuid\_proveedor1,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod08,'Mermelada Light De Ciruela Arcor X 390 Grs',2559.0,329,1150,20,v\_uuid\_proveedor2,'importado');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod09,'Bocadito Holanda Arcor X 24 Unidades',9799.0,871,900,50,v\_uuid\_proveedor2,'nacional');  INSERT INTO Productos(idproducto,descripcion,preciounitario,stock,stockmax,stockmin,idproveedor,origen)  VALUES(v\_uuid\_prod10,'Palmito Rodaja 800 Gramos Marolio',7900.0,852,2500,500,v\_uuid\_proveedor1,'importado');  -- Insertar Vendedores  INSERT INTO Vendedor(idvendedor,dni,apellido,nombres,email,comision)  VALUES(v\_uuid\_vendedor1,'36113214','Garay','Mauricio Elio','mgaray@msn.com',10.15);  INSERT INTO Vendedor(idvendedor,dni,apellido,nombres,email,comision)  VALUES(v\_uuid\_vendedor2,'28101438','Cabral Perez','Matías','mcp@outlook.com',23.2);  INSERT INTO Vendedor(idvendedor,dni,apellido,nombres,email,comision)  VALUES(v\_uuid\_vendedor3,'24741573','Castellanos','Matías','mcastellanos@gmail.com',14.6);  -- Insertar Pedidos y capturar numeropedido  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido01,v\_uuid\_cliente1,v\_uuid\_vendedor1,DATE '2025-02-23','confirmado')  RETURNING numeropedido INTO v\_numPedido1;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido02,v\_uuid\_cliente5,v\_uuid\_vendedor2,DATE '2025-03-14','confirmado')  RETURNING numeropedido INTO v\_numPedido2;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido03,v\_uuid\_cliente1,v\_uuid\_vendedor1,DATE '2025-04-04','pendiente')  RETURNING numeropedido INTO v\_numPedido3;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido04,v\_uuid\_cliente4,v\_uuid\_vendedor2,DATE '2025-01-28','confirmado')  RETURNING numeropedido INTO v\_numPedido4;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido05,v\_uuid\_cliente2,v\_uuid\_vendedor3,DATE '2025-04-11','confirmado')  RETURNING numeropedido INTO v\_numPedido5;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido06,v\_uuid\_cliente2,v\_uuid\_vendedor3,DATE '2025-02-18','pendiente')  RETURNING numeropedido INTO v\_numPedido6;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido07,v\_uuid\_cliente1,v\_uuid\_vendedor3,DATE '2025-01-08','confirmado')  RETURNING numeropedido INTO v\_numPedido7;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido08,v\_uuid\_cliente3,v\_uuid\_vendedor2,DATE '2025-03-05','confirmado')  RETURNING numeropedido INTO v\_numPedido8;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido09,v\_uuid\_cliente4,v\_uuid\_vendedor2,DATE '2025-04-10','pendiente')  RETURNING numeropedido INTO v\_numPedido9;  INSERT INTO Pedidos(idpedido,idcliente,idvendedor,fecha,estado)  VALUES(v\_uuid\_pedido10,v\_uuid\_cliente3,v\_uuid\_vendedor2,DATE '2025-03-21','confirmado')  RETURNING numeropedido INTO v\_numPedido10;  -- Insertar en DetallePedidos  -- Nota, los números de pedido son autoincrementales (no se introducen manualmente),  -- así que recupero el valor que necesito en cada caso, realizando una consulta (tengo/conozco el @uuid\_pedidoNN)  -- Pedido 01 de 10 (3 renglones)  INSERT INTO DetallePedidos(iddetallepedido,numeropedido,renglon,idproducto,cantidad)  VALUES(SYS\_GUID(),v\_numPedido1,1,v\_uuid\_prod01,58);  INSERT INTO DetallePedidos(iddetallepedido,numeropedido,renglon,idproducto,cantidad)  VALUES(SYS\_GUID(),v\_numPedido1,2,v\_uuid\_prod02,32);  INSERT INTO DetallePedidos(iddetallepedido,numeropedido,renglon,idproducto,cantidad)  VALUES(SYS\_GUID(),v\_numPedido1,3,v\_uuid\_prod03,211);  -- Pedido 02 de 10 (1 renglón)  INSERT INTO DetallePedidos(iddetallepedido,numeropedido,renglon,idproducto,cantidad)  VALUES(SYS\_GUID(),v\_numPedido2,1,v\_uuid\_prod05,36);  -- Pedido 03 de 10 (2 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido3, 1, v\_uuid\_prod01, 9);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido3, 2, v\_uuid\_prod04, 12);  -- Pedido 04 de 10 (3 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido4, 1, v\_uuid\_prod09, 15);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido4, 2, v\_uuid\_prod06, 22);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido4, 3, v\_uuid\_prod08, 10);  -- Pedido 05 de 10 (1 renglón)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido5, 1, v\_uuid\_prod10, 14);  -- Pedido 06 de 10 (2 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido6, 1, v\_uuid\_prod04, 75);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido6, 2, v\_uuid\_prod08, 23);  -- Pedido 07 de 10 (3 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido7, 1, v\_uuid\_prod07, 38);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido7, 2, v\_uuid\_prod04, 52);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido7, 3, v\_uuid\_prod01, 92);  -- Pedido 08 de 10 (2 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido8, 1, v\_uuid\_prod08, 108);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido8, 2, v\_uuid\_prod06, 625);  -- Pedido 09 de 10 (1 renglón)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido9, 1, v\_uuid\_prod02, 458);  -- Pedido 10 de 10 (3 renglones)  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido10, 1, v\_uuid\_prod05, 15);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido10, 2, v\_uuid\_prod03, 22);  INSERT INTO DetallePedidos(iddetallepedido, numeropedido, renglon, idproducto, cantidad)  VALUES (SYS\_GUID(), v\_numPedido10, 3, v\_uuid\_prod08,210);  COMMIT;  END;  / |

:

# Consignas específicas del TP2

Se desarrolla en los siguientes títulos, correspondientes a los 5 puntos específicos del trabajo práctico 2 (las consultas se realizaron con el software *Oracle SQL Developer 24.3.1*).

## 1. Bloque PL/SQL para realización de un pedido

Crear un bloque PL SQL que permita, mediante una transacción, realizar el registro de un pedido con su detalle (renglones). El proceso debe contemplar la actualización del stock de los productos pedidos. En caso de producirse un error, la transacción debe ser cancelada.

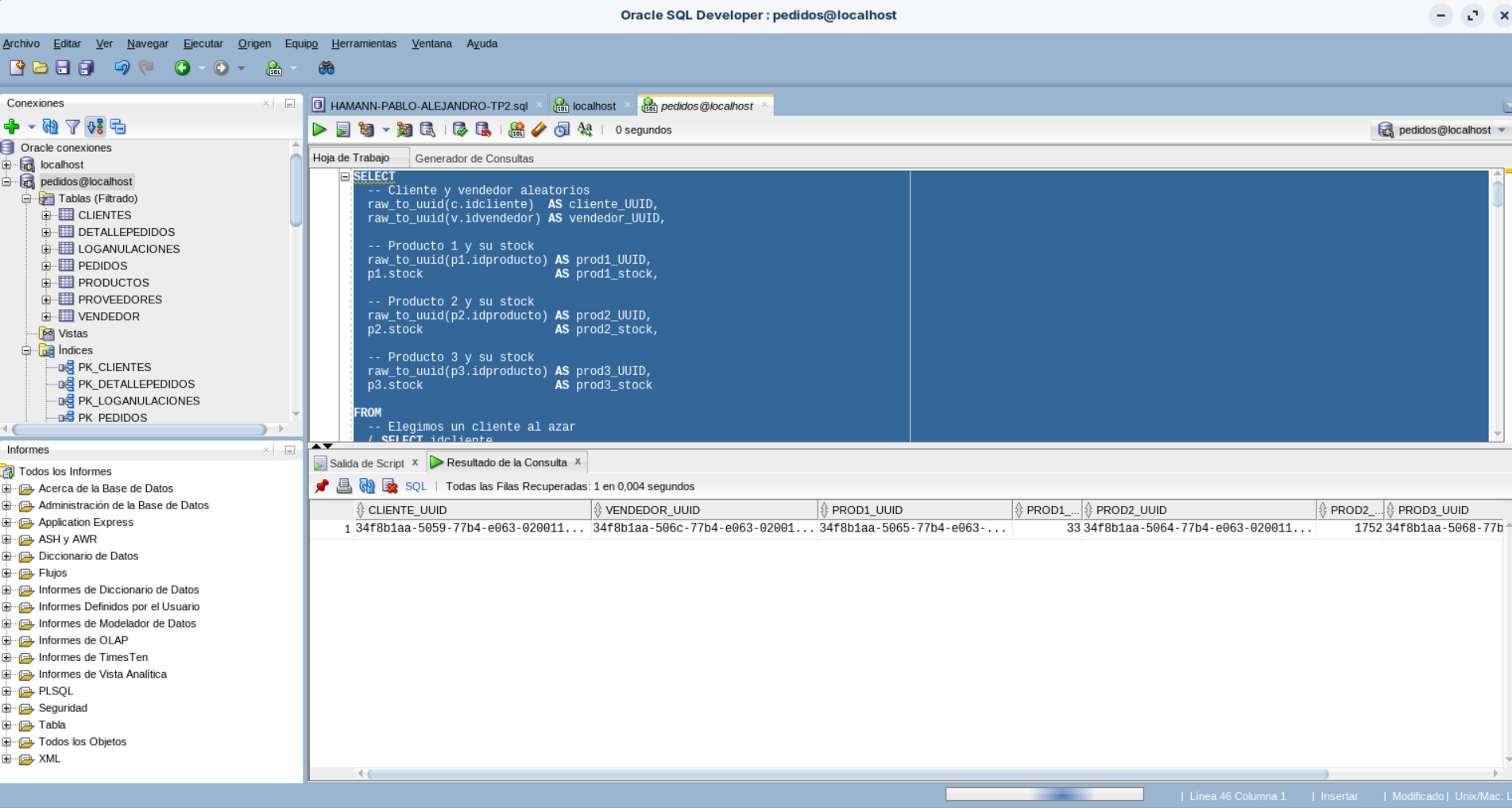
RESTRICCIÓN PLANTEADA: se pueden tomar hasta 3 productos por pedido.

|  |
| --- |
| SET SERVEROUTPUT ON  /  -- Pedidos de valores al usuario (intercalado ID y respectiva cantidad)  ACCEPT cli\_uuid CHAR PROMPT 'UUID Cliente (36 chars, incl. guiones): '  ACCEPT vend\_uuid CHAR PROMPT 'UUID Vendedor (36 chars, incl. guiones): '  ACCEPT prod1\_uuid CHAR PROMPT '1) UUID Producto 1: '  ACCEPT qty1 NUMBER PROMPT ' Cantidad producto 1: '  ACCEPT prod2\_uuid CHAR PROMPT '2) UUID Producto 2 [ENTER para omitir]: '  ACCEPT qty2 NUMBER PROMPT ' Cantidad producto 2: '  ACCEPT prod3\_uuid CHAR PROMPT '3) UUID Producto 3 [ENTER para omitir]: '  ACCEPT qty3 NUMBER PROMPT ' Cantidad producto 3: '  /  DECLARE  -- Cabecera  v\_idPedido RAW(16) := SYS\_GUID();  v\_numPedido NUMBER;  -- Valores ingresados (texto)  v\_cli\_hex VARCHAR2(36) := '&cli\_uuid';  v\_vend\_hex VARCHAR2(36) := '&vend\_uuid';  v\_prod1\_hex VARCHAR2(36) := '&prod1\_uuid';  v\_prod2\_hex VARCHAR2(36) := '&prod2\_uuid';  v\_prod3\_hex VARCHAR2(36) := '&prod3\_uuid';  v\_qty1 PLS\_INTEGER := &qty1;  v\_qty2 PLS\_INTEGER := &qty2;  v\_qty3 PLS\_INTEGER := &qty3;  -- Conversión a RAW  v\_idCliente RAW(16);  v\_idVendedor RAW(16);  -- Tablas asociativas para detalle  TYPE t\_raw\_tab IS TABLE OF RAW(16) INDEX BY PLS\_INTEGER;  TYPE t\_int\_tab IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  v\_ids t\_raw\_tab;  v\_qtys t\_int\_tab;  v\_stock NUMBER;  v\_n PLS\_INTEGER := 1; -- mínimo 1 renglón  BEGIN  -- Validar y convertir Cliente  IF NOT REGEXP\_LIKE(v\_cli\_hex, '^[0-9A-Fa-f]{8}(-[0-9A-Fa-f]{4}){3}-[0-9A-Fa-f]{12}$') THEN  RAISE\_APPLICATION\_ERROR(-20030, 'UUID Cliente inválido: '||v\_cli\_hex);  END IF;  v\_idCliente := uuid\_to\_raw(v\_cli\_hex);  -- Validar y convertir Vendedor  IF NOT REGEXP\_LIKE(v\_vend\_hex, '^[0-9A-Fa-f]{8}(-[0-9A-Fa-f]{4}){3}-[0-9A-Fa-f]{12}$') THEN  RAISE\_APPLICATION\_ERROR(-20031, 'UUID Vendedor inválido: '||v\_vend\_hex);  END IF;  v\_idVendedor := uuid\_to\_raw(v\_vend\_hex);  -- Producto 1 (obligatorio)  IF NOT REGEXP\_LIKE(v\_prod1\_hex, '^[0-9A-Fa-f]{8}(-[0-9A-Fa-f]{4}){3}-[0-9A-Fa-f]{12}$') THEN  RAISE\_APPLICATION\_ERROR(-20032, 'UUID Producto 1 inválido: '||v\_prod1\_hex);  END IF;  v\_ids(1) := uuid\_to\_raw(v\_prod1\_hex);  v\_qtys(1) := v\_qty1;  -- Producto 2 (opcional)  IF TRIM(v\_prod2\_hex) IS NOT NULL THEN  IF v\_qty2 IS NULL THEN  RAISE\_APPLICATION\_ERROR(-20033, 'Debe indicar cantidad para producto 2');  END IF;  IF NOT REGEXP\_LIKE(v\_prod2\_hex, '^[0-9A-Fa-f]{8}(-[0-9A-Fa-f]{4}){3}-[0-9A-Fa-f]{12}$') THEN  RAISE\_APPLICATION\_ERROR(-20034, 'UUID Producto 2 inválido: '||v\_prod2\_hex);  END IF;  v\_ids(2) := uuid\_to\_raw(v\_prod2\_hex);  v\_qtys(2) := v\_qty2;  v\_n := 2;  END IF;  -- Producto 3 (opcional)  IF TRIM(v\_prod3\_hex) IS NOT NULL THEN  IF v\_qty3 IS NULL THEN  RAISE\_APPLICATION\_ERROR(-20035, 'Debe indicar cantidad para producto 3');  END IF;  IF NOT REGEXP\_LIKE(v\_prod3\_hex, '^[0-9A-Fa-f]{8}(-[0-9A-Fa-f]{4}){3}-[0-9A-Fa-f]{12}$') THEN  RAISE\_APPLICATION\_ERROR(-20036, 'UUID Producto 3 inválido: '||v\_prod3\_hex);  END IF;  v\_ids(3) := uuid\_to\_raw(v\_prod3\_hex);  v\_qtys(3) := v\_qty3;  v\_n := 3;  END IF;  -- Insertar cabecera y obtener numero de pedido  INSERT INTO Pedidos(idpedido, idcliente, idvendedor, fecha, estado)  VALUES(v\_idPedido, v\_idCliente, v\_idVendedor, SYSDATE, 'pendiente')  RETURNING numeropedido INTO v\_numPedido;  -- Recorrer 1..v\_n y procesar cada línea  FOR i IN 1..v\_n LOOP  -- Intento leer el stock, pero si no existe el producto, capturo el error  BEGIN  SELECT stock INTO v\_stock  FROM Productos  WHERE idproducto = v\_ids(i)  FOR UPDATE;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(  -20040,  'Producto inexistente: ' || raw\_to\_uuid(v\_ids(i))  );  END;  IF v\_stock < v\_qtys(i) THEN  RAISE\_APPLICATION\_ERROR(  -20010,  'Stock insuficiente (prod '||raw\_to\_uuid(v\_ids(i))||  '): dispo '||v\_stock||', solicitado '||v\_qtys(i)  );  END IF;  -- Insertar detalle  INSERT INTO DetallePedidos(  iddetallepedido, numeropedido, renglon, idproducto, cantidad  ) VALUES(  SYS\_GUID(), v\_numPedido, i, v\_ids(i), v\_qtys(i)  );  -- Actualizar stock  UPDATE Productos  SET stock = stock - v\_qtys(i)  WHERE idproducto = v\_ids(i);  END LOOP;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE(  'Pedido '||v\_numPedido||' registrado con '||v\_n||' renglones.'  );  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('ERROR: '||SQLERRM);  END;  / |

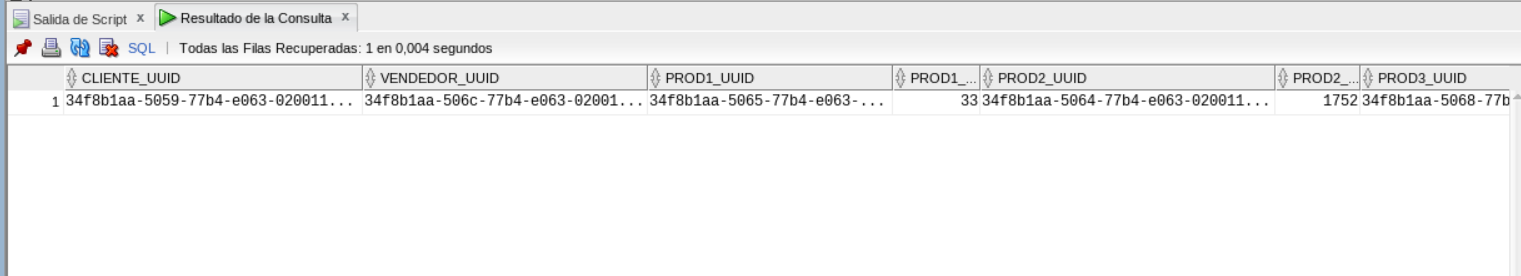
A los fines de poder probar el bloque PL/SQL desarrollado, necesito obtener los ID (definidos como UUIDs generados aleatoriamente), para poder introducir en el bloque PL/SQL (cuando sea ejecutado). Entonces, realizamos la siguiente consulta para obtenerlos:

|  |
| --- |
| SELECT  -- Cliente y vendedor aleatorios  raw\_to\_uuid(c.idcliente) AS cliente\_UUID,  raw\_to\_uuid(v.idvendedor) AS vendedor\_UUID,  -- Producto 1 y su stock  raw\_to\_uuid(p1.idproducto) AS prod1\_UUID,  p1.stock AS prod1\_stock,  -- Producto 2 y su stock  raw\_to\_uuid(p2.idproducto) AS prod2\_UUID,  p2.stock AS prod2\_stock,  -- Producto 3 y su stock  raw\_to\_uuid(p3.idproducto) AS prod3\_UUID,  p3.stock AS prod3\_stock  FROM  -- Elegimos un cliente al azar  ( SELECT idcliente  FROM (SELECT idcliente FROM Clientes ORDER BY DBMS\_RANDOM.VALUE)  WHERE ROWNUM = 1  ) c  -- Elegimos un vendedor al azar  ,( SELECT idvendedor  FROM (SELECT idvendedor FROM Vendedor ORDER BY DBMS\_RANDOM.VALUE)  WHERE ROWNUM = 1  ) v  -- Producto 1 aleatorio  ,( SELECT idproducto, stock  FROM (SELECT idproducto, stock FROM Productos ORDER BY DBMS\_RANDOM.VALUE)  WHERE ROWNUM = 1  ) p1  -- Producto 2 aleatorio  ,( SELECT idproducto, stock  FROM (SELECT idproducto, stock FROM Productos ORDER BY DBMS\_RANDOM.VALUE)  WHERE ROWNUM = 1  ) p2  -- Producto 3 aleatorio  ,( SELECT idproducto, stock  FROM (SELECT idproducto, stock FROM Productos ORDER BY DBMS\_RANDOM.VALUE)  WHERE ROWNUM = 1  ) p3  ; |

### Resultado de la consulta 1



Vista del resultado de la consulta:



#### Puedo observar que dispongo de:

ID Cliente: 34f8b1aa-5059-77b4-e063-020011acf62e

ID Vendedor: 34f8b1aa-506c-77b4-e063-020011acf62e

ID Producto 1: 34f8b1aa-5065-77b4-e063-020011acf62e

Disponibilidad de Prod. 1: 33

ID Producto 2: 34f8b1aa-5064-77b4-e063-020011acf62e

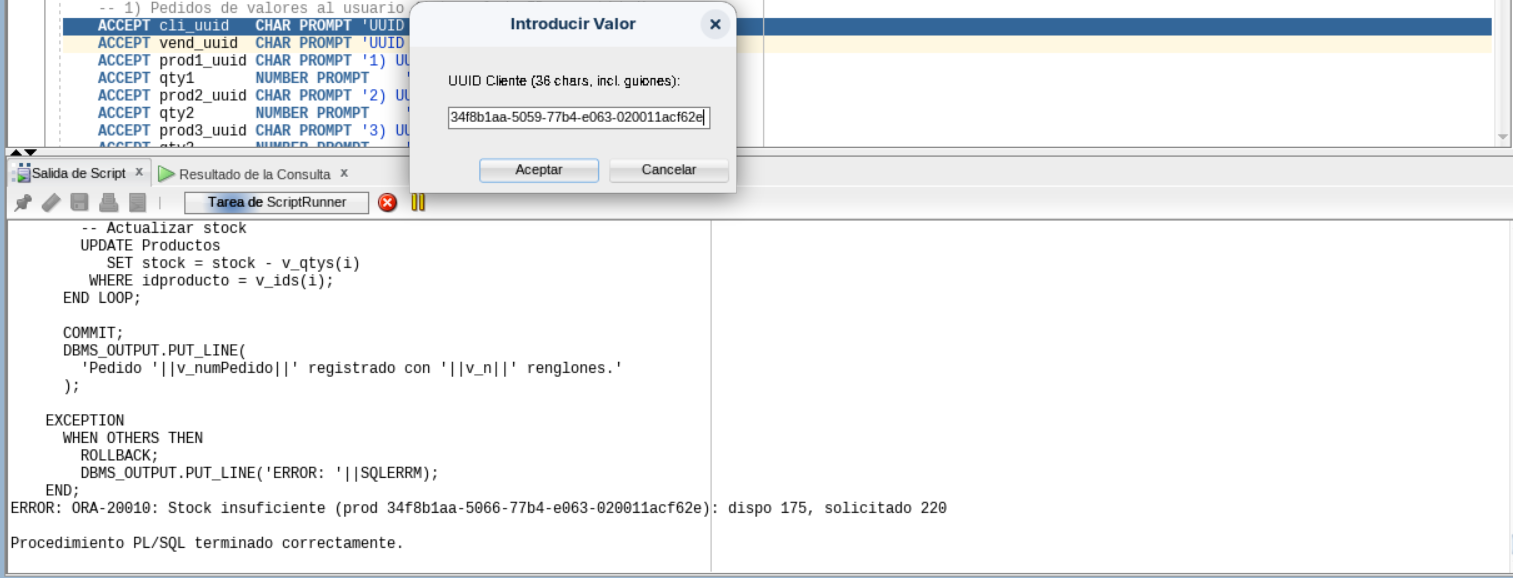
Disponibilidad de Prod. 2: 1752

ID Producto 3: 34f8b1aa-5068-77b4-e063-020011acf62e

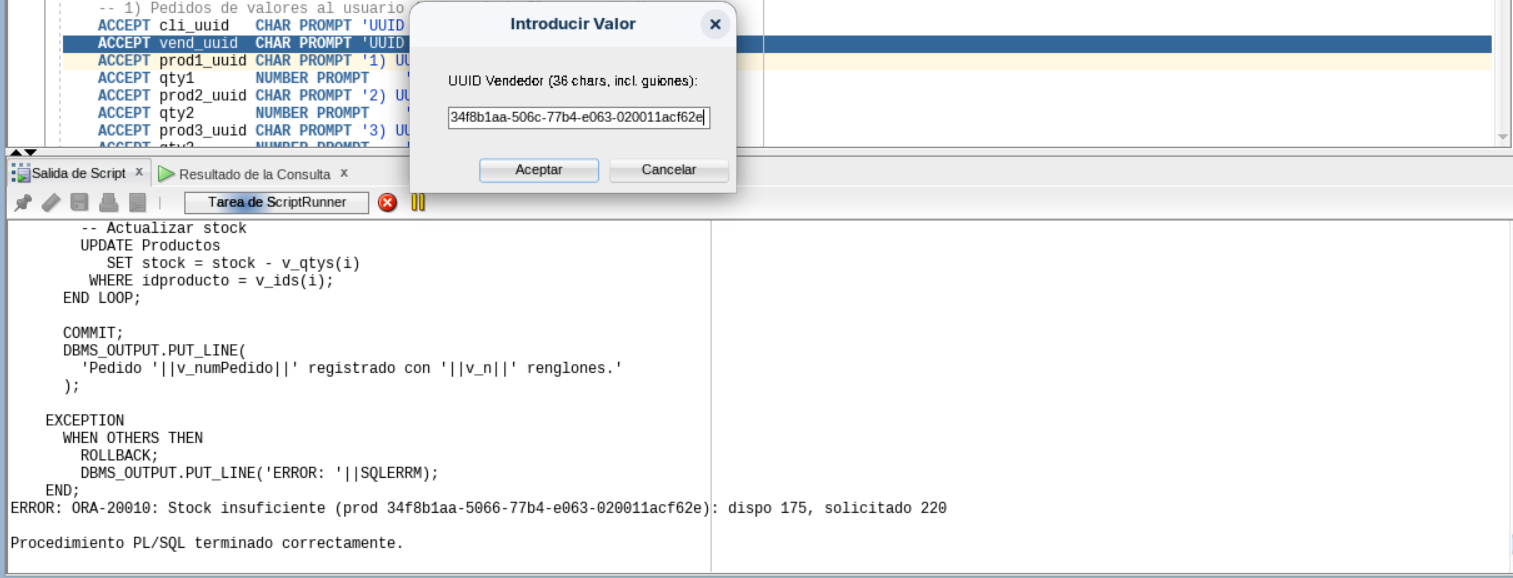
Disponibilidad de Prod. 3: 856

Entonces, ahora puedo probar el bloque PL/SQL desarrollado:

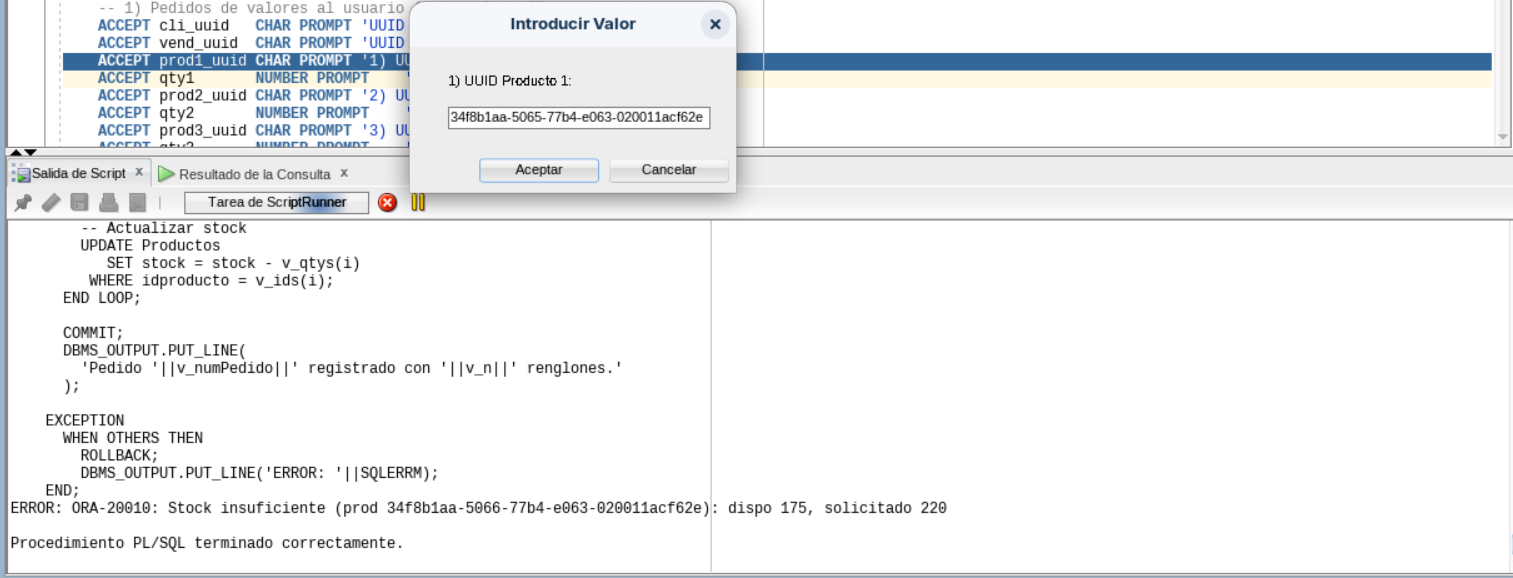
### Solicitud del ID de cliente:



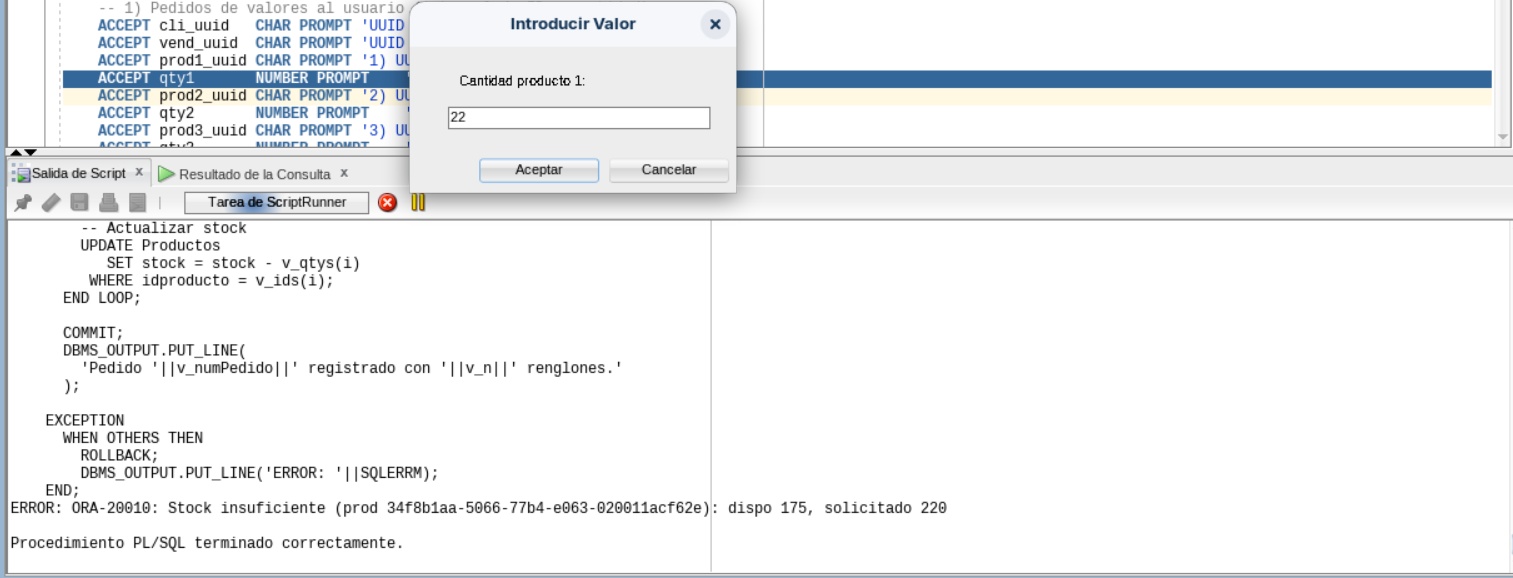
### Solicitud del ID de Vendedor



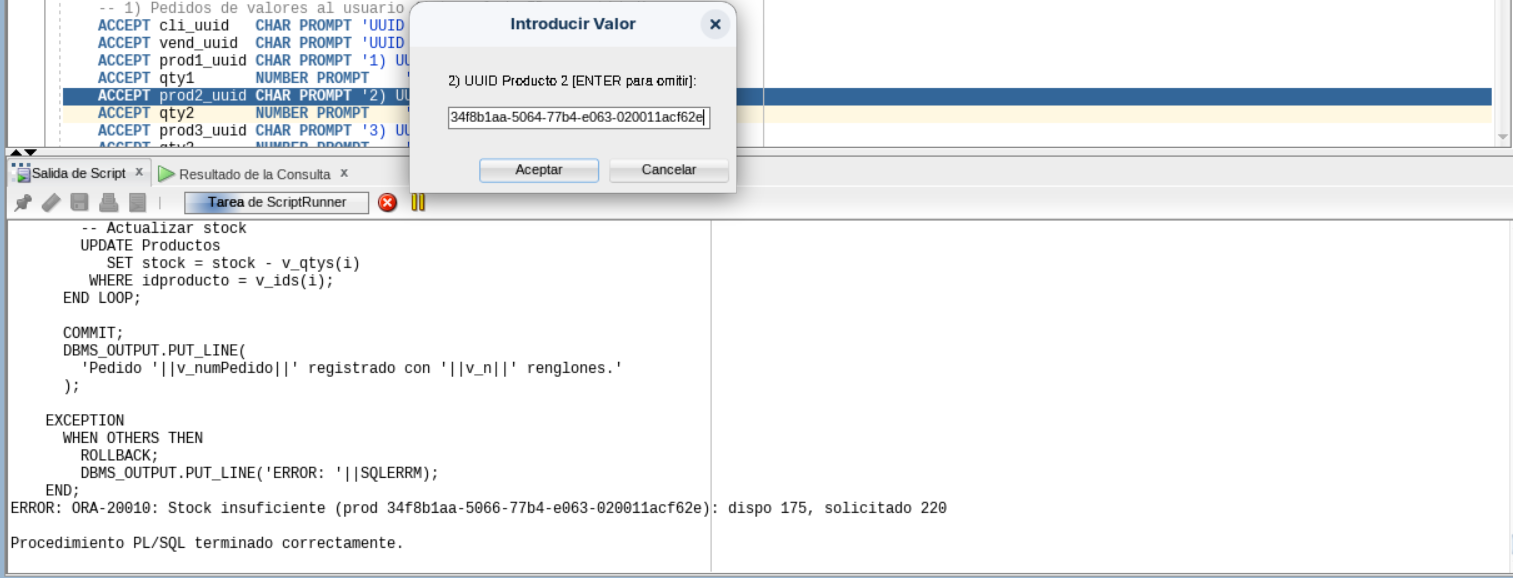
### Solicitud de ID de 1er Producto



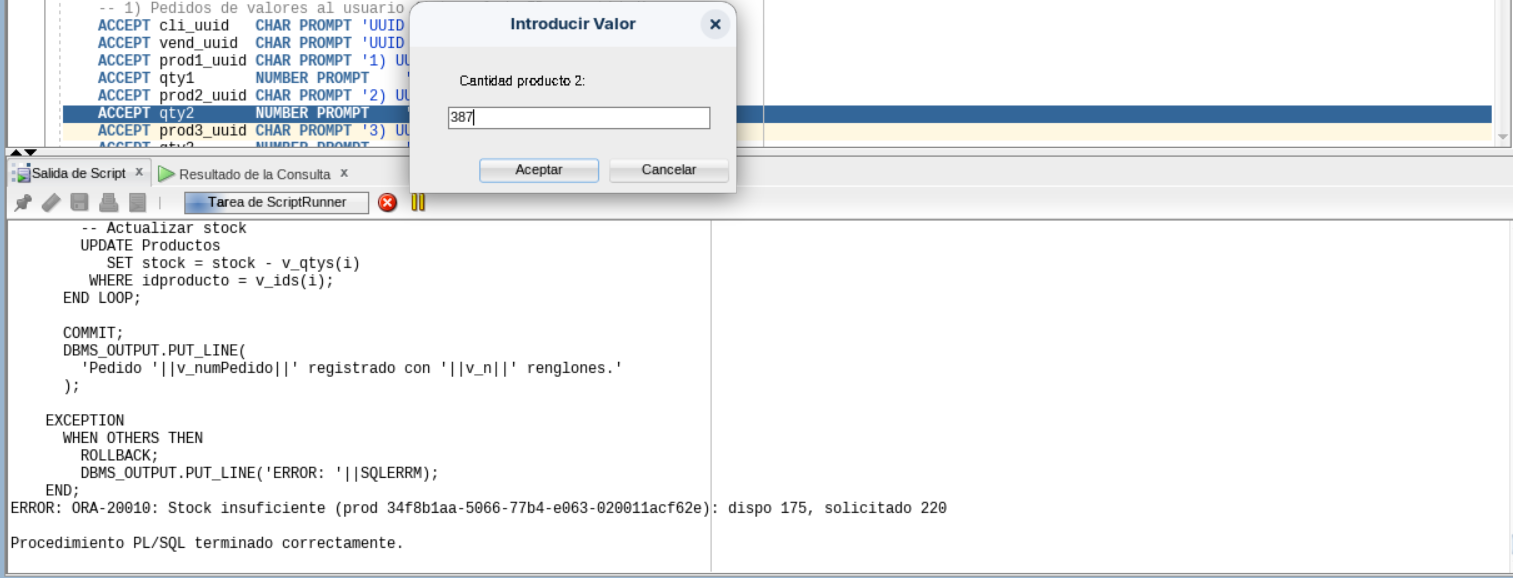
### Solicitud de la cantidad de unidades del 1er producto



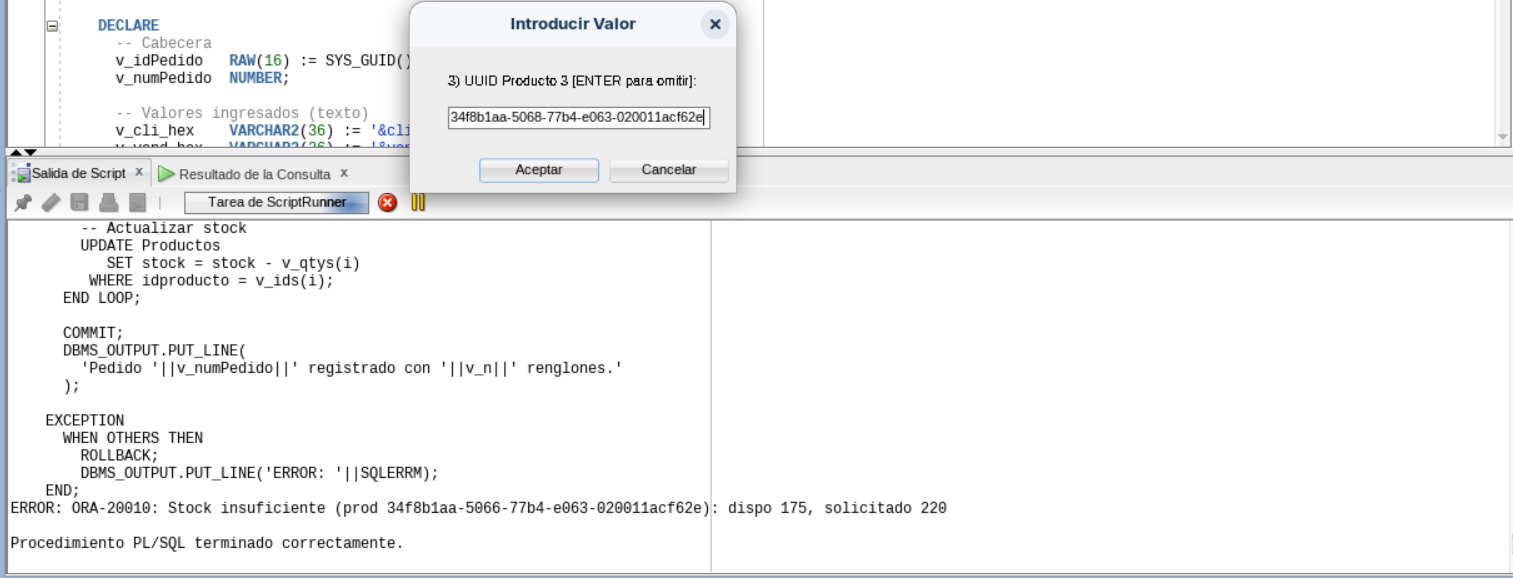
### Solicitud del ID de 2do producto (opcional)



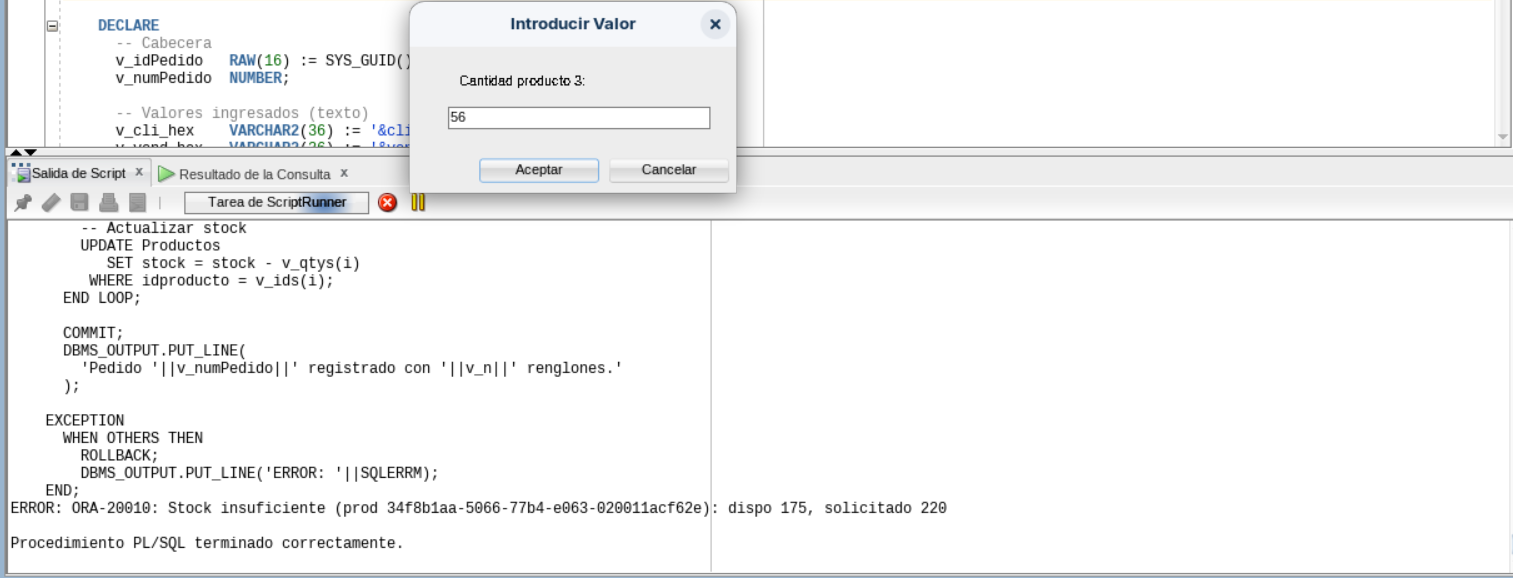
### Cantidad de la cantidad de unidades del 2do producto



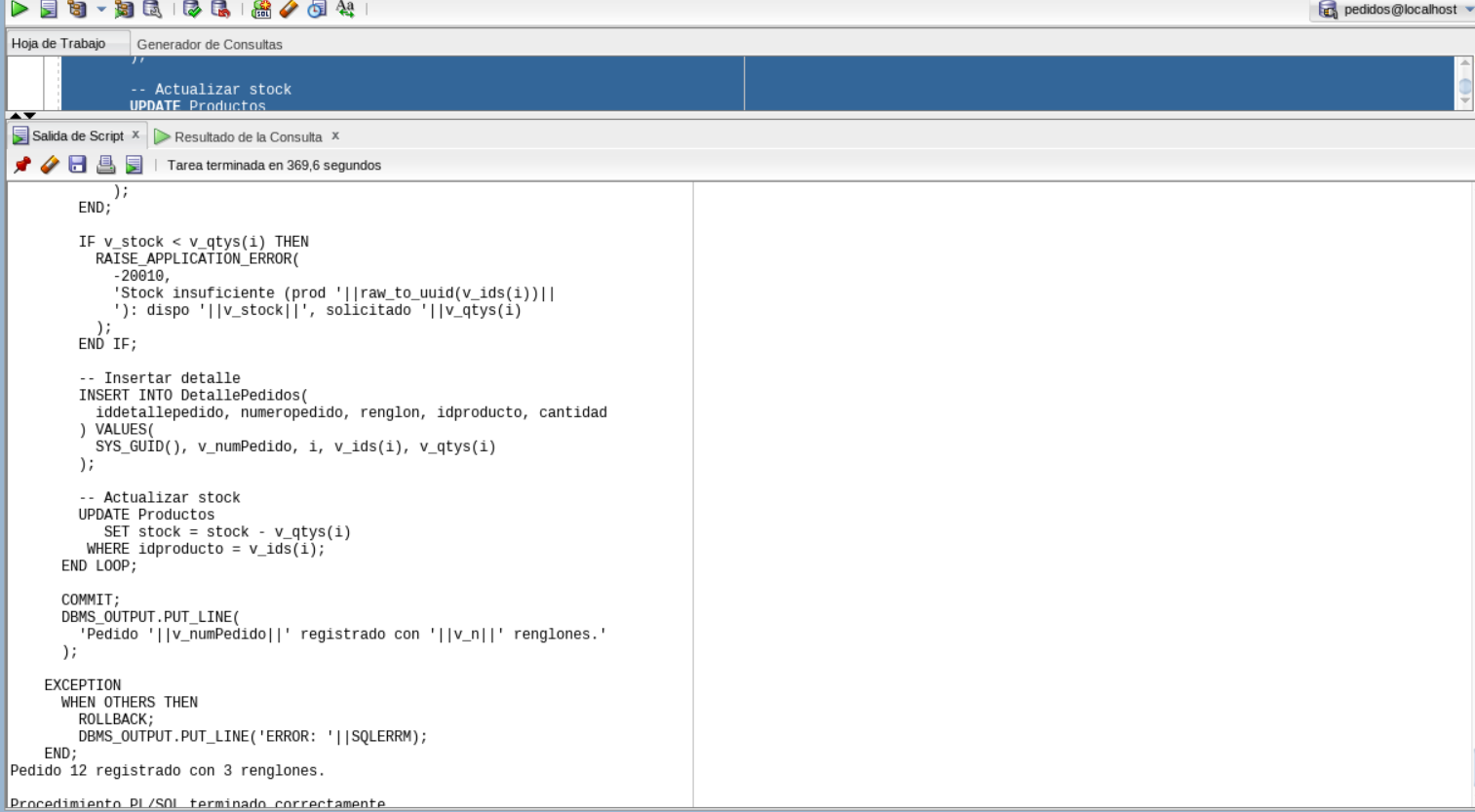
### Solicitud de ID del 3er producto



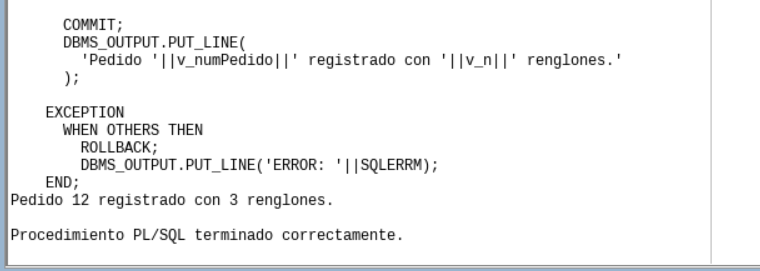
### Solicitud de la cantidad de unidades del 3er producto



### Resultado:



**Se puede observar que el pedido 12 se ha registrado con 3 renglones**



## 2. Procedimiento almacenado para anular un pedido confirmado

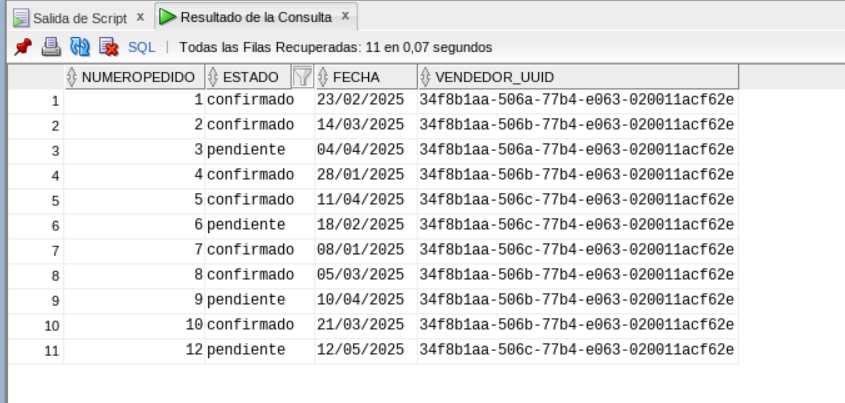
Crear un procedimiento almacenado que permita anular un pedido confirmado. El proceso de anulación debe actualizar los stocks de los artículos del pedido.

|  |
| --- |
| -- Recibimos por parámetro, el número de pedido  CREATE OR REPLACE PROCEDURE anular\_pedido\_confirmado (  p\_numPedido IN NUMBER  ) IS  v\_idPedido RAW(16);  v\_estado VARCHAR2(15);  v\_stock NUMBER;  BEGIN  -- Obtenemos idpedido y el estado del pedido  SELECT idpedido, estado  INTO v\_idPedido, v\_estado  FROM Pedidos  WHERE numeropedido = p\_numPedido;  -- Verificamos que esté confirmado  IF v\_estado <> 'confirmado' THEN  RAISE\_APPLICATION\_ERROR(  -20020,  'No se puede anular el pedido ' || p\_numPedido ||  ' porque su estado actual es "' || v\_estado ||  '". Solo los pedidos en estado CONFIRMADO pueden anularse.'  );  END IF;  -- Reponemos stock para cada renglón  FOR reg IN (  SELECT idproducto, cantidad  FROM DetallePedidos  WHERE numeropedido = p\_numPedido  ) LOOP  UPDATE Productos  SET stock = stock + reg.cantidad  WHERE idproducto = reg.idproducto;  END LOOP;  -- Marcamos como anulado el pedido  UPDATE Pedidos  SET estado = 'anulado'  WHERE numeropedido = p\_numPedido;  -- Registramos en tabla de log  INSERT INTO LogAnulaciones (  idLogAnulaciones,  idpedido,  FechaAnulacion,  Observaciones  ) VALUES (  SYS\_GUID(),  v\_idPedido,  SYSTIMESTAMP,  'Pedido '||p\_numPedido||' anulado.'  );  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(  -20021,  'Pedido no encontrado: '||p\_numPedido  );  WHEN OTHERS THEN  ROLLBACK;  RAISE; -- propagamos otros errores  END anular\_pedido\_confirmado;  / |

Vamos a obtener un listado de pedidos de forma tal que podamos seleccionar alguno que nos permita probar el procedimiento almacenado anterior y (posteriormente a su ejecución) luego poder verificar que se anuló.

|  |
| --- |
| SELECT  p.numeropedido AS NumeroPedido,  p.estado AS Estado,  p.fecha AS Fecha,  raw\_to\_uuid(p.idvendedor) AS Vendedor\_UUID  FROM  Pedidos p  ORDER BY  p.numeropedido; |

### Listado de pedidos y sus estados

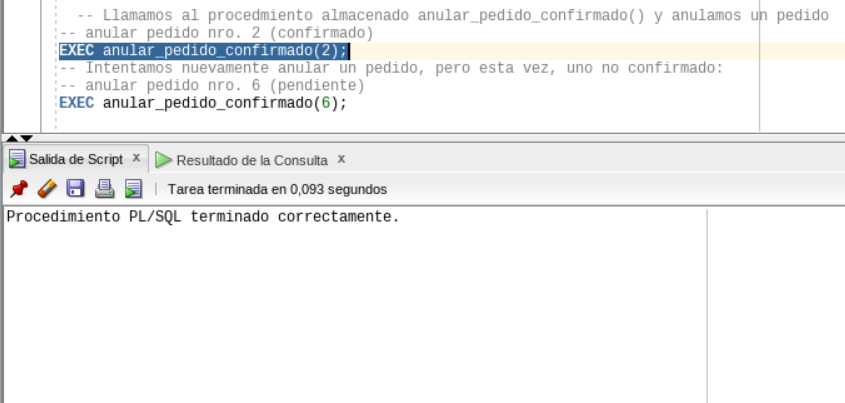


### Primera ejecución del procedimiento almacenado

Llamamos al procedimiento almacenado anular\_pedido\_confirmado() y anulamos un pedido **confirmado**.

|  |
| --- |
| -- anular pedido nro. 2 (confirmado)  EXEC anular\_pedido\_confirmado(2); |

### Resultado de la primera ejección del procedimiento almacenado

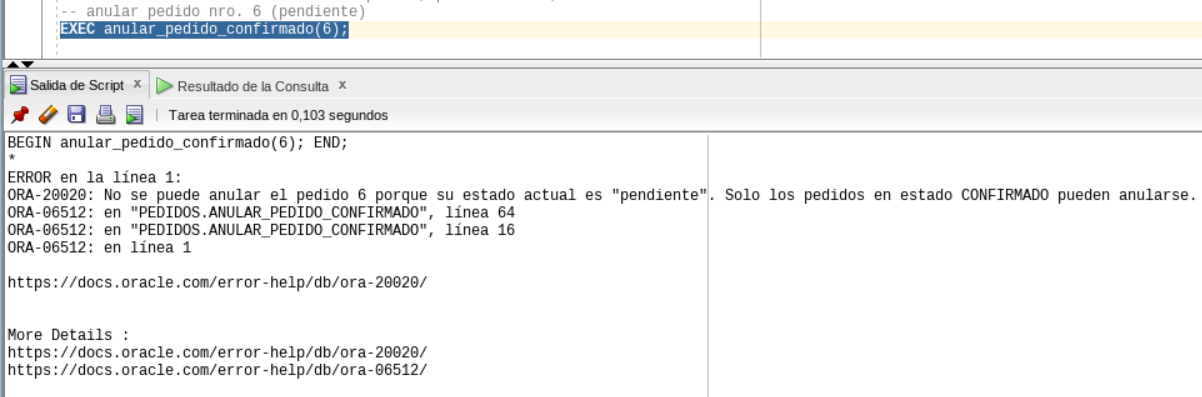


### Segunda ejecución del procedimiento almacenado

Llamamos nuevamente al procedimiento almacenado anular\_pedido\_confirmado() pero esta vez, para intentar eliminar un pedido **pendiente**, por ejemplo, el pedido número 6.

|  |
| --- |
| -- anular pedido nro. 6 (pendiente)  EXEC anular\_pedido\_confirmado(6); |

### Resultado de la segunda ejección del procedimiento almacenado



## 3. Creación de una tabla log

Crear una tabla denominada log (idlog, numeroPedido, FechaAnulacion).

Como se comentó, si bien ya contábamos con una tabla LogAnulaciones (desde el TP1), existen sutiles diferencias con la solicitada en este punto:

* **nombre:** 'log' (en vez de 'LogAnulaciones')
* **campo:** 'numeroPedido' (en lugar de 'idpedido')
* **sin campo** 'Observaciones'

|  |
| --- |
| CREATE TABLE log (  idlog RAW(16) NOT NULL  CONSTRAINT pk\_log PRIMARY KEY,  numeroPedido NUMBER NOT NULL  CONSTRAINT fk\_log\_numPedido  REFERENCES Pedidos(numeroPedido),  fechaAnulacion TIMESTAMP DEFAULT SYSTIMESTAMP NOT NULL  ); |

## 4. Creación de un *trigger* para registros en la tabla log

Crear un *trigger* que permita, al momento de anularse un pedido, registrar en la tabla log, el número de pedido anulado y la fecha de anulación.

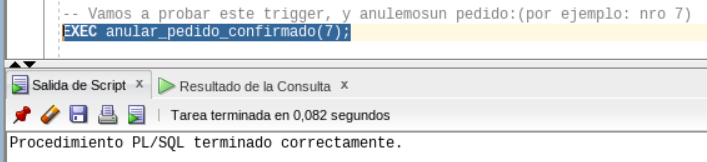
|  |
| --- |
| CREATE OR REPLACE TRIGGER trg\_after\_update\_anulacion  AFTER UPDATE OF estado  ON Pedidos  FOR EACH ROW  WHEN (  NEW.estado = 'anulado'  AND OLD.estado <> 'anulado'  )  BEGIN  INSERT INTO log (  idlog,  numeroPedido,  fechaAnulacion  ) VALUES (  SYS\_GUID(),  :NEW.numeroPedido,  SYSTIMESTAMP  );  END;  / |

### Prueba de funcionamiento del trigger

Vamos a probar este trigger, anulando un pedido, por ejemplo, el pedido número 7.

|  |
| --- |
| -- Anular pedido nro 7  EXEC anular\_pedido\_confirmado(7); |

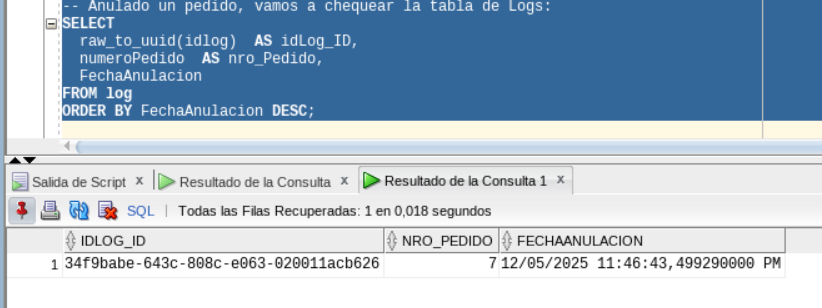
Resultado:



### Consulta a la tabla de log

|  |
| --- |
| -- Anulado un pedido, vamos a chequear la tabla de Logs:  SELECT  raw\_to\_uuid(idlog) AS idLog\_ID,  numeroPedido AS nro\_Pedido,  FechaAnulacion  FROM log  ORDER BY FechaAnulacion DESC; |

### Resultado de la consulta a la tabla log



## 5. Procedimiento almacenado para actualización de precios

Crear un procedimiento almacenado que permita actualizar el precio de los artículos de un determinado origen en un determinado porcentaje.

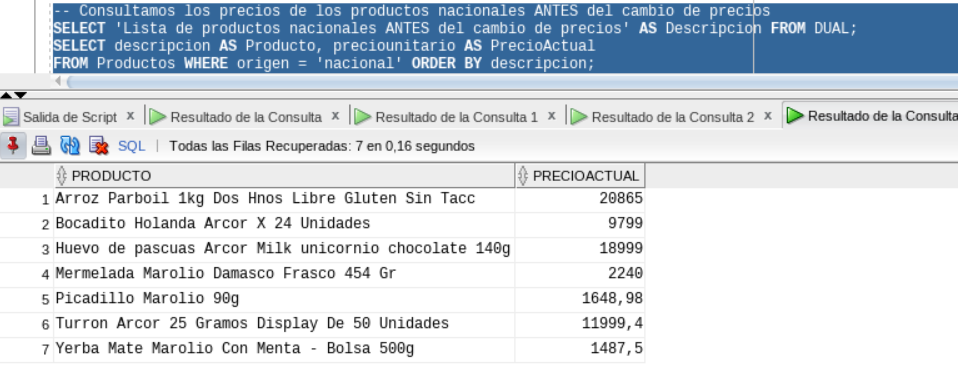
|  |
| --- |
| CREATE OR REPLACE PROCEDURE actualizar\_precio\_por\_origen (  p\_origen IN VARCHAR2,  p\_porcentaje IN NUMBER  ) IS  -- factor de ajuste calculado a partir del porcentaje  v\_factor NUMBER := 1 + p\_porcentaje/100;  BEGIN  -- actualizams precios multiplicando por factor de ajuste. Redondeamos 2 dec.  UPDATE Productos  SET preciounitario = ROUND(preciounitario \* v\_factor, 2)  WHERE origen = p\_origen;  COMMIT;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK; -- si lacosa falla, deshacer cambios y generar error con código personalizado  RAISE\_APPLICATION\_ERROR(  -20050,  'Error al actualizar precios para origen "'||p\_origen||'": '||SQLERRM  );  END actualizar\_precio\_por\_origen;  / |

### Consulta de artículos nacionales antes de su modificación de precios

Consultamos los precios de los productos nacionales ANTES del cambio de precios

|  |
| --- |
| SELECT 'Lista de productos nacionales ANTES del cambio de precios' AS Descripcion FROM DUAL;  SELECT descripcion AS Producto, preciounitario AS PrecioActual  FROM Productos WHERE origen = 'nacional' ORDER BY descripcion; |

Resultado de la consulta

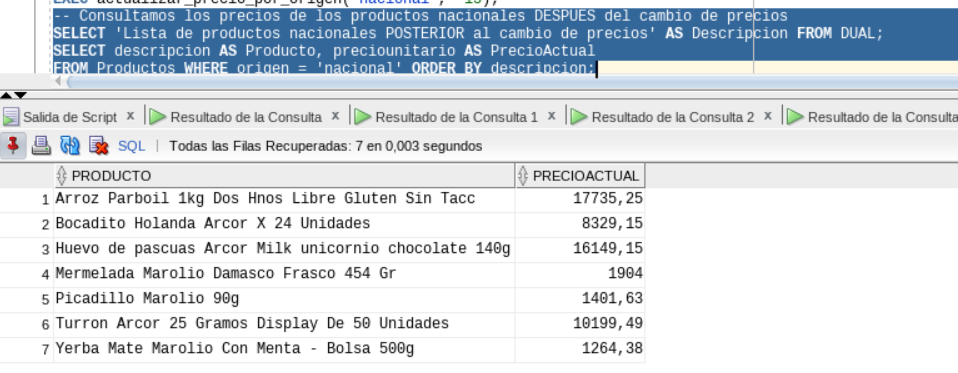


### Consulta de artículos nacionales posterior a su modificación de precios

Realizamos una baja del 15% en los precios para productos nacionales.

|  |
| --- |
| -- Bajamos los productos nacionales un 15%  EXEC actualizar\_precio\_por\_origen('nacional', -15); |

Y consultamos los precios:



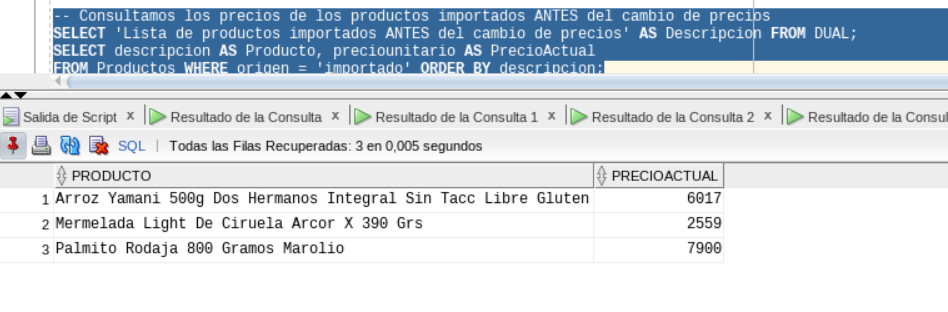
*(continúa en página siguiente)*

### Consulta de artículos importados antes de su modificación de precios

Consultamos los precios de los productos importados ANTES del cambio de precios

|  |
| --- |
| SELECT 'Lista de productos importados ANTES del cambio de precios' AS Descripcion FROM DUAL;  SELECT descripcion AS Producto, preciounitario AS PrecioActual  FROM Productos WHERE origen = 'importado' ORDER BY descripcion; |

Resultado de la consulta



### Consulta de artículos importados posterior a su modificación de precios

Realizamos un aumento del 13% en los precios de los productos importados.

|  |
| --- |
| BEGIN  actualizar\_precio\_por\_origen('importado', 13);  END;  / |

Y consultamos los precios:

