



## Proyecto 3 - Bases de Datos

Laura Garzón (is360018) - Juan Rozo (is360012) - Maria P. Rodríguez  
(is360019) - Daniel Castro (is360015)

---

### Creación de DDL

En primer lugar, se realizó la lectura completa del enunciado para establecer una planeación adecuada de la creación física de la base de datos, incluyendo toda la información brindada en el enunciado, la creación del atributo "VALORTOTAL" en la tabla de "DETALLEPEDIDOS", la creación de dos tablas de auditoría "CLIENTES\_LOG" y "PEDIDOS\_LOG", agregar el atributo "DESCUENTO" a "GAMAPRODUCTOS" y, finalmente, una nueva tabla "NOTIFICACIONES". De esta forma, se presenta la eliminación, creación de tablas con detalles técnicos de almacenamiento, inserción de tuplas (se omite en este documento por cuestión de extensión), establecimiento de llaves foráneas y primarias y brindar acceso a JPALACIO.

-----  
-- Delete all tables  
-----

```
DROP TABLE CLIENTES_LOG;  
DROP TABLE PEDIDOS_LOG;  
DROP TABLE DETALLEPEDIDOS;  
DROP TABLE PRODUCTOS;  
DROP TABLE GAMASPRODUCTOS;  
DROP TABLE PEDIDOS;  
DROP TABLE PAGOS;  
DROP TABLE NOTIFICACIONES;  
DROP TABLE CLIENTES;  
DROP TABLE EMPLEADOS;  
DROP TABLE OFICINAS;
```

-----  
--DDL FOR TABLE NOTIFICACIONES  
-----

```
CREATE TABLE "NOTIFICACIONES"  
(  
    "ID_NOTIFICACION" NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    "FECHA" DATE DEFAULT SYSDATE,  
    "DESTINATARIO" NUMBER(*,0),  
    "MENSAJE" VARCHAR2(100 BYTE)  
)  
SEGMENT CREATION IMMEDIATE  
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255  
NOCOMPRESS LOGGING  
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645  
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
```

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT);

-----  
-- DDL for Table CLIENTES  
-----

```
CREATE TABLE "CLIENTES"
(
  "CODIGOCLIENTE" NUMBER(*,0),
  "NOMBRECLIENTE" VARCHAR2(50 BYTE),
  "NOMBRECONTACTO" VARCHAR2(30 BYTE) DEFAULT NULL,
  "APELLIDOCONTACTO" VARCHAR2(30 BYTE) DEFAULT NULL,
  "TELEFONO" VARCHAR2(15 BYTE),
  "FAX" VARCHAR2(15 BYTE),
  "LINEADIRECCION1" VARCHAR2(50 BYTE),
  "LINEADIRECCION2" VARCHAR2(50 BYTE) DEFAULT NULL,
  "CIUDAD" VARCHAR2(50 BYTE),
  "REGION" VARCHAR2(50 BYTE) DEFAULT NULL,
  "PAIS" VARCHAR2(50 BYTE) DEFAULT NULL,
  "CODIGOPOSTAL" VARCHAR2(10 BYTE) DEFAULT NULL,
  "CODIGOEMPLEADOREPVENTAS" NUMBER(*,0) DEFAULT NULL,
  "LIMITECREDITO" NUMBER(15,2) DEFAULT NULL
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
;
```

-----  
-- DDL for Table DETALLEPEDIDOS  
-----

```
CREATE TABLE "DETALLEPEDIDOS"
(
  "CODIGOPEDIDO" NUMBER(*,0),
  "VALORTOTAL" NUMBER(15,2),
  "CODIGOPRODUCTO" VARCHAR2(50 BYTE),
  "CANTIDAD" NUMBER(*,0),
  "PRECIOUNIDAD" NUMBER(15,2),
  "NUMEROLINEA" NUMBER(*,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
;
```

-----

-- DDL for Table EMPLEADOS

---

```
CREATE TABLE "EMPLEADOS"
(
  "CODIGOEMPLEADO" NUMBER(*,0),
  "NOMBRE" VARCHAR2(50 BYTE),
  "APELLIDO1" VARCHAR2(50 BYTE),
  "APELLIDO2" VARCHAR2(50 BYTE) DEFAULT NULL,
  "EXTENSION" VARCHAR2(10 BYTE),
  "EMAIL" VARCHAR2(100 BYTE),
  "CODIGOOFICINA" VARCHAR2(10 BYTE),
  "CODIGOJEFE" NUMBER(*,0) DEFAULT NULL,
  "PUESTO" VARCHAR2(50 BYTE) DEFAULT NULL
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
;
```

---

-- DDL for Table GAMASPRODUCTOS

---

```
CREATE TABLE "GAMASPRODUCTOS"
(
  "GAMA" VARCHAR2(50 BYTE),
  "DESCUENTO" NUMBER(5,2),
  "DESCRIPCIONTEXTO" CLOB,
  "DESCRIPCIONHTML" CLOB,
  "IMAGEN" BLOB
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

LOB ("DESCRIPCIONTEXTO") STORE AS SECUREFILE (
  ENABLE STORAGE IN ROW CHUNK 8192
  NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES
  STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT))
LOB ("DESCRIPCIONHTML") STORE AS SECUREFILE (
  ENABLE STORAGE IN ROW CHUNK 8192
  NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES
  STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
```

```

BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT))
LOB ("IMAGEN") STORE AS SECUREFILE (
  ENABLE STORAGE IN ROW CHUNK 8192
  NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES
  STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)) ;

```

```

-----
-- DDL for Table OFICINAS
-----

```

```

CREATE TABLE "OFICINAS"
(
  "CODIGOOFICINA" VARCHAR2(10 BYTE),
  "CIUDAD" VARCHAR2(30 BYTE),
  "PAIS" VARCHAR2(50 BYTE),
  "REGION" VARCHAR2(50 BYTE) DEFAULT NULL,
  "CODIGOPOSTAL" VARCHAR2(10 BYTE),
  "TELEFONO" VARCHAR2(20 BYTE),
  "LINEADIRECCION1" VARCHAR2(50 BYTE),
  "LINEADIRECCION2" VARCHAR2(50 BYTE) DEFAULT NULL
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
;

```

```

-----
-- DDL for Table PAGOS
-----

```

```

CREATE TABLE "PAGOS"
(
  "CODIGOCLIENTE" NUMBER(*,0),
  "FORMAPAGO" VARCHAR2(40 BYTE),
  "IDTRANSACCION" VARCHAR2(50 BYTE),
  "FECHAPAGO" DATE,
  "CANTIDAD" NUMBER(15,2)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
;

```

```

-----
-- DDL for Table PEDIDOS
-----

```

```

CREATE TABLE "PEDIDOS"
(
  "CODIGOPEDIDO" NUMBER(*,0),
  "FECHAPEDIDO" DATE,
  "FECHAESPERADA" DATE,
  "FECHAENTREGA" DATE DEFAULT NULL,
  "ESTADO" VARCHAR2(15 BYTE),
  "COMENTARIOS" CLOB,
  "CODIGOCLIENTE" NUMBER(*,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

LOB ("COMENTARIOS") STORE AS SECUREFILE (
  ENABLE STORAGE IN ROW CHUNK 8192
  NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES
  STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)) ;

```

-----  
-- DDL for Table PRODUCTOS  
-----

```

CREATE TABLE "PRODUCTOS"
(
  "CODIGOPRODUCTO" VARCHAR2(15 BYTE),
  "NOMBRE" VARCHAR2(70 BYTE),
  "GAMA" VARCHAR2(50 BYTE),
  "DIMENSIONES" VARCHAR2(25 BYTE),
  "PROVEEDOR" VARCHAR2(50 BYTE) DEFAULT NULL,
  "DESCRIPCION" CLOB,
  "CANTIDADENSTOCK" NUMBER(*,0),
  "PRECIOVENTA" NUMBER(15,2),
  "PRECIOPROVEEDOR" NUMBER(15,2) DEFAULT NULL
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

LOB ("DESCRIPCION") STORE AS SECUREFILE (
  ENABLE STORAGE IN ROW CHUNK 8192
  NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES

```

```
STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT));
```

```
-----
-- DDL for Table CLIENTES_LOG
-----
```

```
CREATE TABLE "CLIENTES_LOG" (
  "ID_AUDITORIA" NUMBER,
  "CODIGOCLIENTE" NUMBER,
  "NOMBRE_ANTERIOR" VARCHAR2(100),
  "NOMBRE_NUEVO" VARCHAR2(100),
  "APELLIDO_ANTERIOR" VARCHAR2(100),
  "APELLIDO_NUEVO" VARCHAR2(100),
  "FECHA_MODIFICACION" TIMESTAMP,
  "USUARIO_MODIFICADOR" VARCHAR2(100)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(
  INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT
);
```

```
-----
-- DDL for Table PEDIDOS_LOG
-----
```

```
CREATE TABLE "PEDIDOS_LOG" (
  "ID_AUDITORIA" NUMBER,
  "CODIGOPEDIDO" NUMBER,
  "FECHA_MODIFICACION" TIMESTAMP,
  "USUARIO_MODIFICADOR" VARCHAR2(100),
  "ESTADO_ANTERIOR" VARCHAR2(100),
  "ESTADO_NUEVO" VARCHAR2(100)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(
  INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT
);
```

-----  
-- Constraints for Table PAGOS  
-----

```
ALTER TABLE "PAGOS" MODIFY ("CODIGOCLIENTE" NOT NULL ENABLE);
ALTER TABLE "PAGOS" MODIFY ("FORMAPAGO" NOT NULL ENABLE);
ALTER TABLE "PAGOS" MODIFY ("IDTRANSACCION" NOT NULL ENABLE);
ALTER TABLE "PAGOS" MODIFY ("FECHAPAGO" NOT NULL ENABLE);
ALTER TABLE "PAGOS" MODIFY ("CANTIDAD" NOT NULL ENABLE);
ALTER TABLE "PAGOS" ADD PRIMARY KEY ("CODIGOCLIENTE", "IDTRANSACCION")
USING INDEX PCTFREE 10 INTRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table OFICINAS  
-----

```
ALTER TABLE "OFICINAS" MODIFY ("CODIGOOFICINA" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" MODIFY ("CIUDAD" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" MODIFY ("PAIS" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" MODIFY ("CODIGOPOSTAL" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" MODIFY ("TELEFONO" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" MODIFY ("LINEADIRECCION1" NOT NULL ENABLE);
ALTER TABLE "OFICINAS" ADD PRIMARY KEY ("CODIGOOFICINA")
USING INDEX PCTFREE 10 INTRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table EMPLEADOS  
-----

```
ALTER TABLE "EMPLEADOS" MODIFY ("CODIGOEMPLEADO" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" MODIFY ("NOMBRE" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" MODIFY ("APELLIDO1" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" MODIFY ("EXTENSION" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" MODIFY ("EMAIL" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" MODIFY ("CODIGOOFICINA" NOT NULL ENABLE);
ALTER TABLE "EMPLEADOS" ADD PRIMARY KEY ("CODIGOEMPLEADO")
USING INDEX PCTFREE 10 INTRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table CLIENTES  
-----

```
ALTER TABLE "CLIENTES" MODIFY ("CODIGOCLIENTE" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" MODIFY ("NOMBRECLIENTE" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" MODIFY ("TELEFONO" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" MODIFY ("FAX" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" MODIFY ("LINEADIRECCION1" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" MODIFY ("CIUDAD" NOT NULL ENABLE);
ALTER TABLE "CLIENTES" ADD PRIMARY KEY ("CODIGOCLIENTE")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table DETALLEPEDIDOS  
-----

```
ALTER TABLE "DETALLEPEDIDOS" MODIFY ("CODIGOPEDIDO" NOT NULL ENABLE);
ALTER TABLE "DETALLEPEDIDOS" MODIFY ("CODIGOPRODUCTO" NOT NULL
ENABLE);
ALTER TABLE "DETALLEPEDIDOS" MODIFY ("CANTIDAD" NOT NULL ENABLE);
ALTER TABLE "DETALLEPEDIDOS" MODIFY ("PRECIOUNIDAD" NOT NULL ENABLE);
ALTER TABLE "DETALLEPEDIDOS" MODIFY ("NUMEROLINEA" NOT NULL ENABLE);
ALTER TABLE "DETALLEPEDIDOS" ADD PRIMARY KEY ("CODIGOPEDIDO",
"CODIGOPRODUCTO")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table PRODUCTOS  
-----

```
ALTER TABLE "PRODUCTOS" MODIFY ("CODIGOPRODUCTO" NOT NULL ENABLE);
ALTER TABLE "PRODUCTOS" MODIFY ("NOMBRE" NOT NULL ENABLE);
ALTER TABLE "PRODUCTOS" MODIFY ("GAMA" NOT NULL ENABLE);
ALTER TABLE "PRODUCTOS" MODIFY ("CANTIDADENSTOCK" NOT NULL ENABLE);
ALTER TABLE "PRODUCTOS" MODIFY ("PRECIOVENTA" NOT NULL ENABLE);
ALTER TABLE "PRODUCTOS" ADD PRIMARY KEY ("CODIGOPRODUCTO")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```



-----  
-- Constraints for Table PEDIDOS  
-----

```
ALTER TABLE "PEDIDOS" MODIFY ("CODIGOPEDIDO" NOT NULL ENABLE);
ALTER TABLE "PEDIDOS" MODIFY ("FECHAPEDIDO" NOT NULL ENABLE);
ALTER TABLE "PEDIDOS" MODIFY ("FECHAESPERADA" NOT NULL ENABLE);
ALTER TABLE "PEDIDOS" MODIFY ("ESTADO" NOT NULL ENABLE);
ALTER TABLE "PEDIDOS" MODIFY ("CODIGOCLIENTE" NOT NULL ENABLE);
ALTER TABLE "PEDIDOS" ADD PRIMARY KEY ("CODIGOPEDIDO")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table GAMASPRODUCTOS  
-----

```
ALTER TABLE "GAMASPRODUCTOS" MODIFY ("GAMA" NOT NULL ENABLE);
ALTER TABLE "GAMASPRODUCTOS" ADD PRIMARY KEY ("GAMA")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table CLIENTES\_LOG  
-----

```
ALTER TABLE "CLIENTES_LOG" ADD PRIMARY KEY ("ID_AUDITORIA")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
ENABLE;
```

-----  
-- Constraints for Table PEDIDOS\_LOG  
-----

```
ALTER TABLE "PEDIDOS_LOG" ADD CONSTRAINT "PK_PEDIDOS_LOG" PRIMARY KEY
("ID_AUDITORIA")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
```

ENABLE;

-----  
-- Constrains for Table NOTIFICACIONES  
-----

ALTER TABLE NOTIFICACIONES ADD CONSTRAINT PK\_NOTIFICACIONES PRIMARY  
KEY (ID\_NOTIFICACION)  
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS  
STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645  
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1  
BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)  
ENABLE;

-----  
-- Ref Constraints for Table CLIENTES  
-----

ALTER TABLE "CLIENTES" ADD CONSTRAINT "CLIENTES\_EMPLEADOSFK" FOREIGN  
KEY ("CODIGOEMPLEADOREPVENTAS")  
REFERENCES "EMPLEADOS" ("CODIGOEMPLEADO") ENABLE;

-----  
-- Ref Constraints for Table DETALLEPEDIDOS  
-----

ALTER TABLE "DETALLEPEDIDOS" ADD CONSTRAINT "DETALLEPEDIDOS\_PEDIDOFK"  
FOREIGN KEY ("CODIGOPEDIDO")  
REFERENCES "PEDIDOS" ("CODIGOPEDIDO") ENABLE;  
ALTER TABLE "DETALLEPEDIDOS" ADD CONSTRAINT  
"DETALLEPEDIDOS\_PRODUCTOFK" FOREIGN KEY ("CODIGOPRODUCTO")  
REFERENCES "PRODUCTOS" ("CODIGOPRODUCTO") ENABLE;

-----  
-- Ref Constraints for Table EMPLEADOS  
-----

ALTER TABLE "EMPLEADOS" ADD CONSTRAINT "EMPLEADOS\_OFICINASFK"  
FOREIGN KEY ("CODIGOOFICINA")  
REFERENCES "OFICINAS" ("CODIGOOFICINA") ENABLE;  
ALTER TABLE "EMPLEADOS" ADD CONSTRAINT "EMPLEADOS\_EMPLEADOSFK"  
FOREIGN KEY ("CODIGOJEFE")  
REFERENCES "EMPLEADOS" ("CODIGOEMPLEADO") ENABLE;

-----  
-- Ref Constraints for Table PAGOS  
-----

ALTER TABLE "PAGOS" ADD CONSTRAINT "PAGOS\_CLIENTEFK" FOREIGN KEY  
("CODIGOCLIENTE")  
REFERENCES "CLIENTES" ("CODIGOCLIENTE") ENABLE;

-- Ref Constraints for Table PEDIDOS

---

ALTER TABLE "PEDIDOS" ADD CONSTRAINT "PEDIDOS\_CLIENTE" FOREIGN KEY  
("CODIGOCLIENTE")  
REFERENCES "CLIENTES" ("CODIGOCLIENTE") ENABLE;

---

-- Ref Constraints for Table PRODUCTOS

---

ALTER TABLE "PRODUCTOS" ADD CONSTRAINT "PRODUCTOS\_GAMAFK" FOREIGN  
KEY ("GAMA")  
REFERENCES "GAMASPRODUCTOS" ("GAMA") ENABLE;

---

--- Ref Constrains for Table NOTIFICACIONES

---

ALTER TABLE NOTIFICACIONES ADD CONSTRAINT FK\_DESTINATARIO FOREIGN  
KEY ("DESTINATARIO")  
REFERENCES EMPLEADOS ("CODIGOEMPLEADO") ENABLE;

---

-- Grant all privileges to JPALACIO

---

GRANT ALL ON CLIENTES TO JPALACIO;  
GRANT ALL ON DETALLEPEDIDOS TO JPALACIO;  
GRANT ALL ON EMPLEADOS TO JPALACIO;  
GRANT ALL ON GAMASPRODUCTOS TO JPALACIO;  
GRANT ALL ON OFICINAS TO JPALACIO;  
GRANT ALL ON PAGOS TO JPALACIO;  
GRANT ALL ON PEDIDOS TO JPALACIO;  
GRANT ALL ON PRODUCTOS TO JPALACIO;  
GRANT ALL ON PEDIDOS\_LOG TO JPALACIO;  
GRANT ALL ON CLIENTES\_LOG TO JPALACIO;  
GRANT ALL ON NOTIFICACIONES TO JPALACIO;

## Primeras funcionalidades

Para las siguientes secciones se presentan las estructuras solicitadas y sus respectivas pruebas de funcionamiento con imágenes del resultado obtenido.

1. Elaborar un procedimiento almacenado que permita la actualización del valor de un producto:

```
CREATE OR REPLACE PROCEDURE PROCEDIMIENTOACTUALIZARVALORPRODUCTO (
p_codigoproducto VARCHAR2, p_precioventa NUMBER) AS
BEGIN
    UPDATE PRODUCTOS
    SET PRECIOVENTA = p_precioventa
    WHERE CODIGOPRODUCTO = p_codigoproducto;
    COMMIT;
END PROCEDIMIENTOACTUALIZARVALORPRODUCTO;
/
```

-----  
-- Procedure to update the price of a product  
-----

GRANT EXECUTE ON PROCEDIMIENTOACTUALIZARVALORPRODUCTO TO  
JPALACIO;

-- Precio de venta original: 14.

EXECUTE PROCEDIMIENTOACTUALIZARVALORPRODUCTO('11679', 17);

-- Consulta de prueba de actualización de precio

```
SELECT
    "A1"."CODIGOPRODUCTO" "CODIGOPRODUCTO",
    "A1"."NOMBRE" "NOMBRE",
    "A1"."GAMA" "GAMA",
    "A1"."DIMENSIONES" "DIMENSIONES",
    "A1"."PROVEEDOR" "PROVEEDOR",
    "A1"."DESCRIPCION" "DESCRIPCION",
    "A1"."CANTIDADENSTOCK" "CANTIDADENSTOCK",
    "A1"."PRECIOVENTA" "PRECIOVENTA",
    "A1"."PRECIOPROVEEDOR" "PRECIOPROVEEDOR"
FROM
    "IS360015"."PRODUCTOS" "A1"
WHERE
    "A1"."CODIGOPRODUCTO" = '11679';
```

CODIGOPRODUCTO	NOMBRE	GAMA	DIMENSIONES	PROVEEDOR	DESCRIPCION	CANTIDADENSTOCK	PRECIOVENTA	PRECIOPROVEEDOR
1 11679	Sierra de Poda 400MM	Herramientas	0,258	HiperGarden Tools	(null)	15	14	11

CODIGOPRODUCTO	NOMBRE	GAMA	DIMENSI...	PROVEEDOR	DESCRIPCION	CANTIDADENSTOCK	PRECIOVENTA	PRECIOPROVEEDOR
1 11679	Sierra de Poda 400MM	Herramientas	0,258	HiperGarden Tools	(null)	15	17	11

Se observa el cambio de precio de 14 a 17 para el producto con código “11679”.

2. Elaborar un procedimiento almacenado que permita la actualización del inventario de un producto:

```
CREATE OR REPLACE PROCEDURE PROCEDIMIENTOACTUALIZARINVENTARIO (
    p_codigoproducto IN VARCHAR2,
    p_nuevo_stock IN NUMBER
) AS
    stock_actual NUMBER;
BEGIN
    SELECT CANTIDADENSTOCK INTO stock_actual
    FROM PRODUCTOS
    WHERE CODIGOPRODUCTO = p_codigoproducto;

    IF p_nuevo_stock < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'El nuevo stock no puede ser menor que cero');
    END IF;

    UPDATE PRODUCTOS
    SET CANTIDADENSTOCK = p_nuevo_stock
    WHERE CODIGOPRODUCTO = p_codigoproducto;
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'No se encontró ningún producto con el código
especificado');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END PROCEDIMIENTOACTUALIZARINVENTARIO;
/

-----
-- Procedure to update the stock
-----

GRANT EXECUTE ON PROCEDIMIENTOACTUALIZARINVENTARIO TO JPALACIO;

-- Cantidad de stock original: 15.
EXECUTE PROCEDIMIENTOACTUALIZARINVENTARIO('OR-248', 20);

-- Consulta para verificar que el inventario se actualizó correctamente
SELECT
    "A1"."CODIGOPRODUCTO" "CODIGOPRODUCTO",
    "A1"."NOMBRE"        "NOMBRE",
    "A1"."GAMA"          "GAMA",
    "A1"."DIMENSIONES"   "DIMENSIONES",
    "A1"."PROVEEDOR"     "PROVEEDOR",
    "A1"."DESCRIPCION"   "DESCRIPCION",
```

```

"A1"."CANTIDADENSTOCK" "CANTIDADENSTOCK",
"A1"."PRECIOVENTA" "PRECIOVENTA",
"A1"."PRECIOPROVEEDOR" "PRECIOPROVEEDOR"
FROM
"IS360015"."PRODUCTOS" "A1"
WHERE
"A1"."CODIGOPRODUCTO" = 'OR-248';

```

CODIGOPRODUCTO	NOMBRE	GAMA	DIMENSIONES	PROVEEDOR	DESCRIPCION	CANTIDADENSTOCK	PRECIOVENTA	PRECIOPROVEEDOR
1 OR-248	Washingtonia Robusta Ornamentales	60 - 70	Viveros EL OASIS	(null)		15	3	2
CODIGOPRODUCTO	NOMBRE	GAMA	DIMENSIONES	PROVEEDOR	DESCRIPCION	CANTIDADENSTOCK	PRECIOVENTA	PRECIOPROVEEDOR
1 OR-248	Washingtonia Robusta Ornamentales	60 - 70	Viveros EL OASIS	(null)		20	3	2

Se observa el cambio de cantidad en inventario de 15 a 20 para el producto de código “OR-248”.

- Elaborar un procedimiento almacenado que permita insertar información del detalle de un pedido:

```

CREATE OR REPLACE PROCEDURE PROCEDIMIENTODETALLES PEDIDO (
    p_cod_pedido IN NUMBER,
    p_cod_producto IN VARCHAR2,
    p_cantidad IN NUMBER,
    p_precio_unidad IN NUMBER,
    p_numero_linea IN NUMBER
) AS
BEGIN
    INSERT INTO DETALLE PEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
    PRECIOUNIDAD, NUMEROLINEA)
    VALUES (p_cod_pedido, p_cod_producto, p_cantidad, p_precio_unidad, p_numero_linea);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END PROCEDIMIENTODETALLES PEDIDO;
/

```

```

-----
-- Procedure to enter information of order details
-----

```

```
GRANT EXECUTE ON PROCEDIMIENTODETALLES PEDIDO TO JPALACIO;
```

```
-- Eliminación de tuplas
```

```
DELETE FROM DETALLE PEDIDOS WHERE CODIGOPEDIDO = 140;
```

```
DELETE FROM PEDIDOS WHERE CODIGOPEDIDO = 140;
```

```
-- Inserción de tuplas para no violar restricciones de llaves
```

```
INSERT INTO PEDIDOS (CODIGOPEDIDO, FECHAPEDIDO, FECHAESPERADA,
FECHAENTREGA, ESTADO, CODIGOCLIENTE) VALUES (140, TO_DATE('10/11/08',
'DD/MM/RR'), TO_DATE('10/12/08', 'DD/MM/RR'), TO_DATE('29/12/08', 'DD/MM/RR'),
'Rechazado', 38);
```

```
-- Ejecutar procedimiento para insertar detalle de pedido
EXECUTE PROCEDIMIENTODETALLESPEDIDO(140, 'OR-104', 18, 2, 2);
```

```
-- Consulta para verificar inserción
SELECT
    "A1"."CODIGOPEDIDO"    "CODIGOPEDIDO",
    "A1"."CODIGOPRODUCTO"  "CODIGOPRODUCTO",
    "A1"."CANTIDAD"        "CANTIDAD",
    "A1"."PRECIOUNIDAD"    "PRECIOUNIDAD",
    "A1"."NUMEROLINEA"     "NUMEROLINEA"
FROM
    "DETALLEPEDIDOS" "A1"
WHERE
    "A1"."CODIGOPEDIDO" = 140;
```

	CODIGOPEDIDO	CODIGOPRODUCTO	CANTIDAD	PRECIOUNIDAD	NUMEROLINEA
1	140	OR-104	18	1,7	2

Se observa la inserción de una tupla a través del procedimiento.

4. Elaborar una función que devuelva el valor de producto más vendido:

```
CREATE OR REPLACE FUNCTION FUNCIONPRODUCTOMASVENDIDO RETURN
VARCHAR2 AS
    v_producto_mas_vendido VARCHAR2(100);
BEGIN
    SELECT CODIGOPRODUCTO INTO v_producto_mas_vendido
    FROM (
        SELECT CODIGOPRODUCTO, SUM(CANTIDAD) AS TOTAL_VENDIDO
        FROM DETALLEPEDIDOS
        GROUP BY CODIGOPRODUCTO
        ORDER BY TOTAL_VENDIDO DESC
    )
    WHERE ROWNUM = 1;
    RETURN v_producto_mas_vendido;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END FUNCIONPRODUCTOMASVENDIDO;
/
```

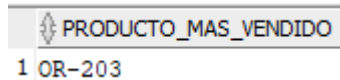
```
-----
-- Function to return most ordered product
```

-----

```
GRANT EXECUTE ON FUNCIONPRODUCTOMASVENDIDO TO JPALACIO;
```

-- El producto más vendido es OR-203 con una cantidad de 900000 (se alteró intencionalmente su valor)

```
SELECT FUNCIONPRODUCTOMASVENDIDO() AS PRODUCTO_MAS_VENDIDO FROM DUAL;
```



PRODUCTO_MAS_VENDIDO
1 OR-203

El producto más vendido es el del código “OR-203”.

5. Elaborar una función que devuelva la gama de productos más vendido:

```
CREATE OR REPLACE FUNCTION FUNCIONGAMAMASVENDIDA RETURN VARCHAR2  
AS
```

```
    v_gama_mas_vendida VARCHAR2(100);  
BEGIN  
    SELECT GAMA  
    INTO v_gama_mas_vendida  
    FROM (  
        SELECT GAMA, SUM(CANTIDAD) AS total_vendido  
        FROM DETALLEPEDIDOS dp  
        JOIN PRODUCTOS p ON dp.CODIGOPRODUCTO = p.CODIGOPRODUCTO  
        GROUP BY GAMA  
        ORDER BY SUM(CANTIDAD) DESC  
    ) WHERE ROWNUM = 1;  
    RETURN v_gama_mas_vendida;  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
        RETURN NULL;  
    WHEN OTHERS THEN  
        RAISE;  
END FUNCIONGAMAMASVENDIDA;  
/
```

-----

```
-- Function to return most ordered category
```


-----

```
GRANT EXECUTE ON FUNCIONGAMAMASVENDIDA TO JPALACIO;
```

-- La gama más vendida son las flores ornamentales a la que pertenece el producto OR-203

```
SELECT FUNCIONGAMAMASVENDIDA() AS GAMA_MAS_VENDIDA FROM DUAL;
```



	GAMA_MAS_VENDIDA
1	Ornamentales

La gama más vendida es “Ornamentales”, pues el producto más vendido, “OR-203” es de este tipo.

6. Elabore un (procedimiento almacenado / función) que permita el registro de nuevos medios de pago, debe verificar la existencia del “códigocliente” de manera previa:

```
CREATE OR REPLACE PROCEDURE PROCEDIMIENTOMEDIOPAGO (
  p_codigo_cliente IN NUMBER,
  p_forma_pago IN VARCHAR2
) AS
  v_id_transaccion PAGOS.IDTRANSACCION%TYPE;
BEGIN
  SELECT IDTRANSACCION
  INTO v_id_transaccion
  FROM PAGOS
  WHERE CODIGOCLIENTE = p_codigo_cliente
  AND FORMAPAGO = p_forma_pago;
  RAISE_APPLICATION_ERROR(-20002, 'Ya existe un registro para este cliente y forma de pago');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    SELECT NVL(MAX(SUBSTR(IDTRANSACCION, INSTR(IDTRANSACCION, '-', -1) + 1)),
0) + 1
    INTO v_id_transaccion
    FROM PAGOS
    WHERE CODIGOCLIENTE = p_codigo_cliente;
    INSERT INTO PAGOS (CODIGOCLIENTE, FORMAPAGO, IDTRANSACCION,
FECHAPAGO, CANTIDAD)
    VALUES (p_codigo_cliente, p_forma_pago, 'ak-std-' || LPAD(v_id_transaccion, 6, '0'),
SYSDATE, 0);
    COMMIT;
END PROCEDIMIENTOMEDIOPAGO;
/
```

```
-----
-- Procedure to add new pay form
-----
```

```
GRANT EXECUTE ON PROCEDIMIENTOMEDIOPAGO TO JPALACIO;
```

```
-- Eliminar resultado de procedimiento
```

```
DELETE FROM PAGOS WHERE CODIGOCLIENTE = 5 AND FORMAPAGO = 'DaviPlata';
```

```
-- Prueba con cliente inexistente y cliente correcto
```

```
EXECUTE PROCEDIMIENTOMEDIOPAGO(2, 'NEQUI');
```

```
EXECUTE PROCEDIMIENTOMEDIOPAGO(5, 'DaviPlata');
```

```
-- Consulta para verificar procedimiento
SELECT
    "CODIGOCLIENTE" AS "CODIGOCLIENTE",
    "FORMAPAGO" AS "FORMAPAGO",
    "IDTRANSACCION" AS "IDTRANSACCION",
    "FECHAPAGO" AS "FECHAPAGO",
    "CANTIDAD" AS "CANTIDAD"
FROM
    "PAGOS"
WHERE
    "CODIGOCLIENTE" = 5;
```

Error que empieza en la línea: 107 del comando :

```
BEGIN PROCEDIMIENTOMEDIOPAGO(2, 'NEQUI'); END;
```

Informe de error -

ORA-02291: restricción de integridad (IS360015.PAGOS\_CLIENTEFK) violada - clave principal no encontrada

ORA-06512: en "IS360015.PROCEDIMIENTOMEDIOPAGO", línea 19

ORA-01403: No se ha encontrado ningún dato

ORA-06512: en "IS360015.PROCEDIMIENTOMEDIOPAGO", línea 7

ORA-06512: en línea 1

02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"

\*Cause: A foreign key value has no matching primary key value.

\*Action: Delete the foreign key or add a matching primary key.

	CODIGOCLIENTE	FORMAPAGO	IDTRANSACCION	FECHAPAGO	CANTIDAD
1	5	Transferencia	ak-std-000011	18/01/06	23794
2	5	DaviPlata	ak-std-000012	19/05/24	0

Se observa que, al tratar de añadir una nueva forma de pago a un cliente con código inexistente, se genera un error, mientras que al hacerlo en uno registrado, se crea una nueva tupla con la cantidad 0.

- Elaborar un disparador que permita descontar las existencias de los productos cuando se haga un pedido:

```
CREATE OR REPLACE TRIGGER DISPARADORACTUALIZARINVENTARIO
AFTER INSERT ON DETALLEPEDIDOS
FOR EACH ROW
DECLARE
    v_estado_pedido VARCHAR2(50);
    stock_actual NUMBER;
BEGIN
    SELECT ESTADO INTO v_estado_pedido
    FROM PEDIDOS
    WHERE CODIGOPEDIDO = :NEW.CODIGOPEDIDO;

    IF v_estado_pedido = 'Entregado' THEN
        SELECT CANTIDADENSTOCK INTO stock_actual
        FROM PRODUCTOS
        WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;

        IF stock_actual < :NEW.CANTIDAD THEN
```

```

        RAISE_APPLICATION_ERROR(-20001, 'No hay suficiente stock para el producto ' ||
:NEW.CODIGOPRODUCTO);
    ELSE
        UPDATE PRODUCTOS
        SET CANTIDADENSTOCK = stock_actual - :NEW.CANTIDAD
        WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;
    END IF;
END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE;
END;
/

```

```

-----
-- Trigger to update the stock
-----

```

```

-- Eliminar las inserciones anteriores
DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO IN (130, 131);
DELETE FROM PEDIDOS WHERE CODIGOPEDIDO IN (130, 131);

-- Cantidades originales en stock de productos - OR-250: 15, OR-251: 1500
-- Resultados esperados - Inserción 1 no actualiza stock por estar Reprobado, Inserción 2 no es
posible por tener una cantidad superior de productos en pedido que en el stock, Inserción 3 actualiza
stock
INSERT INTO PEDIDOS (CODIGOPEDIDO, FECHAPEDIDO, FECHAESPERADA,
FECHAENTREGA, ESTADO, CODIGOCLIENTE) VALUES (130, TO_DATE('10/11/08',
'DD/MM/RR'), TO_DATE('10/12/08', 'DD/MM/RR'), TO_DATE('29/12/08', 'DD/MM/RR'),
'Entregado', 38);
INSERT INTO PEDIDOS (CODIGOPEDIDO, FECHAPEDIDO, FECHAESPERADA,
FECHAENTREGA, ESTADO, CODIGOCLIENTE) VALUES (131, TO_DATE('10/11/08',
'DD/MM/RR'), TO_DATE('10/12/08', 'DD/MM/RR'), TO_DATE('29/12/08', 'DD/MM/RR'),
'Entregado', 38);
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA) VALUES (130, 'OR-250', 18, 2, 2);
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA) VALUES (131, 'OR-251', 10, 2, 2);

-- Consulta para verificar que el inventario se actualizó correctamente. Tener en cuenta que al
ejecutar varias veces los insert, se resta también el stock el mismo número de veces para OR-251
SELECT
    "A1"."CODIGOPRODUCTO" "CODIGOPRODUCTO",
    "A1"."CANTIDADENSTOCK" "CANTIDADENSTOCK"
FROM
    "IS360015"."PRODUCTOS" "A1"
WHERE
    "A1"."CODIGOPRODUCTO" = 'OR-251'

```

OR "A1"."CODIGOPRODUCTO" = 'OR-250';

Error que empieza en la línea: 134 del comando -  
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA) VALUES (130, 'OR-250', 18, 2, 2)  
Error en la línea de comandos : 134 Columna : 17  
Informe de error -  
Error SQL: ORA-20001: La cantidad solicitada excede la cantidad disponible en inventario.  
ORA-06512: en "IS360015.DISPARADORVERIFICARCANTIDADINVENTARIO", línea 8  
ORA-04088: error durante la ejecución del disparador 'IS360015.DISPARADORVERIFICARCANTIDADINVENTARIO'

CODIGOPRODUCTO	CANTIDADENSTOCK
1 OR-250	15
2 OR-251	1480

Se observa que el disparador impide generar un pedido con una cantidad de un producto superior a la que se tiene en el inventario, por lo que no permite hacer el pedido de 18 unidades de “OR-250” que solo cuenta con 15 en el inventario, pero descuenta exitosamente 20 unidades de “OR-251” que inicialmente tenía 1480 unidades.

8. Elabore un disparador que evite que un empleado sea su propio jefe. Este se deberá activar en caso de que desee registrar a un empleado su jefe, ya sea al momento del registro de un nuevo empleado o en un proceso de actualización:

```
CREATE OR REPLACE TRIGGER DISPARADOREMPLEADOPROPIOJEFE
```

```
AFTER INSERT OR UPDATE ON EMPLEADOS
```

```
FOR EACH ROW
```

```
DECLARE
```

```
BEGIN
```

```
IF :NEW.CODIGOJEFE = :NEW.CODIGOEMPLEADO THEN
```

```
RAISE_APPLICATION_ERROR(-20001, 'No se puede asignar a un empleado como su propio jefe.');
```

```
END IF;
```

```
END;
```

```
/
```

```
-----  
-- Trigger to verify no employee is its own boss  
-----
```

```
-- Insert y update donde se intenta crear un empleado que es su propio jefe
```

```
INSERT INTO EMPLEADOS
```

```
(CODIGOEMPLEADO,NOMBRE,APELLIDO1,APELLIDO2,EXTENSION,EMAIL,CODIGOOFI  
CINA,CODIGOJEFE,PUESTO) VALUES
```

```
(32,'Juan','Perez','Gomez','2845','jperez@jardineria.es','TAL-ES',32,'Representante Ventas');
```

```
UPDATE EMPLEADOS SET CODIGOJEFE = 1 WHERE CODIGOEMPLEADO = 1;
```

```
-- Consulta para verificar que no se crearon
```

```
SELECT
```

```
E.CODIGOEMPLEADO AS "Codigo Empleado",
```

```
E.NOMBRE AS "Nombre",
```

```
E.APELLIDO1 AS "Apellido1",
```

```

E.APELLIDO2 AS "Apellido2",
E.EXTENSION AS "Extension",
E.EMAIL AS "Email",
E.CODIGOOFICINA AS "Codigo Oficina",
E.CODIGOJEFE AS "Codigo Jefe",
E.PUESTO AS "Puesto"
FROM
  EMPLEADOS E
WHERE
  E.CODIGOEMPLEADO = E.CODIGOJEFE;

```

```

Error que empieza en la línea: 152 del comando -
      INSERT INTO EMPLEADOS (CODIGOEMPLEADO,NOMBRE,APELLIDO1,APELLIDO2,EXTENSION,EMAIL,CODIGOOF
Error en la línea de comandos : 152 Columna : 17
Informe de error -
Error SQL: ORA-20001: No se puede asignar a un empleado como su propio jefe.
ORA-06512: en "IS360015.DISPARADOREMPLEADOPROPIOJEFE", línea 4
ORA-04088: error durante la ejecución del disparador 'IS360015.DISPARADOREMPLEADOPROPIOJEFE'

Error que empieza en la línea: 153 del comando -
      UPDATE EMPLEADOS SET CODIGOJEFE = 1 WHERE CODIGOEMPLEADO = 1
Error en la línea de comandos : 153 Columna : 12
Informe de error -
Error SQL: ORA-20001: No se puede asignar a un empleado como su propio jefe.
ORA-06512: en "IS360015.DISPARADOREMPLEADOPROPIOJEFE", línea 4
ORA-04088: error durante la ejecución del disparador 'IS360015.DISPARADOREMPLEADOPROPIOJEFE'

```

Codigo E...	Nombre	Apellido1	Apellido2	Extension	Email	Codigo Of...	Codigo Jefe	Puesto
-------------	--------	-----------	-----------	-----------	-------	--------------	-------------	--------

Se observa que el trigger impide insertar o actualizar a un empleado para que sea su propio jefe, por lo que las pruebas devuelven una tabla vacía al tratar de encontrar a un empleado que sea su propio jefe.

### Ampliando la base datos

- Elaborar unos disparadores que permitan realizar el proceso de auditoría a las tablas, clientes y pedidos, para los procesos de inserción, actualización y borrado (cabe resaltar que se crearon secuencias individuales para cada tabla por medio de la interfaz de Oracle):

```

CREATE OR REPLACE TRIGGER DISPARADORPEDIDOSLOG
AFTER INSERT OR UPDATE OR DELETE ON PEDIDOS
FOR EACH ROW
DECLARE
  V_ESTADO_ANTERIOR VARCHAR2(100);
  V_ESTADO_NUEVO VARCHAR2(100);
BEGIN
  IF INSERTING THEN
    INSERT INTO PEDIDOS_LOG (ID_AUDITORIA, CODIGOPEDIDO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR, ESTADO_ANTERIOR,
ESTADO_NUEVO)

```

```

VALUES (SECUENCIAPEDIDOSLOG.NEXTVAL, :NEW.CODIGOPEDIDO,
SYSTIMESTAMP, USER, NULL, :NEW.ESTADO);
ELSIF UPDATING THEN
    V_ESTADO_ANTERIOR := NVL(:OLD.ESTADO, 'NULL');
    V_ESTADO_NUEVO := NVL(:NEW.ESTADO, 'NULL');
    INSERT INTO PEDIDOS_LOG (ID_AUDITORIA, CODIGOPEDIDO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR, ESTADO_ANTERIOR,
ESTADO_NUEVO)
        VALUES (SECUENCIAPEDIDOSLOG.NEXTVAL, :OLD.CODIGOPEDIDO,
SYSTIMESTAMP, USER, V_ESTADO_ANTERIOR, V_ESTADO_NUEVO);
ELSIF DELETING THEN
    V_ESTADO_ANTERIOR := NVL(:OLD.ESTADO, 'NULL');
    INSERT INTO PEDIDOS_LOG (ID_AUDITORIA, CODIGOPEDIDO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR, ESTADO_ANTERIOR,
ESTADO_NUEVO)
        VALUES (SECUENCIAPEDIDOSLOG.NEXTVAL, :OLD.CODIGOPEDIDO,
SYSTIMESTAMP, USER, V_ESTADO_ANTERIOR, NULL);
    END IF;
END;
/

```

```

CREATE OR REPLACE TRIGGER DISPARADORCLIENTESLOG
AFTER INSERT OR UPDATE OR DELETE ON CLIENTES
FOR EACH ROW
DECLARE
    V_NOMBRE_ANTERIOR VARCHAR2(100);
    V_NOMBRE_NUEVO VARCHAR2(100);
    V_APELLIDO_ANTERIOR VARCHAR2(100);
    V_APELLIDO_NUEVO VARCHAR2(100);
BEGIN
    IF INSERTING THEN
        INSERT INTO CLIENTES_LOG (ID_AUDITORIA, CODIGOCLIENTE,
NOMBRE_ANTERIOR, NOMBRE_NUEVO, APELLIDO_ANTERIOR, APELLIDO_NUEVO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR)
            VALUES (SECUENCIACLIENTESLOG.NEXTVAL, :NEW.CODIGOCLIENTE, NULL,
:NEW.NOMBRECLIENTE, NULL, :NEW.NOMBRECONTACTO, SYSTIMESTAMP, USER);
    ELSIF UPDATING THEN
        V_NOMBRE_ANTERIOR := NVL(:OLD.NOMBRECLIENTE, 'NULL');
        V_NOMBRE_NUEVO := NVL(:NEW.NOMBRECLIENTE, 'NULL');
        V_APELLIDO_ANTERIOR := NVL(:OLD.NOMBRECONTACTO, 'NULL');
        V_APELLIDO_NUEVO := NVL(:NEW.NOMBRECONTACTO, 'NULL');
        INSERT INTO CLIENTES_LOG (ID_AUDITORIA, CODIGOCLIENTE,
NOMBRE_ANTERIOR, NOMBRE_NUEVO, APELLIDO_ANTERIOR, APELLIDO_NUEVO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR)
            VALUES (SECUENCIACLIENTESLOG.NEXTVAL, :OLD.CODIGOCLIENTE,
V_NOMBRE_ANTERIOR, V_NOMBRE_NUEVO, V_APELLIDO_ANTERIOR,
V_APELLIDO_NUEVO, SYSTIMESTAMP, USER);
    ELSIF DELETING THEN

```

```

V_NOMBRE_ANTERIOR := NVL(:OLD.NOMBRECLIENTE, 'NULL');
V_APELLIDO_ANTERIOR := NVL(:OLD.NOMBRECONTACTO, 'NULL');
INSERT INTO CLIENTES_LOG (ID_AUDITORIA, CODIGOCLIENTE,
NOMBRE_ANTERIOR, NOMBRE_NUEVO, APELLIDO_ANTERIOR, APELLIDO_NUEVO,
FECHA_MODIFICACION, USUARIO_MODIFICADOR)
VALUES (SECUENCIACLIENTESLOG.NEXTVAL, :OLD.CODIGOCLIENTE,
V_NOMBRE_ANTERIOR, NULL, V_APELLIDO_ANTERIOR, NULL, SYSTIMESTAMP,
USER);
END IF;
END;
/

```

```

-----
-- Trigger when creating, updating or deleting an user
-----

```

```

-- Inserción
INSERT INTO CLIENTES
(CODIGOCLIENTE,NOMBRECLIENTE,NOMBRECONTACTO,APELLIDOCONTACTO,TELEF
ONO,FAX,LINEADIRECCION1,LINEADIRECCION2,CIUDAD,REGION,PAIS,CODIGOPOSTA
L,CODIGOEMPLEADOREPVENTAS,LIMITECREDITO) VALUES (50,'DGPRODUCTIONS
GARDEN','Daniel G','GoldFish','5556901745','5556901746','False Street 52 2 A','Wall-e Avenue','San
Francisco','Wall-e Avenue','USA','24006',19,3000);

```

```

-- Actualización
UPDATE CLIENTES SET NOMBRECLIENTE = 'DagobertoSAS' WHERE CODIGOCLIENTE =
50;

```

```

-- Eliminación
DELETE FROM CLIENTES WHERE CODIGOCLIENTE = 50;

```

```

-- Consulta para ver el log
SELECT
"ID_AUDITORIA" AS ID_AUDITORIA,
"CODIGOCLIENTE" AS CODIGOCLIENTE,
"NOMBRE_ANTERIOR" AS NOMBRE_ANTERIOR,
"NOMBRE_NUEVO" AS NOMBRE_NUEVO,
"APELLIDO_ANTERIOR" AS APELLIDO_ANTERIOR,
"APELLIDO_NUEVO" AS APELLIDO_NUEVO,
"FECHA_MODIFICACION" AS FECHA_MODIFICACION,
"USUARIO_MODIFICADOR" AS USUARIO_MODIFICADOR
FROM
CLIENTES_LOG;

```

	ID_AUDITORIA	CODIGOCLIENTE	NOMBRE_ANTERIOR	NOMBRE_NUEVO	APELLIDO_ANTERIOR	APELLIDO_NUEVO	FECHA_MODIFICACION	USUARIO_MODIFICADOR
1	101	50	(null)	DGPRODUCTIONS GARDEN	(null)	Daniel G	19/05/24 05:23:35,370814000 PM	IS360015
2	102	50	DGPRODUCTIONS GARDEN	DagobertoSAS	Daniel G	Daniel G	19/05/24 05:23:35,455242000 PM	IS360015
3	103	50	DagobertoSAS	(null)	Daniel G	(null)	19/05/24 05:23:35,505211000 PM	IS360015

Se observa que en la primera tupla de la tabla de auditoría, por ser una inserción los valores anteriores se rellenan con null mientras que la fecha corresponde a la hora registrada en el sistema y el usuario quien ejecuta la consulta. En la segunda tupla, por ser update no hay campos vacíos y, en la tercera, eliminación, se no existen valores nuevos, por lo que se ponen en null. También destaca la múltiple declaración de variables en los disparadores para evitar errores de mutación.

```
-----
-- Trigger when creating, updating or deleting an order
-----
```

```
-- Inserción
```

```
INSERT INTO PEDIDOS (CODIGOPEDIDO, FECHAPEDIDO, FECHAESPERADA,
FECHAENTREGA, ESTADO, CODIGOCLIENTE) VALUES (155, SYSDATE, SYSDATE+5,
SYSDATE+10, 'Pendiente', 30);
```

```
-- Actualización
```

```
UPDATE PEDIDOS SET ESTADO = 'Entregado' WHERE CODIGOPEDIDO = 155;
```

```
-- Eliminación
```

```
DELETE FROM PEDIDOS WHERE CODIGOPEDIDO = 155;
```

```
-- Consulta para ver el log
```

```
SELECT
  "ID_AUDITORIA" AS ID_AUDITORIA,
  "CODIGOPEDIDO" AS CODIGOPEDIDO,
  "FECHA_MODIFICACION" AS FECHA_MODIFICACION,
  "USUARIO_MODIFICADOR" AS USUARIO_MODIFICADOR,
  "ESTADO_ANTERIOR" AS ESTADO_ANTERIOR,
  "ESTADO_NUEVO" AS ESTADO_NUEVO
FROM
  PEDIDOS_LOG;
```

	ID_AUDITORIA	CODIGOPEDIDO	FECHA_MODIFICACION	USUARIO_MODIFICADOR	ESTADO_ANTERIOR	ESTADO_NUEVO
1	125	140	19/05/24 04:12:22,500736000 PM	IS360015	(null)	Rechazado
2	126	130	19/05/24 04:29:16,648889000 PM	IS360015	(null)	Entregado
3	127	131	19/05/24 04:29:16,669133000 PM	IS360015	(null)	Entregado
4	128	155	19/05/24 05:23:35,753552000 PM	IS360015	(null)	Pendiente
5	129	155	19/05/24 05:23:35,790121000 PM	IS360015	Pendiente	Entregado
6	130	155	19/05/24 05:23:35,825666000 PM	IS360015	Entregado	(null)

Este resultado se comparta similar al anterior.

- Elaborar un procedimiento almacenado que permita actualizar el valor total de un pedido según el detalle de los pedidos, el código de pedido deberá pasar por parámetro de entrada:

```
CREATE OR REPLACE PROCEDURE PROCEDIMIENTOCALCULARTOTALPEDIDO (
  codigo_pedido_param IN DETALLEPEDIDOS.CODIGOPEDIDO%TYPE
)
AS
  v_valor_total_pedido DETALLEPEDIDOS.VALORTOTAL%TYPE := 0;
BEGIN
```



```

SELECT SUM(CANTIDAD * PRECIOUNIDAD) INTO v_valor_total_pedido
FROM DETALLEPEDIDOS
WHERE CODIGOPEDIDO = codigo_pedido_param;
UPDATE DETALLEPEDIDOS
SET VALORTOTAL = v_valor_total_pedido
WHERE CODIGOPEDIDO = codigo_pedido_param;
DBMS_OUTPUT.PUT_LINE('Valor total del pedido actualizado correctamente.');
```

EXCEPTION

```

  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No se encontraron detalles de pedido para el código de pedido
proporcionado.');
```

```

  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error al actualizar el valor total del pedido: ' || SQLERRM);
END PROCEDIMIENTO CALCULAR TOTAL PEDIDO;
/
```

```

-----
-- Procedure to update the total value of an order
-----
```

```

GRANT EXECUTE ON PROCEDIMIENTO ACTUALIZAR VALOR TOTAL PEDIDO TO
JPALACIO;
```

```

EXECUTE PROCEDIMIENTO ACTUALIZAR VALOR TOTAL PEDIDO(1);
```

```

-- Consulta para visualizar valor total del pedido
```

```

SELECT
  "CODIGOPEDIDO" AS codigo_pedido,
  "VALORTOTAL" AS valor_total,
  "CODIGOPRODUCTO" AS codigo_producto,
  "CANTIDAD" AS cantidad,
  "PRECIOUNIDAD" AS precio_unidad,
  "NUMEROLINEA" AS numero_linea
FROM
  DETALLEPEDIDOS
WHERE
  "CODIGOPEDIDO" = 1;
```

	⚡ CODIGO_PEDIDO	⚡ VALOR_TOTAL	⚡ CODIGO_PRODUCTO	⚡ CANTIDAD	⚡ PRECIO_UNIDAD	⚡ NUMERO_LINEA
1	1	1567	FR-67	10	70	3
2	1	1567	OR-127	40	3,4	1
3	1	1567	OR-141	25	3,4	2
4	1	1567	OR-241	15	16,15	4
5	1	1567	OR-99	23	11,9	5

Se observa que para pedidos con el mismo código, “1”, se calcula el valor total que es aplicado a todas las tuplas.

11. Elaborar un procedimiento almacenado que permita actualizar el valor de todos los pedidos a partir del procedimiento anterior:

```
CREATE OR REPLACE PROCEDURE PROCEDIMIENTOCALCULARTOTALPEDIDOSUMA
AS
    v_suma_total_pedidos NUMBER := 0;
BEGIN
    FOR detalle IN (SELECT DISTINCT CODIGOPEDIDO FROM DETALLEPEDIDOS)
    LOOP
        PROCEDIMIENTOCALCULARTOTALPEDIDO(detalle.CODIGOPEDIDO);
    END LOOP;
    SELECT SUM(VALORTOTAL) INTO v_suma_total_pedidos
    FROM DETALLEPEDIDOS;
    DBMS_OUTPUT.PUT_LINE('La suma total de los valores de todos los pedidos es: ' ||
v_suma_total_pedidos);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error al calcular la suma total de los valores de los pedidos: ' ||
SQLERRM);
END PROCEDIMIENTOCALCULARTOTALPEDIDOSUMA;
/
```

```
-----
-- Procedure to get the total value of all orders
-----
```

```
GRANT EXECUTE ON PROCEDIMIENTOCALCULARTOTALPEDIDOSUMA TO
JPALACIO;
```

```
EXECUTE PROCEDIMIENTOCALCULARTOTALPEDIDOSUMA;
```

```

Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
Valor total del pedido actualizado correctamente.
La suma total de los valores de todos los pedidos es: 36743476,68

```

Se observa que la suma total de los pedidos es 36743476,68 tras actualizar el valor total de cada pedido.

12. Elaborar un disparador que registre de manera automática el valor total de un pedido a partir del ingreso, actualización o eliminación en la tabla “detallepedido”.

```

CREATE OR REPLACE TRIGGER DISPARADORACTUALIZARVALORTOTALPEDIDO
AFTER INSERT OR UPDATE OR DELETE ON DETALLEPEDIDOS
FOR EACH ROW
DECLARE
    v_valor_total_pedido DETALLEPEDIDOS.VALORTOTAL%TYPE := 0;
BEGIN
    IF INSERTING THEN
        PROCEDIMIENTOCALCULARTOTALPEDIDO(:NEW.CODIGOPEDIDO);

    ELSIF UPDATING THEN
        PROCEDIMIENTOCALCULARTOTALPEDIDO(:NEW.CODIGOPEDIDO);

    ELSIF DELETING THEN
        PROCEDIMIENTOCALCULARTOTALPEDIDO(:OLD.CODIGOPEDIDO);
    END IF;
END;

```

```

-----
-- Trigger to automatically calculate total of an order
-----

```

```

DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO = 127 AND
CODIGOPRODUCTO = 'OR-150';

```

```

INSERT INTO DETALLEPEDIDOS
(CODIGOPEDIDO,CODIGOPRODUCTO,CANTIDAD,PRECIOUNIDAD,NUMEROLINEA)
values (127,'OR-150',18,2,2);

```

```

-- Consulta para verificar cálculo
SELECT
  "CODIGOPEDIDO" AS "Codigo_Pedido",
  "VALORTOTAL" AS "Valor_Total",
  "CODIGOPRODUCTO" AS "Codigo_Producto",
  "CANTIDAD" AS "Cantidad",
  "PRECIOUNIDAD" AS "Precio_Unidad",
  "NUMEROLINEA" AS "Numero_Linea"
FROM
  "DETALLEPEDIDOS"
WHERE
  "CODIGOPEDIDO" = 127;

```

Codigo_Pedido	Valor_Total	Codigo_Producto	Cantidad	Precio_Unidad	Numero_Linea
127		OR-150	18	1.7	2

El trigger "DISPARADORACTUALIZARVALORTOTALPEDIDO" opera en la tabla "DETALLEPEDIDOS" y se activa después de una inserción, actualización o eliminación. Su función es calcular el valor total de un pedido en función de los detalles de dicho pedido y actualizar automáticamente el campo correspondiente en la tabla de pedidos ("PEDIDOS"). Esto garantiza que el valor total del pedido siempre esté actualizado y refleje con precisión los cambios en los detalles del mismo.

13. Elabore un disparador que al momento de ingresar o actualizar un producto, verifique que el precio de venta de un producto sea superior un 10% del precio del “proveedor”

```

CREATE OR REPLACE TRIGGER DISPARADORVERIFICARPRECIOVENTA
BEFORE INSERT OR UPDATE ON PRODUCTOS
FOR EACH ROW
DECLARE
  v_precio_proveedor NUMBER;
BEGIN
  v_precio_proveedor := :NEW.PRECIOPROVEEDOR;

  IF :NEW.PRECIOVENTA <= (v_precio_proveedor * 1.1) THEN
    RAISE_APPLICATION_ERROR(-20001, 'El precio de venta debe ser al menos un 10% superior
al precio del proveedor.');
```

-----

-- Trigger to verify sale price is at least 10% greater than supplier price

-----

-- Precio de venta: 11, por lo que debe dar error

UPDATE PRODUCTOS SET PRECIOVENTA = 12 WHERE CODIGOPRODUCTO = 'OR-99';

```
Error starting at line : 275 in command -
UPDATE PRODUCTOS SET PRECIOVENTA = 12 WHERE CODIGOPRODUCTO = 'OR-99'
Error at Command Line : 275 Column : 8
Error report -|
SQL Error: ORA-20001: El precio de venta debe ser al menos un 10% superior al precio del proveedor.
ORA-06512: at "IS360019.DISPARADORVERIFICARPRECIOVENTA", line 7
ORA-04088: error during execution of trigger 'IS360019.DISPARADORVERIFICARPRECIOVENTA'
```

El trigger captura el precio del proveedor del nuevo registro o del registro actualizado. Luego, verifica si el precio de venta es menor o igual al 110% del precio del proveedor. Si esta condición no se cumple, el trigger levanta un error de aplicación con el código -20001 y un mensaje que indica que el precio de venta debe ser al menos un 10% superior al precio del proveedor.

El comentario proporcionado indica una prueba del trigger, donde intenta actualizar el precio de venta a 12 para un producto identificado como 'OR-99'. Como el precio de venta (12) no es al menos un 10% mayor que el precio del proveedor, el trigger debería arrojar un error durante esta actualización.

14. Elabore un disparador que verifique al momento de realizar los pedidos, el inventario no quede vacío

```
CREATE OR REPLACE TRIGGER DISPARADORVERIFICARCANTIDADINVENTARIO
BEFORE INSERT OR UPDATE ON DETALLEPEDIDOS
FOR EACH ROW
DECLARE
    v_cantidad_disponible NUMBER;
BEGIN
    SELECT CANTIDADENSTOCK INTO v_cantidad_disponible
    FROM PRODUCTOS
    WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;
    IF :NEW.CANTIDAD > v_cantidad_disponible THEN
        RAISE_APPLICATION_ERROR(-20001, 'La cantidad solicitada excede la cantidad disponible
en inventario.');
```

-----

-- Trigger to verify non-Empty Inventory

-----

-- Excede el stock

DELETE FROM PRODUCTOS WHERE CODIGOPRODUCTO = 'OR-137';

```
INSERT INTO PRODUCTOS (CODIGOPRODUCTO, NOMBRE, GAMA, DIMENSIONES,
PROVEEDOR, CANTIDADENSTOCK, PRECIOVENTA, PRECIOPROVEEDOR)
VALUES ('OR-137', 'ROSAL TREPADOR', 'Ornamentales', NULL, 'Viveros EL OASIS', 100, 4,
3);
```

```
-- No excede el stock
DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO = 1;
```

```
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA)
VALUES (1, 'OR-137', 150, 4, 1);
```

```
Error starting at line : 290 in command -
      INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA)
      VALUES (1, 'OR-137', 150, 4, 1)
Error at Command Line : 290 Column : 17
Error report -
SQL Error: ORA-20001: La cantidad solicitada excede la cantidad disponible en inventario.
ORA-06512: at "IS360019.DISPARADORVERIFICARCANTIDADINVENTARIO", line 8
ORA-04088: error during execution of trigger 'IS360019.DISPARADORVERIFICARCANTIDADINVENTARIO'
```

El disparador "DISPARADORVERIFICARINVENTARIO" opera antes de una inserción o actualización en la tabla "DETALLEPEDIDOS". Su objetivo es garantizar que la cantidad de productos solicitada en un pedido no exceda la cantidad disponible en el inventario. Si la cantidad solicitada supera la cantidad en stock del producto, el disparador genera un error para evitar transacciones que agoten el inventario. En el ejemplo proporcionado, se muestra cómo el disparador funciona al tratar con un producto que no tiene suficiente stock ("OR-137") y otro que sí lo tiene.

15. Ampliar la base de datos en "gamaproductos" agregando un atributo "descuento" el cual será un descuento que se realizará a cualquier compra de esa gama de productos.

```
-- DDL for Table GAMASPRODUCTOS
-----

CREATE TABLE "GAMASPRODUCTOS"
(   "GAMA" VARCHAR2(50 BYTE),
    "DESCUENTO" NUMBER(5,2),
    "DESCRIPCIONTEXTO" CLOB,
    "DESCRIPCIONHTML" CLOB,
    "IMAGEN" BLOB
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
```

16. Elaborar un disparador que se active después de una acción de manipulación (inserción, actualización) sobre la tabla de "detallepedido" el cual deberá verificar si el producto que se

está facturando se encuentra en una gama con descuento, si en efecto tiene descuento entonces automáticamente ha de aplicar al momento de facturar.

```
CREATE OR REPLACE TRIGGER DISPARADORAPLICARDESCUENTO
BEFORE INSERT OR UPDATE ON DETALLEPEDIDOS
FOR EACH ROW
```

```
DECLARE
```

```
    v_descuento NUMBER := 0;
```

```
    v_gama VARCHAR2(50);
```

```
BEGIN
```

```
    -- Obtener la gama del producto
```

```
    SELECT GAMA INTO v_gama
```

```
    FROM PRODUCTOS
```

```
    WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;
```

```
    -- Obtener el descuento de la gama del producto si existe
```

```
    SELECT DESCUENTO INTO v_descuento
```

```
    FROM GAMASPRODUCTOS
```

```
    WHERE GAMA = v_gama;
```

```
    -- Aplicar el descuento si existe
```

```
    IF v_descuento IS NOT NULL THEN
```

```
        :NEW.PRECIOUNIDAD := :NEW.PRECIOUNIDAD - (:NEW.PRECIOUNIDAD *
(v_descuento / 100));
```

```
    END IF;
```

```
END;
```

```
-----
-- Trigger to verify if the product is in a range with discount
-----
```

```
DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO = 1 AND
CODIGOPRODUCTO = 'OR-137';
```

```
INSERT INTO PRODUCTOS (CODIGOPRODUCTO, NOMBRE, GAMA, DIMENSIONES,
PROVEEDOR, CANTIDADENSTOCK, PRECIOVENTA, PRECIOPROVEEDOR)
```

```
VALUES ('OR-137', 'ROSAL TREPADOR', 'Ornamentales', NULL, 'Viveros EL OASIS', 100, 4,
3);
```

```
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA)
```

```
VALUES (1, 'OR-137', 10, 4, 1);
```

```
    -- Para este producto hay un descuento del 15 pq es ornamentales el precio por unidad es de 4 por lo
tanto el precio unidad es de 3.4 con descuento gracias al trigger.
```

```
SELECT
```

```
    CODIGOPEDIDO AS Codigo_Pedido,
```

```
    CODIGOPRODUCTO AS Codigo_Producto,
```

```
    CANTIDAD AS Cantidad,
```

```

PRECIOUNIDAD AS Precio_Unidad,
NUMEROLINEA AS Numero_Linea
FROM
DETALLEPEDIDOS
WHERE
CODIGOPEDIDO = 1;

```

	CODIGO_PEDIDO	CODIGO_PRODUCTO	CANTIDAD	PRECIO_UNIDAD	NUMERO_LINEA
1	1	OR-137	10	3.4	1

Obtiene la gama del producto que se está insertando o actualizando. Luego, busca si hay un descuento asociado a esa gama en la tabla "GAMASPRODUCTOS". Si encuentra un descuento, lo aplica al precio unitario del producto. Si no hay descuento disponible para esa gama, el precio unitario permanece sin cambios. En el ejemplo proporcionado, se muestra cómo el disparador modifica el precio unitario de un producto ("OR-137") basado en el descuento asociado a su gama en la tabla "GAMASPRODUCTOS".

17. Se desea mantener un control estricto de la cantidad disponible de productos en inventario, el disparador deberá garantizar que la cantidad solicitada por el pedido no exceda la cantidad disponible en el inventario. Construir un disparador que permita actualizar automáticamente la cantidad disponible en el inventario a partir del registro de un nuevo pedido

```

CREATE OR REPLACE TRIGGER DISPARADOR_ACTUALIZAR_INVENTARIO
AFTER INSERT ON DETALLEPEDIDOS
FOR EACH ROW
DECLARE
    v_cantidad_disponible NUMBER;
BEGIN

    UPDATE PRODUCTOS
    SET CANTIDADENSTOCK = CANTIDADENSTOCK - :NEW.CANTIDAD
    WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;

END;

```

```

-----
-- Trigger to verify strict control of the available quantity of products
-----

DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO = 1 AND
CODIGOPRODUCTO = 'OR-137';

-- Falla por insuficiente inventario
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA)
VALUES (1, 'OR-137', 150, 4, 1);

```



```

Error starting at line : 323 in command -
      INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA)
      VALUES (1, 'OR-137', 150, 4, 1)
Error at Command Line : 323 Column : 17
Error report -
SQL Error: ORA-20001: La cantidad solicitada excede la cantidad disponible en inventario.
ORA-06512: at "IS360019.DISPARADORVERIFICARCANTIDADINVENTARIO", line 8
ORA-04088: error during execution of trigger 'IS360019.DISPARADORVERIFICARCANTIDADINVENTARIO'

```

-----

-- Trigger to verify automatically update quantity available in inventory

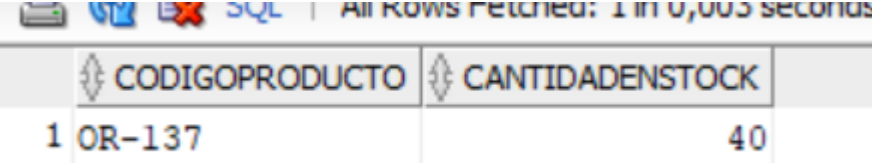
-----

```

DELETE FROM DETALLEPEDIDOS WHERE CODIGOPEDIDO = 1 AND
CODIGOPRODUCTO = 'OR-137';
INSERT INTO DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD,
PRECIOUNIDAD, NUMEROLINEA)
VALUES (1, 'OR-137', 20, 4, 1);

SELECT CODIGOPRODUCTO, CANTIDADENSTOCK
FROM PRODUCTOS
WHERE CODIGOPRODUCTO = 'OR-137';

```



	CODIGOPRODUCTO	CANTIDADENSTOCK
1	OR-137	40

18. Ahora se necesita dotar la base de datos de una capacidad de notificaciones en caso de presentarse alguna novedad, por lo cual se deberá realizar una nueva tabla llamada “notificaciones” compuesta por un identificador, fecha, destinatario y mensaje

```

-----
--DDL FOR TABLE NOTIFICACIONES
-----
CREATE TABLE "NOTIFICACIONES"
(
  "ID_NOTIFICACION" NUMBER GENERATED BY DEFAULT AS IDENTITY,
  "FECHA" DATE DEFAULT SYSDATE,
  "DESTINATARIO" NUMBER(*,0),
  "MENSAJE" VARCHAR2(50 BYTE)
)
SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT);

```

19. Construir un disparador que en caso de que un cliente registre un pago y este supere el límite de crédito realice un registro de una notificación de “ha superado el límite de crédito” en la tabla de notificaciones incluyendo como destinatario el código del empleado que tiene

asignado el cliente y la fecha en la cual se hace el registro de manera automática (capturada del sistema).

```
CREATE OR REPLACE TRIGGER TRIGGER_SUPERAR_LIMITE_CREDITO
AFTER INSERT ON PAGOS
FOR EACH ROW
DECLARE
    v_limite_credito NUMBER;
BEGIN
    -- Attempt to select the credit limit for the given client
    BEGIN
        SELECT LIMITECREDITO INTO v_limite_credito
        FROM CLIENTES
        WHERE CODIGOCLIENTE = :NEW.CODIGOCLIENTE;
        DBMS_OUTPUT.PUT_LINE('Limite de credito: ' || v_limite_credito);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20002, 'El cliente con código ' ||
:NEW.CODIGOCLIENTE || ' no existe.');
```

END;

```

    -- Check if the payment exceeds the credit limit
    IF :NEW.CANTIDAD > v_limite_credito THEN
        BEGIN
            INSERT INTO NOTIFICACIONES (ID_NOTIFICACION, FECHA, DESTINATARIO,
MENSAJE)
            VALUES (SEQ_NOTIFICACION.NEXTVAL, SYSDATE, :NEW.CODIGOCLIENTE ,
                'El cliente con código ' || :NEW.CODIGOCLIENTE || ' ha superado su límite de crédito al
realizar un pago.');
```

DBMS\_OUTPUT.PUT\_LINE('El cliente ha superado su límite de crédito.');

```

        EXCEPTION
            WHEN OTHERS THEN
                RAISE_APPLICATION_ERROR(-20003, 'Error al insertar en NOTIFICACIONES: ' ||
SQLERRM);
        END;
    ELSE
        DBMS_OUTPUT.PUT_LINE('El cliente no ha superado su límite de crédito.');
```

END IF;

END;

-----

-- Trigger to verify that in case a client registers a payment and this exceed the credit limit make a registration of a notification

-----

```
DELETE FROM PAGOS WHERE CODIGOCLIENTE = 1;
```

```

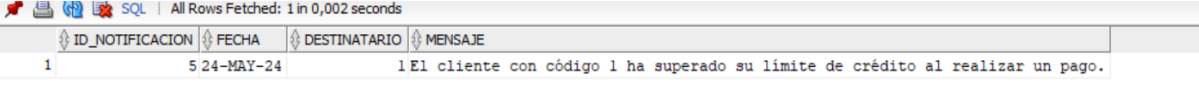
INSERT INTO PAGOS (CODIGOCLIENTE, FORMAPAGO, IDTRANSACCION,
FECHAPAGO, CANTIDAD)
VALUES (1, 'Efectivo', 'ak-std-0000051', SYSDATE, 100000);

```

```

SELECT ID_NOTIFICACION,
       FECHA,
       DESTINATARIO,
       MENSAJE
FROM NOTIFICACIONES;

```



ID_NOTIFICACION	FECHA	DESTINATARIO	MENSAJE
1	5/24-MAY-24	1	El cliente con código 1 ha superado su límite de crédito al realizar un pago.

Obtiene el límite de crédito del cliente asociado con el pago insertado. Si la cantidad del pago excede el límite de crédito, se registra una notificación en la tabla "NOTIFICACIONES" indicando que el cliente ha superado su límite de crédito al realizar el pago. Además, se emite un mensaje indicando que el cliente ha superado su límite de crédito.

Si el cliente no existe, se imprime un mensaje indicando que el cliente no existe. Si se produce algún otro error durante el procesamiento del pago, se imprime un mensaje de error con detalles sobre el error.