

11078/11278 – BD II/BDD – Primer Examen Parcial

1. Dadas las siguientes tablas, de una implementación de tipo B (todas las FK's son NO ACTION):
FABRICA(CUIT,RAZON) PK(CUIT)
PRODUCTO(CODIGO,DESCR,PRECIO,STOCK) PK(CODIGO)
FACTURA(NRO,FECHA,MONTO) PK(NRO)
BULTO(CUIT,CODBAR,CODIGO,CANTIDAD,PRECIO NULL,NRO NULL)
PK(CUIT,CODBAR) FK(CUIT,FABRICA) FK(CODIGO,PRODUCTO) FK(NRO,FACTURA)

Implemente la siguiente regla semántica:

-Cuando se da de alta un bulto, no hay precio de venta ni número de factura, se debe sumar cantidad al stock del producto

-Cuando se factura un bulto, debe existir número de factura y precio de venta (no menor a producto.precio), se debe restar cantidad del stock del producto, se debe sumar precio de venta * cantidad a monto factura. **Restricciones:** No se permiten cambios en PK de BULTO. No se permiten cambios en CODIGO, CANTIDAD. Si PRECIO es NULL también lo es NRO y viceversa. No puedo borrar un BULTO facturado. Puede agregar/quitar tablas, campos, datos de control, etc.

a) Escriba un script pl/sql con las instrucciones de DDL para crear los datos de control, tablas, etc.

b) complete el cuadro con el pseudo-código para cada evento en cada tabla. Los nombres de los triggers son TRG_<TIPO EVENTO><TABLA> donde <TIPO EVENTO> puede ser {BI,BD,BU,AI,AD,AU} B=Before, A=After, I=Insert, D=Delete, U=Update.

c) Escriba el código PL/SQL de los triggers que describió en el cuadro anterior

2. Dadas la tablas del punto 1: implementar un **select procedure** (P_VENTA) que reciba como parámetros de entrada un año de 4 dígitos (Ej: 2026) y tenga como parámetros de salida ANIO, MES, CODIGO, DESCR, CANTIDAD, TOTAL; en donde CANTIDAD se refiere la cantidad total en unidades vendidas, TOTAL es la cantidad vendida en pesos. BULTO.PRECIO es precio unitario. Los productos que no tuvieron ventas en un año-mes determinado, no deben estar.

3. Dada la tabla: EMPLEADO(LEGAJO,APENOM,SU_JEFE_ES NULL) PK(LEGAJO) FK(SU_JEFE_ES,EMPLEADO) implemente un select procedure no recursivo P_JEFE que reciba como parámetro de entrada un legajo y emita como salida LEGAJO,APENOM con la lista de todos sus superiores hasta llegar al gerente general. P_JEFE no puede quedar en loop. Un empleado no puede tener más de 10 niveles jerárquicos.

4. Describa 3 anotaciones que se utilizan en JPA, indicando en qué ámbito se usa (clase, atributo, método), cuáles son sus parámetros y para qué sirve.

11078/11278 – BD II/BDD – Primer Examen Parcial

1. Dadas las siguientes tablas, de una implementación de tipo B (todas las FK's son NO ACTION):
FABRICA(CUIT,RAZON) PK(CUIT)
PRODUCTO(CODIGO,DESCR,PRECIO,STOCK) PK(CODIGO)
FACTURA(NRO,FECHA,MONTO) PK(NRO)
BULTO(CUIT,CODBAR,CODIGO,CANTIDAD,PRECIO NULL,NRO NULL)
PK(CUIT,CODBAR) FK(CUIT,FABRICA) FK(CODIGO,PRODUCTO) FK(NRO,FACTURA)

Implemente la siguiente regla semántica:

-Cuando se da de alta un bulto, no hay precio de venta ni número de factura, se debe sumar cantidad al stock del producto

-Cuando se factura un bulto, debe existir número de factura y precio de venta (no menor a producto.precio), se debe restar cantidad del stock del producto, se debe sumar precio de venta * cantidad a monto factura. **Restricciones:** No se permiten cambios en PK de BULTO. No se permiten cambios en CODIGO, CANTIDAD. Si PRECIO es NULL también lo es NRO y viceversa. No puedo borrar un BULTO facturado. Puede agregar/quitar tablas, campos, datos de control, etc.

a) Escriba un script pl/sql con las instrucciones de DDL para crear los datos de control, tablas, etc.

b) complete el cuadro con el pseudo-código para cada evento en cada tabla. Los nombres de los triggers son TRG_<TIPO EVENTO><TABLA> donde <TIPO EVENTO> puede ser {BI,BD,BU,AI,AD,AU} B=Before, A=After, I=Insert, D=Delete, U=Update.

c) Escriba el código PL/SQL de los triggers que describió en el cuadro anterior

2. Dadas la tablas del punto 1: implementar un **select procedure** (P_VENTA) que reciba como parámetros de entrada un año de 4 dígitos (Ej: 2026) y tenga como parámetros de salida ANIO, MES, CODIGO, DESCR, CANTIDAD, TOTAL; en donde CANTIDAD se refiere la cantidad total en unidades vendidas, TOTAL es la cantidad vendida en pesos. BULTO.PRECIO es precio unitario. Los productos que no tuvieron ventas en un año-mes determinado, no deben estar.

3. Dada la tabla: EMPLEADO(LEGAJO,APENOM,SU_JEFE_ES NULL) PK(LEGAJO) FK(SU_JEFE_ES,EMPLEADO) implemente un select procedure no recursivo P_JEFE que reciba como parámetro de entrada un legajo y emita como salida LEGAJO,APENOM con la lista de todos sus superiores hasta llegar al gerente general. P_JEFE no puede quedar en loop. Un empleado no puede tener más de 10 niveles jerárquicos.

4. Describa 3 anotaciones que se utilizan en JPA, indicando en qué ámbito se usa (clase, atributo, método), cuáles son sus parámetros y para qué sirve.

SOLUCION POSIBLE

1. Dadas las siguientes tablas, de una implementación de tipo B (todas las FK's son NO ACTION):
 FABRICA(CUIT,RAZON) PK(CUIT)
 PRODUCTO(CODIGO,DESCR,PRECIO,STOCK) PK(CODIGO)
 FACTURA(NRO,FECHA,MONTO) PK(NRO)
 BULTO(CUIT,CODBAR,CODIGO,CANTIDAD,PRECIO NULL,NRO NULL)
 PK(CUIT,CODBAR) FK(CUIT,FABRICA) FK(CODIGO,PRODUCTO) FK(NRO,FACTURA)

Implemente la siguiente regla semántica:

-Cuando se da de alta un bulto, no hay precio de venta ni número de factura, se debe sumar cantidad al stock del producto
 -Cuando se factura un bulto, debe existir número de factura y precio de venta (no menor a producto.precio), se debe restar cantidad del stock del producto, se debe sumar precio de venta * cantidad a monto factura. **Restricciones:** No se permiten cambios en PK de BULTO. No se permiten cambios en CODIGO, CANTIDAD. Si PRECIO es NULL también lo es NRO y viceversa. No puedo borrar un BULTO facturado. Puede agregar/quitar tablas, campos, datos de control, etc.

a) Escriba un script pl/sql con las instrucciones de DDL para crear los datos de control, tablas, etc.

CREATE EXCEPTION EX_ERROR 'Error' ;

b) complete el cuadro con el pseudo-código para cada evento en cada tabla. Los nombres de los triggers son TRG_<TIPO EVENTO><TABLA> donde <TIPO EVENTO> puede ser {BI,BD,BU,AI,AD,AU} B=Before, A=After, I=Insert, D=Delete, U=Update.

EV.		TABLAS		
		BULTO		
I	B	PRECIO=NULL NRO=NULL Si CANTIDAD <= 0 -> Error!		
	A	PRODUCTO.STOCK = PRODUCTO.STOCK + CANTIDAD		
U	B	Si cambia PK -> Error! Si cambia CODIGO -> Error! Si cambia CANTIDAD -> Error! Si PRECIO <> NULL Si NRO = NULL -> Error! Si PRECIO < PRODUCTO.PRECIO -> Error! FinSi Si NRO <> NULL y PRECIO = NULL -> Error!		
	A	Si old.NRO <> NULL y new.NRO <> NULL Resto de factura old.nro y sumo en factura new.nro FinSi Si old.NRO = NULL y new.NRO <> NULL Sumo en factura new.nro Resto stock FinSi Si old.NRO <> NULL y new.NRO = NULL Resto de factura old.nro Sumo stock FinSi		
D	B	Si old.NRO <> NULL o old.PRECIO <> NULL -> Error!		
	A	PRODUCTO.STOCK = PRODUCTO.STOCK - CANTIDAD		

c) Escriba el código PL/SQL de los triggers que describió en el cuadro anterior

--BI

--PRECIO=NULL

--NRO=NULL

--Si CANTIDAD <= 0 -> Error!

CREATE TRIGGER TRG_BIBULTO FOR BULTO BEFORE INSERT AS

BEGIN

NEW.PRECIO=NULL;

NEW.NRO=NULL;

IF (NEW.CANTIDAD <= 0) THEN EXCEPTION EX_ERROR 'MAL CANTIDAD' ;

END

--AI

-- PRODUCTO.STOCK = PRODUCTO.STOCK + CANTIDAD

CREATE TRIGGER TRG_AIBULTO FOR BULTO AFTER INSERT AS

BEGIN

UPDATE PRODUCTO SET STOCK = STOCK + NEW.CANTIDAD

WHERE CODIGO = NEW.CODIGO;

END

--BU

--Si cambia PK -> Error!

--Si cambia CODIGO -> Error!

--Si cambia CANTIDAD -> Error!

--Si PRECIO <> NULL

-- Si NRO = NULL -> Error!

-- Si PRECIO < PRODUCTO.PRECIO -> Error!

--FinSi

CREATE TRIGGER TRG_BUBULTO FOR BULTO BEFORE UPDATE AS

BEGIN

IF (NEW.CUIT <> OLD.CUIT OR NEW.CODBAR <> OLD.CODBAR) THEN

EXCEPTION EX_ERROR 'NO SE PUEDE CAMBIAR PK DE BULTO' ;

IF (OLD.CODIGO <> NEW.CODIGO THEN

EXCEPTION EX_ERROR 'NO SE PUEDE CAMBIAR CODIGO EN BULTO' ;

IF (OLD.CANTIDAD <> NEW.CANTIDAD THEN

EXCEPTION EX_ERROR 'NO SE PUEDE CAMBIAR CANTIDAD EN BULTO' ;

IF (NEW.PRECIO IS NOT NULL) THEN BEGIN

IF (NEW.NRO IS NULL) THEN

EXCEPTION EX_ERROR 'INDICA PRECIO PERO NO INDICA NRO' ;

IF (NEW.PRECIO < (SELECT PRECIO

FROM PRODUCTO

WHERE CODIGO = NEW.CODIGO)) THEN

EXCEPTION EX_ERROR 'PRECIO EQUIVOCADO' ;

END

END

--AU

```
--Si old.NRO <> NULL y new.NRO <> NULL
--  Resto de factura old.nro y sumo en factura new.nro
--FinSi
--Si old.NRO = NULL y new.NRO <> NULL
--  Sumo en factura new.nro
--  Resto stock
--FinSi
--Si old.NRO <> NULL y new.NRO = NULL
--  Resto de factura old.nro
--  Sumo stock
--FinSi
CREATE TRIGGER TRG_AUBULTO FOR BULTO AFTER UPDATE AS
BEGIN
    IF (OLD.NRO IS NOT NULL AND NEW.NRO IS NOT NULL) THEN BEGIN
        UPDATE FACTURA SET MONTO = MONTO - ( OLD.CANTIDAD*OLD.PRECIO )
        WHERE NRO = OLD.NRO;
        UPDATE FACTURA SET MONTO = MONTO + ( NEW.CANTIDAD*NEW.PRECIO )
        WHERE NRO = NEW.NRO;
    END
    IF (OLD.NRO IS NULL AND NEW.NRO IS NOT NULL) THEN BEGIN
        UPDATE FACTURA SET MONTO = MONTO + ( NEW.CANTIDAD*NEW.PRECIO )
        WHERE NRO = NEW.NRO;
        UPDATE PRODUCTO SET STOCK = STOCK - NEW.CANTIDAD
        WHERE CODIGO = NEW.CODIGO;
    END
    IF (OLD.NRO IS NOT NULL AND NEW.NRO IS NULL) THEN BEGIN
        UPDATE FACTURA SET MONTO = MONTO - ( OLD.CANTIDAD*OLD.PRECIO )
        WHERE NRO = OLD.NRO;
        UPDATE PRODUCTO SET STOCK = STOCK + NEW.CANTIDAD
        WHERE CODIGO = OLD.CODIGO;
    END
END
END
```

--BD

```
--Si old.NRO <> NULL o old.PRECIO <> NULL -> Error!
CREATE TRIGGER TRG_BDBULTO FOR BULTO BEFORE DELETE AS
BEGIN
    IF ( OLD.NRO IS NOT NULL OR OLD.PRECIO IS NOT NULL ) THEN
        EXCEPTION EX_ERROR 'NO SE PUEDE BORRAR BULTO!';
    END
END
```

--AD

```
-- PRODUCTO.STOCK = PRODUCTO.STOCK - CANTIDAD
CREATE TRIGGER TRG_BDBULTO FOR BULTO BEFORE DELETE AS
BEGIN
    UPDATE PRODUCTO SET STOCK = STOCK - OLD.CANTIDAD
    WHERE CODIGO = OLD.CODIGO;
END
```

2. Dadas la tablas del punto 1: implementar un **select procedure** (P_VENTA) que reciba como parámetros de entrada un año de 4 dígitos (Ej: 2026) y tenga como parámetros de salida ANIO, MES, CODIGO, DESCR, CANTIDAD, TOTAL; en donde CANTIDAD se refiere la cantidad total en unidades vendidas, TOTAL es la cantidad vendida en pesos. BULTO.PRECIO es precio unitario.

```
FABRICA(CUIT,RAZON) PK(CUIT)
PRODUCTO(CODIGO,DESCR,PRECIO,STOCK) PK(CODIGO)
FACTURA(NRO,FECHA,MONTO) PK(NRO)
BULTO(CUIT,CODBAR,CODIGO,CANTIDAD,PRECIO NULL,NRO NULL)
    PK(CUIT,CODBAR) FK(CUIT,FABRICA) FK(CODIGO,PRODUCTO) FK(NRO,FACTURA)
```

```
CREATE PROCEDURE P_VENTA (PANIO INTEGER)
RETURNS (
    ANIO INTEGER,
    MES INTEGER,
    CODIGO TYPE OF COLUMN PRODUCTO.CODIGO,
    DESCR TYPE OF COLUMN PRODUCTO.DESCR,
    CANTIDAD TYPE OF COLUMN BULTO.CANTIDAD,
    TOTAL TYPE OF COLUMN FACTURA.MONTO
) AS
BEGIN
    ANIO = PANIO;
    MES=1;
    WHILE (MES<=12) DO BEGIN
        FOR SELECT P.CODIGO,P.DESCR FROM PRODUCTO P
            WHERE EXISTS (
                SELECT * FROM BULTO B INNER JOIN FACTURA F ON B.NRO=F.NRO
                WHERE B.CODIGO = P.CODIGO AND
                    EXTRACT(YEAR FROM F.FECHA) = :ANIO AND
                    EXTRACT(MONTH FROM F.FECHA) = :MES
            )
        INTO :CODIGO, :DESCR DO BEGIN
            SELECT SUM(B.CANTIDAD) ,SUM(B.CANTIDAD*B.PRECIO)
            FROM BULTO B INNER JOIN FACTURA F ON B.NRO=F.NRO
            WHERE B.CODIGO = :CODIGO AND
                EXTRACT(YEAR FROM F.FECHA) = :ANIO AND
                EXTRACT(MONTH FROM F.FECHA) = :MES
            INTO :CANTIDAD, :TOTAL;
            SUSPEND;
        END
        MES=MES+1;
    END
END
```

3. Dada la tabla: EMPLEADO(LEGAJO,APENOM,SU_JEFE_ES NULL) PK(LEGAJO) FK(SU_JEFE_ES,EMPLEADO) implemente un select procedure no recursivo P_JEFE que reciba como parámetro de entrada un legajo y emita como salida LEGAJO,APENOM con la lista de todos sus superiores hasta llegar al gerente general. P_JEFE no puede quedar en loop. Un empleado no puede tener más de 10 niveles jerárquicos.

Código para analizar: los select into no cargan valores nulos en legajo y apenom, por eso se usó ROW_COUNT, código probado en /var/lib/firebird/3.0/data/db1314.fdb

```
CREATE PROCEDURE P_JEFE (
    PLEGAJO TYPE OF COLUMN EMPLEADO.LEGAJO
) RETURNS (
    LEGAJO TYPE OF COLUMN EMPLEADO.LEGAJO,
    APENOM TYPE OF COLUMN EMPLEADO.APENOM
) AS
    DECLARE VARIABLE I INTEGER;
BEGIN
    I=0;
    SELECT E.LEGAJO,E.APENOM FROM EMPLEADO E
    WHERE E.LEGAJO IN ( SELECT EE.SU_JEFE_ES
                        FROM EMPLEADO EE
                        WHERE EE.LEGAJO = :PLEGAJO AND
                              EE.SU_JEFE_ES IS NOT NULL )
    INTO :LEGAJO,:APENOM;
    WHILE (I < 10 AND ROW_COUNT = 1) DO BEGIN
        SUSPEND;
        I=I+1;
        SELECT E.LEGAJO,E.APENOM FROM EMPLEADO E
        WHERE E.LEGAJO IN ( SELECT EE.SU_JEFE_ES
                            FROM EMPLEADO EE
                            WHERE EE.LEGAJO = :LEGAJO AND
                                  EE.SU_JEFE_ES IS NOT NULL )
        INTO :LEGAJO,:APENOM;
    END
END
```

4. Describa 3 anotaciones que se utilizan en JPA, indicando en qué ámbito se usa (clase, atributo, método), cuáles son sus parámetros y para qué sirve.

```
@Entity ámbito clase, sirve para indicar que esta clase es una entidad a
ser persistida en BDR
@Table(name=...,schema=...,catalog=...,...) ámbito clase, sirve para indicar
en que tabla será persistida esta entidad
@Id sin parámetros, ámbito atributo, sirve para indicar que esta columna
es un atributo primo o PK
@Column(name=...,nullable=...,length=...,...) ámbito atributo, sirve para
indicar que esta columna es un atributo de la tabla
..
```

PUNTUACION

EJ 1	3	PTOS
EJ 2	3	PTOS
EJ 3	2	PTOS
EJ 4	2	PTOS

6 PTOS 4,5
 7 PTOS 6
 8 PTOS 7,8
 9 PTOS 9
 10 PTOS 10

Condición de Aprobación: 60% del examen bien contestado y ejercicio 1 bien contestado.
--