# AMPLIACION DE BASE DE DATOS
## Examen Final Convocatoria Extraordinaria de 2021

**→ (para todos los ejercicios: es una posible solución, hay otras variaciones que son también correctas)**

## EJERCICIO 1 ( 2 puntos)

*(explica los conceptos que uses en las respuestas)*

**a)** Qué condiciones debe cumplir una tabla para estar en 3ª Forma Normal (FN)

```
----- SOLU ------
Consulta teoría para explicación :
  Toda DF X→ Y cumple: x e SC o Y parte CC (no DFs transitivas)
```

**b)** **b.1).-** Escribe un ejemplo de una tabla con 5 filas que esté en 3ª FN, explicando por qué está en 3FN basándote en los valores que has dado. La tabla debe tener dos atributos como clave primaria y otros dos atributos más: A,B,C,D y los posibles valores para A: a1, a2,… , etc.; para B: b1, b2,…; para C: c1,c2,…; y para D: d1, d2,…

```
----- SOLU ------ : cumple a).
Consulta teoría para explicación del razonamiento
```

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c2 | d2 |
| a2 | b1 | c3 | d3 |
| a2 | b2 | c4 | d4 |
| a3 | b4 | c5 | d5 |

**b.2).-** Escribe un ejemplo de la misma tabla con 5 filas que esté en **2ª FN**, explicando qué *ha cambiado* para *no* estar en 3FN, basándote en los valores que has dado.

```
----- SOLU ------
Consulta teoría para explicación del razonamiento : hay DF trans: AB → C → D
```

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c2 | d2 |
| a2 | b1 | c3 | d3 |
| a2 | b2 | c4 | d4 |
| a3 | b4 | c5 | d5 |

**c)** ¿Cuando normalizamos una tabla, qué dos propiedades son deseables mantener?

```
----- SOLU ------
Consulta teoría para explicación de las dos propiedades
- Que no se pierden la DFs
- Re-uniones no aditivas o sin pérdida:
  - Descomposición sin pérdida de info. de integridad (mantenida por claves)
  - Reuniones no aditivas (no generar filas/tuplas falsas)
```

## **EJERCICIO 2** ( 3 puntos)

Tenemos creada una colección llamada `Comics`, dentro de la BD llamada `exa`*DB* cuyos documentos (uno para cada Título de comic) tienen los siguientes campos (no todos tiene todos los campos):

*Categoría*: (del comic). *Soporte*: (cómo se publica). *Autor*. *Título*. *Puntuación* (de 0 a 20). *Precio*. *Propiedades* (es una lista cuyos elementos pueden ser diferentes para cada Título)

**a)** Escribe <u>un solo</u> `update`, para actualizar todos los documentos que cumplan las condiciones siguientes: *Categoría* = mangaShojo. *Soporte* = webAbierto. *Autor* = Ikeyamada. *Título* = Moe Kare. Los *cambios* a realizar cuando se cumplen esas condiciones son: *Puntuación* = 8. *Precio* = 0. y las *Propiedades* son dos: protagonista = Hikaru, totalCapítulos = 34. Si **no existe** ningún documento que cumpla esas condiciones, no se actualiza nada.

```
------ SOLU --------
db.comics.update(
       {$and: [{Categoria: "mangaShojo"},
                       {Soporte: "webAbierto"},
                       {Autor: "Ikeyamada"},
                       {Titulo: "MoeKare"}] } ,
        {  $set: { Puntuacion: 8, Precio: 0,
                   Propiedades: {protagonista: "Hikaru", totalCapitulos: 34}
         },
        { upsert: false}, {multi: true} )
```

**b)** Escribe <u>una sola</u> `aggregate` que calcula cuántos capítulos hay en cada *categoría* para los comics con puntuación entre 9 y 20, ambos incluidos. Y muestra el resultado para cada *categoría*: su nombre, el total de capítulos y cuántos comics (documentos).

```
---------- SOLU --------- (hay otras soluciones)
db.comics.aggregate([
  { $match: { "Puntuacion": { $gte: 9, $lte:20 } } },
  { $group:{_id: "$Categoria",   /* si null todas las categorías juntas*/
            totCapitulos:{ $sum: "$Propiedades.totalCapitulos" },
            totalComics: { $sum: 1 } } }])
```

**c)** Escribe <u>una solo</u> `find` que, aplicando una *función*, actualiza solo los Comics que contienen el campo *Precio* del siguiente modo: si tienen puntuación menor o igual a 7 y el total de capítulos es menor que 4, entonces añade un atributo *Interés* con valores "bajo, corto". Si tienen puntuación mayor de 7 y total de capítulos mayor de 20 añade el atributo *Interés* con valores "mucho, largo", y, además, incrementa su precio un 20%. En cualquier otro caso, añade el atributo *Interés* con valores "regular, medio".

```
----------- SOLU ----------------  (hay otras soluciones)

db.comics.find({Precio:{ $exists: true}}).forEach( function (myDoc) {
    if (myDoc.Puntuacion <= 7 &&
        myDoc.Propiedades.totalCapitulos < 4)
           { myDoc.Interes = "bajo corto";}
    else if(myDoc.Puntuacion > 7 &&
            myDoc.Propiedades.totalCapitulos > 20)
            { myDoc.Interes = "mucho largo";
              myDoc.Precio = myDoc.Precio * 1,20;}
                    /* línea anterior opcional si dices que son gratis  */
    else
     myDoc.Interes = "regular medio";
  db.comicX.save(myDoc);}
 );
```

## EJERCICIO 3 ( 5 puntos)

Tenemos este esquema relacional para enviar, desde el usuario actual, avisos a todos los alumnos (usuarios) de ABD:

**ALL_USERS(USERNAME, Número_ID, FechaCreacion)** tabla del sistema con todos los usuarios
**BandejaEntrada_XXXXX(usuarioOrigen,mensaje,fecha):** emails recibidos por XXXXX
**paraEnviar(mensaje, longitud): mensajes pendientes** para enviar a **todos los usuarios**
La tabla paraEnviar se asume ya creada antes de empezar
**Historico(usuarioOrigen,mensaje,fecha,usuarioDestino):** una sola tabla ya creada

→ Cada alumno tiene una tabla **paraEnviarA** con los mensajes de aviso y otra **BandejaEntrada_XXXXX**, en la cual **XXXXX** es el nombre del usuarioOrigen, que es del alumno que ejecuta el procedimiento. Así que usaremos "alumno" y "usuario" *indistintamente*. Como no sabemos el nombre del usuario lo represento en el enunciado con XXXXX, igual que con YYYYY, aunque en las tablas debe aparecer su nombre concreto real.

**a)** Hacer un procedimiento, en PL/SQL llamado *gestionAvisos* que incluya los siguientes pasos y transacciones :

- Crear una tabla **BandejaEntrada_XXXXX** , donde XXXXX es el nombre de usuario que ejecuta el proc. (función USER). La PK tiene todos los atributos. Solo se debe crear esta tabla si **no** existía. Si ya existe, se salta este paso.

- Terminar transacción.

- Para cada alumno YYYYY que sea de la asignatura ABD, que será el usuario destino (es el USERNAME que está en la tabla **ALL_USERS** y el nombre usuario debe empiezar por "ABD"):
  - (debe haber terminado la transacción anterior) Empezamos transacción. Usar el nivel aislamiento adecuado para que ninguna otra transacción vea las actualizaciones hasta que termine esta transacción.
  - Enviamos todos los mensajes pendientes al alumno YYYYY, siguiendo estos pasos:
    - Hacemos lo necesario para que nadie modifique tabla **BandejaEntrada_YYYYY** mientras hacemos el proceso
    - Recorremos la tabla **paraEnviarA,** donde para **cada fila**, enviamos el mensaje de este modo:
      - Insertar el mensaje en la tabla **BandejaEntrada_YYYYY** donde YYYYY debe ser el nombre usuario destino. La fecha es la actual del ordenador (variable sysdate).
    - Si llevamos creados más de 1.000 mensajes (en total con todos los usuarios ya tratados), deshacemos lo hecho de este YYYYY. Además terminamos todo el procedimiento.
    - En caso contrario, liberamos la tabla **BandejaEntrada_YYYYY**, terminamos transacción y seguimos con el siguiente alumno YYYYY.

- *No gestiones las excepciones*.

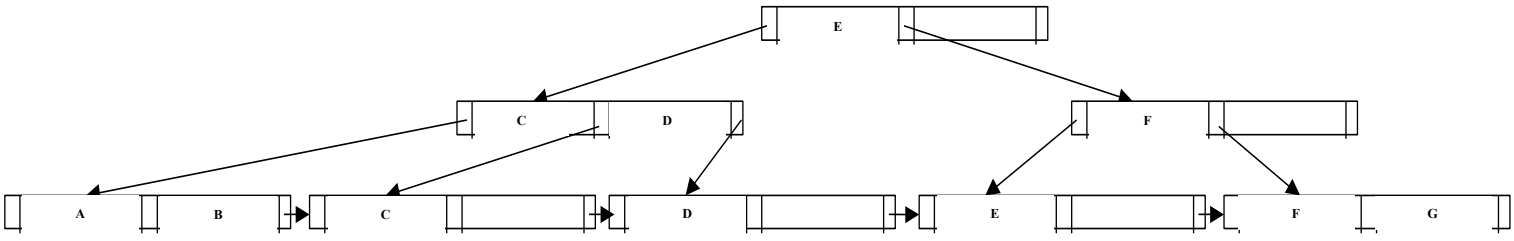
```
create or replace PROCEDURE GestionAvisos AS --- EXTRA-21 EJ3-a--PLSQL
  estaCreada INT;
  numensa INT := 0;
  usuarioEjecutor  CHAR(30);
  CURSOR cursor_usuarios IS
          select USERNAME from ALL_USERS where username like 'ABD%' and username <> USER;
BEGIN
usuarioEjecutor := USER;
                                    -- Comprueba si está creada la tabla de ese usuario
  EXECUTE IMMEDIATE 'BEGIN SELECT COUNT(*) INTO estaCreada
                     FROM BandejaEntrada_' || RTRIM(usuarioEjecutor) || ' ; END;' ;

                                      -- Si no está creada, la crea
  IF estaCreada = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE  BandejaEntrada_' || RTRIM(usuarioEjecutor) ||  '(
                          usuarioOrigen CHAR(30) NOT NULL,
                          mensaje VARCHAR(200),
                          fecha DATE,
                          CONSTRAINT pk_bandeja_' ||  RTRIM(usuarioEjecutor) ||
                            ' PRIMARY KEY(usuarioOrigen,mensaje,fecha))';
  END IF;
```

```
FOR alumnoYYYYY in cursor_usuarios
 LOOP
                                -- Comienza transacción para cada alumno  YYYYY
   set transaction isolation level SERIALIZABLE;
                           -- también habría que poner el nombreYYYYY. delante de BandejaEntrada_
   EXECUTE INMEDIATE 'BEGIN LOCK TABLE BandejaEntrada_' || RTRIM(alumnoYYYYY) ||
                      ' IN EXCLUSIVE MODE END;' ;

                                -- Enviando todos los mensajes al alumno YYYYY
   FOR envio in (select * from paraEnviarA)
      LOOP
         EXECUTE INMEDIATE
             'BEGIN     -- también habría que poner el nombreYYYYY. delante de BandejaEntrada_
                   INSERT INTO  BandejaEntrada_' || RTRIM(alumnoYYYYY) ||
                                 ' VALUES (usuarioEjecutor, envio.mensaje, sysdate);
              END;'; ;
        numensa := numensa + 1;
        IF numensa > 1000 THEN  ROLLBACK ;  exit; END IF;  -- sale del LOOP interno
    END LOOP;                   --- Envio a YYYYY

    commit;
    IF numensa > 1000 THEN    exit;  END IF;  -- sale del LOOP externo

END LOOP;          --- alumnoYYYYY
END;
```

**b) En este apartado usamos un usuario concreto,** el ABD12345: escribe un trigger **`TriggerHIST_ABD12345`** para que, en caso de inserción en la tabla **`BandejaEntrada_ABD12345`**, se incluya una fila en una tabla **`Historico`** ya creada, cuyo **`usuarioDestino`** es **`ABD12345`**. Si la inserción en **`BandejaEntrada_ABD12345`** se deshace en el procedimiento de **(a)**, la fila en **`Historico`** debe deshacerse también.

```
--------- TRIGGER APARTADO  b)
create or replace TRIGGER triggerHIST_ABD12345
AFTER INSERT  ON BandejaEntrada_ABD12345
FOR EACH ROW
BEGIN
   INSERT into Historico VALUES
              (:new.usuarioOrigen, :new.mensaje,  :new.fecha,'ABD12345');
END;
```

**c)** Dado el árbol B+ de un índice de claves alfabéticas, mostrado en la figura, dibuja como queda el árbol después de borrar la clave con valor "E".




**SOLUCION**