CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

EXAMEN MODELO A
2ª EVALUACIÓN
CONSULTAS, EXTENSIONES, GESTIÓN Y

BASES DE DATOS 22/23
CFGS DAW

NOSQL
SOLUCIONES

Autores:
Abelardo Martínez y Pau Miñana

BASES DE DATOS RELACIONALES

[4 PUNTOS] PARTE 1: CONSULTAS

1. Listado sin repeticiones con el N_Serie, SO y Resolución de pantalla de los PC usados por el Personal cuyo Nombre acabe en O bien que empiece por L, durante los meses de marzo y abril de 2022 ordenados por Resolución de pantalla de menor a mayor y SO.

```
SELECT DISTINCT PC.N_Serie, SO, Resolucion FROM Personal P
INNER JOIN Usa U ON P.DNI=U.DNI
INNER JOIN PC ON Terminal=PC.N_Serie
INNER JOIN Monitor M ON Pantalla=M.N_Serie
WHERE (Nombre LIKE'%O' OR Nombre LIKE 'L%')
AND Dia BETWEEN '2022-03-01' AND '2022-04-30'
ORDER BY Resolucion, SO;
```

2. Listado con el N_Serie y la Descripcion de los monitores que no van montados en ningún PC.

Opción 1

```
SELECT Mat.N_Serie,Descripcion
FROM Monitor M INNER JOIN Material Mat ON M.N_Serie=Mat.N_Serie
WHERE M.N_Serie NOT IN (SELECT Pantalla FROM PC);
```

Opción 2

```
SELECT Mat.N_Serie,Descripcion
FROM Monitor M INNER JOIN Material Mat ON M.N_Serie=Mat.N_Serie
WHERE NOT EXISTS (SELECT Pantalla FROM PC
                  WHERE Pantalla=M.N_Serie);
```

3. Muestra el Nombre y Apellido1 del personal que ha usado más de 3 veces el PC 66508656, ordenados por número de usos de mayor a menor y alfabéticamente por Apellido 1.

```
SELECT Nombre, Apellido1, COUNT(*) AS N_Usos
FROM Personal P INNER JOIN Usa U ON P.DNI=U.DNI
WHERE Terminal=66508656
GROUP BY Nombre, Apellido1
HAVING COUNT(*)>3
ORDER BY 3 DESC, Apellido1;
```

4. Crea una vista Departamentos con el nombre de cada departamento y el número de miembros del mismo, el personal sin departamento asignado no debe aparecer aquí. Ordenada por mayor cantidad de personal. Ahora usa la vista para mostrar el DNI y el departamento del Personal perteneciente AL o LOS departamentos con más personal asignado, ordenados alfabéticamente por Departamento.

Vista

```
DROP VIEW IF EXISTS Departamentos;
CREATE VIEW Departamentos AS
SELECT Departamento,COUNT(*) AS Miembros FROM Personal
WHERE Departamento IS NOT NULL
GROUP BY Departamento
ORDER BY 2 DESC;
```

Consulta

Opción 1

```
SELECT DNI, P.Departamento FROM Departamentos D
INNER JOIN Personal P ON P.Departamento=D.Departamento
WHERE Miembros=(SELECT MAX(Miembros) FROM Departamentos)
ORDER BY Departamento;
```

Opción 2

```
SELECT DNI, P.Departamento FROM Departamentos D
INNER JOIN Personal P ON P.Departamento=D.Departamento
WHERE Miembros >= ALL (SELECT Miembros FROM Departamentos)
ORDER BY Departamento;
```

Opción 3

```
SELECT DNI, P.Departamento FROM Departamentos D
INNER JOIN Personal P ON P.Departamento=D.Departamento
CROSS JOIN (SELECT MAX(Miembros) AS maximo
                FROM Departamentos) AS Aux1
WHERE Miembros=maximo
ORDER BY Departamento;
```

5. Muestra el Nombre_Completo, en un único campo, del Personal que más veces ha usado ordenadores y cuántas veces han sido, ordenados alfabéticamente de forma inversa por el Nombre_Completo

Opción 1

```
SELECT CONCAT(Nombre,' ',Apellido1) AS NombreC, COUNT(*) AS N_Usos
FROM Personal P INNER JOIN Usa U ON P.DNI=U.DNI
GROUP BY 1
HAVING COUNT(*)=(SELECT MAX(Us) FROM (SELECT COUNT(*) AS Us
                    FROM Usa
                    GROUP BY DNI) AS Usa_PC)
ORDER BY 1 DESC;
```

Opción 2

```
SELECT CONCAT(Nombre,' ',Apellido1) AS NombreC, COUNT(*) AS N_Usos
FROM Personal P INNER JOIN Usa U ON P.DNI=U.DNI
GROUP BY 1
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM Usa
                        GROUP BY DNI) AS Usa_PC
ORDER BY 1 DESC;
```

Opción 3

```
SELECT CONCAT(Nombre,' ',Apellido1) AS NombreC, COUNT(*) AS N_Usos
FROM Personal P INNER JOIN Usa U ON P.DNI=U.DNI
CROSS JOIN
(SELECT MAX(Us) as maxim FROM (SELECT COUNT(*) AS Us
                               FROM Usa
                               GROUP BY DNI) AS Usa_PC) AS Aux
GROUP BY 1,maxim
HAVING COUNT(*)= maxim
ORDER BY 1 DESC;
```

(O esto mismo con >= ALL como en la 2)

[3 PUNTOS] PARTE 2: EXTENSIONES

1. Crea una Función que reciba un Departamento y un N_Serie y devuelva TRUE si alguien del departamento ha usado ese PC y FALSE en caso contrario. Si alguno de los parámetros es incorrecto que salte una excepción indicándolo. Ignorad que el Departamento en realidad puede tener valor nulo, se considera un parámetro incorrecto igualmente.

```
DELIMITER $$
DROP FUNCTION IF EXISTS Usado$$
CREATE FUNCTION Usado (Dep VARCHAR(10),Ordenador INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN

  IF (SELECT DISTINCT Departamento FROM Personal
      WHERE Departamento=Dep) IS NULL
  THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
    'Parametros inesperados\n*****\n====>
    Departamento INCORRECTO\n*****\n';
  END IF;
  IF (SELECT N_Serie FROM PC WHERE N_Serie=Ordenador) IS NULL
  THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
    'Parametros inesperados\n*****\n====>
    N_Serie INCORRECTO\n*****\n';
  END IF;

  IF (SELECT COUNT(*)  FROM Usa U, Personal P
     WHERE U.DNI=P.DNI AND Departamento=Dep
     AND Terminal=Ordenador)> 0
  THEN
     RETURN TRUE;
  ELSE
     RETURN FALSE;
  END IF;
```

```
END$$
DELIMITER ;

SELECT Usado('IT',35019955) AS Usado;
SELECT Usado('RRHH',90424907) AS Usado;
SELECT Usado(NULL,90424907) AS Usado;
SELECT Usado('RRHH',90424903) AS Usado;
```

2. Crea los Triggers necesarios para impedir que un N_Serie que exista en la tabla PC pueda estar en “Monitor”.

El error es similar a otros casos, solo que usa una variable para poder usar *CONCAT y* mostrar el *NEW.N_Serie.*

```
DELIMITER $$
DROP TRIGGER IF EXISTS Esp_Monitor_INS$$
CREATE TRIGGER Esp_Monitor_INS
BEFORE INSERT
ON Monitor FOR EACH ROW
BEGIN

  DECLARE msg VARCHAR(100);
  IF NEW.N_Serie=(SELECT N_Serie FROM PC
                  WHERE N_Serie=NEW.N_Serie)
  THEN
    SET msg= CONCAT('Operacion no permitida\n*****\n====>',
    NEW.N_Serie,' es un PC.\n*****\n');
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
  END IF;

END$$

DROP TRIGGER IF EXISTS Esp_Monitor_UPD$$
CREATE TRIGGER Esp_Monitor_UPD
BEFORE UPDATE
ON Monitor FOR EACH ROW
BEGIN
```

```
 DECLARE msg VARCHAR(100);

 IF NEW.N_Serie<>OLD.N_Serie
 THEN
   IF NEW.N_Serie=(SELECT N_Serie FROM PC
                          WHERE N_Serie=NEW.N_Serie)
   THEN
     SET msg= CONCAT('Operacion no permitida\n*****\n====>',
     NEW.N_Serie,' es un PC.\n*****\n');
     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
   END IF;
 END IF;

END$$
DELIMITER ;
```

“IF NEW.N_Serie<>OLD.N_Serie” es importante, pues aunque no cambia el resultado evita que se tenga que realizar la consulta (más lenta) cuando la actualización no afecta al N_Serie.

[1 PUNTO] PARTE 3: GESTIÓN DE USUARIOS

1. Crea un usuario *administrador* que tenga permisos completos en la BD desde el servidor y permiso de consulta y actualización en todas las tablas desde la IP de su casa 213.0.87.102.

```
CREATE USER IF NOT EXISTS administrador@'localhost'
IDENTIFIED BY '***';
GRANT ALL PRIVILEGES ON IT.* TO administrador@'localhost';
FLUSH PRIVILEGES;
CREATE USER IF NOT EXISTS administrador@213.0.87.102
IDENTIFIED BY '***';
REVOKE ALL PRIVILEGES ON IT.* FROM administrador@213.0.87.102;
GRANT SELECT, UPDATE ON IT.* TO administrador@213.0.87.102;
FLUSH PRIVILEGES;
```

2. Crea un usuario *personal* que solo pueda usar la función creada en el apartado anterior, pero pueda otorgar permiso para usarla a otros. Desde cualquier IP.

```
CREATE USER IF NOT EXISTS personal@'%' IDENTIFIED BY '***';
REVOKE ALL PRIVILEGES ON IT.* FROM personal@'%';
GRANT EXECUTE ON PROCEDURE IT.Usado TO personal@'%'
WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

NOSQL

[2 PUNTOS] PARTE 4: MONGODB

Vamos a adaptar para MongoDB solo un extracto de parte del Personal de la Base de datos. Deja que se asignen los _id automáticamente y usa los siguientes datos:

DNI	Nombre	Apellido1	Departamento	Emails
"45393627E"	"CARMEN"	"GALLEGO"	"IT"	ARRAY
"26330456X"	"ALBERTO"	"FRANCO"	"Finanzas"	ARRAY

El ARRAY en Emails contendrá documentos con el email y la cantidad de correo No_Leido:

Email	No_leido
45393627E	
"cargal@ceedtech.com"	0
"IT@ceedtech.com"	0
26330456X	
"finanzas@ceedtech.com"	9
"albfra@ceedtech.com"	0

1. Borrar la base de datos si existiese, crearla de nuevo, crear la colección indicada e insertar los documentos indicados. Debes tener en cuenta el tipo de datos más adecuado para cada campo. Una vez insertados debes listar todos los documentos con todos sus campos ordenados por Departamento.

```
// limpiar pantalla (buena praxis)
cls
// conectar con la BD y borrarla (la crea si no existe)
use IT
```

```
db.dropDatabase()
use IT
// crear una nueva colección
db.createCollection("personal")
// usar el siguiente JSON para insertar en Compass o crear un array/2 documentos e insertarlos con db.personal.insertMany
[{
 "DNI":"45393627E",
"Nombre":"CARMEN",
"Apellido1":"GALLEGO",
"Departamento":"IT",
"Emails":[{"Email":"cargal@ceedtech.com","No_Leido":0},
{"Email":"IT@ceedtech.com","No_Leido":0}]
},
{
 "DNI":"26330456X",
"Nombre":"ALBERTO",
"Apellido1":"FRANCO",
"Departamento":"Finanzas",
"Emails":[{"Email":"finanzas@ceedtech.com","No_Leido":0},
{"Email":"albfra@ceedtech.com","No_Leido":9}]
}]

// consulta
db.Personal.find({}).sort({"Departamento":1})
```

2. Realiza una consulta que muestre el nombre y el apellido del personal que tenga correo No_Leido en cualquiera de sus Emails o sea del departamento de "IT", ordenados por Nombre y Apellido1 de forma alfabética inversa.

```
var j_valorA     = {$gt:0}
var j_filtro1    = {"Emails.No_Leido":j_valorA}
var j_filtro2    = {"Departamento":"IT"}
var j_filtroFinal = {$or:[j_filtro1, j_filtro2]}
var j_proy       = {"Nombre":1,"Apellido1":1,"_id":0}
var j_orden      = {"Nombre":-1,"Apellido1":-1}
```

```
// consulta
db.personal.find(j_filtroFinal, j_proy).sort(j_orden)


// Opción Directa
db.Personal.find(
{$or:[{"Emails.No_Leido":{$gt:0}},{"Departamento":"IT"}]},
{"Nombre":1,"Apellido1":1,"_id":0}).sort({"Nombre":-1,"Apellido1":-1})
```