# **EPS**

Asignatura: IIOP22 BASES DE DATOS

Curso: 2020/2021 Examen: Final Fecha: 24-05-21

Semestre: 2 Convocatoria: Ordinaria

Duración del examen: 2:30 horas

Se tiene el siguiente sistema de información sobre una biblioteca:

-Un autor crea varias obras (al menos una) y una obra es creada por varios autores (al menos uno). La información de los autores es id\_autor, nombre, país.

-En cuanto a las obras, pueden ser de 3 tipos: CD, libro o película. La información común de cada una obra es: id\_obra, nombre, año. La información de cada CD es numero de canciones. La información de cada libro es numero de paginas. La información de cada película es duración.

-Una obra tiene asociadas varias copias, siendo la existencia de una copia totalmente dependiente de la existencia de la obra a la que pertenece (es decir, una copia no puede existir si no existe la obra a la que están asociada). La información de una copia es numero de copia, si está deteriorada, comentarios.

-Un socio pide prestadas varias copias (opcionalmente ninguna). Una copia es prestada a varios socios (opcionalmente ninguno). La información del socio es: nombre, apellidos, cod\_socio, teléfono, dirección. Nota: el cod\_socio es un identificador único para los socios. Se almacena la información siguiente sobre los préstamos realizados: fecha\_prestamo, fecha\_devolución\_prestamo, fecha\_tope\_prestamo.

1. (2 puntos) Realizar el diagrama entidad-relación, indicando entidades, relaciones, atributos, clave primaria, cardinalidad de las relaciones.

2. (2 puntos) Realizar el diagrama o modelo relacional, es decir, escribir los esquemas relacionales, indicando bien en cada esquema el nombre de la tabla, atributos e indicar que atributo(s) forma(n) la clave primaria.

Autor= (id\_autor, nombre, país)

Obra= (id\_obra, nombre, año)

Socio= (cod\_socio, nombre, apellidos, tlf, dirección)

Copias= (n\_copia, id\_obra, deteriorado, comm)

CD= (id\_obra, n\_canciones)

Libro= (id\_obra, n\_paginas)

Película= (id\_obra, duración)

Autor\_Obra = ( id\_autor, id\_obra )

3. Dado el siguiente esquema relacional de un banco: cada sucursal tiene al menos un cliente (y un cliente solo pertenece a una sucursal); y un cliente tiene al menos una cuenta (y una cuenta pertenece a un cliente solamente):

sucursal (numero\_sucursal, ciudad)

cliente (numero\_cliente, nombre, apellido, teléfono, numero\_sucursal)

cuenta (numero\_cuenta, saldo, fecha\_apertura, numero\_cliente)

Respecto a los tipos de datos, todos son varchar, menos saldo que es decimal, y fecha, que es de tipo datetime.

Codifica las siguientes sentencias en MySQL para realizar lo siguiente:

1. (0,75 puntos) Crear las tres tablas, con la primary key (clave primaria) y las foreign key (clave forénea). Las sentencias de creación de tablas deben estar completas.

CREATE TABLE sucursal(numero\_sucursal VARCHAR PRIMARY KEY, ciudad VARCHAR);

CREATE TABLE cliente(numero\_cuenta VARCHAR(50) PRIMARY KEY, nombre VARCHAR(50), apellido VARCHAR(50), numero cliente VARCHAR(50), FOREIGN KEY(numnero\_sucursal) REFERENCES sucursal(numero\_sucursal));

CREATE TABLE cuenta(n\_cuenta VARCHAR(50) PRIMARY KEY, saldo DECIMAL, fecha\_apertura DATETIME,n\_cliente VARCHAR(50) FOREIGN KEY(n\_cliente) REFERENCES cliente(numero\_cliente));

1. (1 puntos) Mostrar los números de las sucursales junto a los nombres y apellidos de los clientes de cada cuenta, y los números de cuenta de cada cliente. Debe mostrarlo por orden ascendente de nombre de sucursal. Además en la cabecera (o título) de cada atributo debe mostrar Sucursal (para número\_sucursal), Cliente (para nombre apellido del cliente), Cuenta (para número\_cuenta).

Un ejemplo de la salida es el siguiente:

SELECT sucursal.numero\_sucursal AS Sucursal, CONCAT(cliente.nombre,cliente.apellido) AS Cliente, cuenta.n\_cuenta AS Cuenta ORDER BY sucursal.nombre\_sucursal ASC;

Texto, Tabla

Descripción generada automáticamente

1. (0,5 puntos) Crear una vista que muestre el atributo ciudad de las sucursales y la cantidad de clientes que tiene cada sucursal.

CREATE VIEW vista AS SELECT sucursal.ciudad, COUNT(\*) FROM clientes GROUP BY numer\_sucursal;

1. (0,75 ) Crear una vista que muestre los números de cliente de la sucursal 1 (numero\_ sucursal es 1), y la suma de saldo de todas las cuentas de los clientes de esa sucursal.

CREATE VIEW vista2 AS SELECT numero\_cliente,SUM(saldo), FROM cliente WHERE cliente.sucursal=”1”, GROUP BY cliente.numero\_cliente;

1. (0,25 puntos) Borra el índice primario de la tabla sucursal.

DROP INDEX ‘PRIMARY’ ON sucursal

1. (1,25 puntos) Crear un procedimiento almacenado que realice lo siguiente:

- Parametros de entrada: n\_cliente (de tipo varchar(10)), limite (de tipo decimal(13,2))

-Parametro de salida: tipo (de tipo varchar(10)

-Crear una variable llamada resultado de tipo int

-Guardar en la variable resultado, la cantidad de cuentas cuyo numero\_cliente coincida con n\_cliente, pero que cuyo saldo NO supere limite.

-Si resultado es mayor que 1, guardar en la variable tipo el valor ‘ALTO’; si resultado es igual a 1, guardar en la variable tipo el valor ‘MEDIO’, si resultado es igual a 0, guardar en la variable tipo el valor ‘BAJO’.

DELIMITER //

CREATE PROCEDURE procedimiento(IN n\_cliente VARCHAR(10), IN limite DECIMAL(13,2), OUT tipo VARCHAR(10))

BEGIN

DECLARE resultado INT

SELECT COUNT(numero\_cuenta) INTO resultado FROM cuenta WHERE cuenta.numero\_cliente=n\_cliente AND cuenta.saldo<=limite;

IF resultado>1 THEN

SET Tipo=’ALTO’

ELSEIF resultado = 1 THEN

SET Tipo = ‘MEDIO’

ELSE

SET Tipo =’BAJO’

END IF;

END //

DELIMITER

1. (0,5 puntos) Invocar a este procedimiento del apartado g. con el n\_cliente 10100, limite de 1000.50. Mostrar el valor del parámetro de salida llamado tipo.

CALL procedimiento(‘1’,12000,@out) SELECT @out;

h. (0,25 puntos) Crear una tabla llamada movimientos que contenga los siguientes atributos: id\_cuenta varchar(10), cantidad decimal(13,2), fecha datetime). Esta tabla va a contener las actualizaciones que se realicen sobre la tabla cuenta.

CREATE TABLE movimientos (id\_cuenta VARCHAR(10), cantidad DECIMAL(13,2),fecha DATETIME)

i. (0,75 puntos) Crear un disparador que realice lo siguiente:

Después de cada actualización sobre la tabla cuenta, si el saldo nuevo es mayor del anterior, debe insertar una fila nueva en la tabla movimientos, con el numero\_cuenta que se ha visto actualizada, la diferencia entre los saldos (saldo nuevo menos saldo anterior), y el día y hora actuales.

DELIMITER //

CREATE TRIGGER update\_movimientos

AFTER UPDATE on cuenta FOR EACH ROW

BEGIN

IF new.saldo >= old.saldo

INSERT INTO movientos VALUES(numero\_cuenta,new.saldo-old.saldo,GETDATE())

END IF;

END //

DELIMITER;