Examen Tercer Trimestre - BAE

NOMBRE:
APELLIDOS:
FECHA: 26-05-2022

IES Puerto de la Cruz Telesforo Bravo

BLOQUE 1

1) Crea una **base de datos** llamada `ExamenFinal` que contenga una **tabla** llamada `ejercicio`. La tabla debe tener una única columna llamada `número` y el tipo de dato de esta columna debe ser `INT UNSIGNED`.
Una vez creada la base de datos y la tabla deberá **crear un procedimiento** llamado `calcular_números` con las siguientes características. El procedimiento recibe un parámetro de entrada llamado `valor_inicial` de tipo `INT UNSIGNED` y deberá almacenar en la tabla `ejercicio` toda la secuencia de números desde el valor inicial pasado como entrada hasta el 1.
Tenga en cuenta que el procedimiento deberá eliminar el contenido actual de las tablas antes de insertar los nuevos valores.

Utilice un bucle `WHILE` para resolver el procedimiento.

2) Crea una **base de datos** llamada `procedimientos` que contenga una **tabla** llamada `pares` y otra **tabla** llamada `impares`. Las dos tablas deben tener única columna llamada `número` y el tipo de dato de esta columna debe ser `INT UNSIGNED`.

Una vez creada la base de datos y las tablas deberá **crear un procedimiento** llamado `calcular_pares_impares` con las siguientes características. El procedimiento recibe un parámetro de entrada llamado `tope` de tipo `INT UNSIGNED` y deberá almacenar en la tabla `pares` aquellos números pares que existan entre el número 1 el valor introducido como parámetro. Habrá que realizar la misma operación para almacenar los números impares en la tabla `impares`. Para realizar el cálculo de si es par o impar deberá realizar una función que deberá ser llamada desde el procedimiento calcular_pares_impares.

Tenga en cuenta que el procedimiento deberá eliminar el contenido actual de las tablas antes de insertar los nuevos valores.

Utilice un bucle `REPEAT` para resolver el procedimiento del ejercicio anterior.

BLOQUE 2

Dada la siguiente base de datos tienda:

```sql
CREATE DATABASE tienda;
USE tienda;

CREATE TABLE fabricante (
   codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL
);

CREATE TABLE producto (
   codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   precio DOUBLE NOT NULL,
```

```
  codigo_fabricante INT UNSIGNED NOT NULL,
  FOREIGN KEY (codigo_fabricante) REFERENCES fabricante(codigo)
);

INSERT INTO fabricante VALUES(1, 'Asus');
INSERT INTO fabricante VALUES(2, 'Lenovo');
INSERT INTO fabricante VALUES(3, 'Hewlett-Packard');
INSERT INTO fabricante VALUES(4, 'Samsung');
INSERT INTO fabricante VALUES(5, 'Seagate');
INSERT INTO fabricante VALUES(6, 'Crucial');
INSERT INTO fabricante VALUES(7, 'Gigabyte');
INSERT INTO fabricante VALUES(8, 'Huawei');
INSERT INTO fabricante VALUES(9, 'Xiaomi');

INSERT INTO producto VALUES(1, 'Disco duro SATA3 1TB', 86.99, 5);
INSERT INTO producto VALUES(2, 'Memoria RAM DDR4 8GB', 120, 6);
INSERT INTO producto VALUES(3, 'Disco SSD 1 TB', 150.99, 4);
INSERT INTO producto VALUES(4, 'GeForce GTX 1050Ti', 185, 7);
INSERT INTO producto VALUES(5, 'GeForce GTX 1080 Xtreme', 755, 6);
INSERT INTO producto VALUES(6, 'Monitor 24 LED Full HD', 202, 1);
INSERT INTO producto VALUES(7, 'Monitor 27 LED Full HD', 245.99, 1);
INSERT INTO producto VALUES(8, 'Portátil Yoga 520', 559, 2);
INSERT INTO producto VALUES(9, 'Portátil Ideapd 320', 444, 2);
INSERT INTO producto VALUES(10, 'Impresora HP Deskjet 3720', 59.99, 3);
INSERT INTO producto VALUES(11, 'Impresora HP Laserjet Pro M26nw', 180, 3);
```

3) Escribe una función para la base de datos `tienda` que devuelva el número total de productos que hay en la tabla `productos`.
4) Escribe una función para la base de datos `tienda` que devuelva el valor medio del precio de los productos de un determinado fabricante que se recibirá como parámetro de entrada. El parámetro de entrada será el nombre del fabricante.
5) Escribe un procedimiento que muestre el nombre del fabricante donde su ninguno de la descripción de sus productos contenga la S.

BLOQUE 3:

6)Crea una base de datos llamada `bloque3` que contenga una tabla llamada `alumno`. La tabla debe tener cuatro columnas:

- `id`: entero sin signo (clave primaria).
- `nombre`: cadena de 50 caracteres.
- `apellido1`: cadena de 50 caracteres.
- `apellido2`: cadena de 50 caracteres.

Una vez creada la base de datos y la tabla deberá **crear un procedimiento** llamado `insertar_alumno` con las siguientes características. El procedimiento recibe cuatro parámetros de **entrada** (`id`, `nombre`, `apellido1`, `apellido2`) y los insertará en la tabla `alumno`. El procedimiento devolverá como **salida** un parámetro llamado `error` que tendrá un valor igual a 0 si la operación se ha podido realizar con éxito y un valor igual a 1 en caso contrario.
Deberá manejar los errores que puedan ocurrir cuando se intenta insertar una fila que contiene una clave primaria repetida.

7) Crea una base de datos llamada `bloque3` que contenga una tabla llamada `alumnos` con las siguientes columnas.

Tabla `alumnos`:

- `id` (entero sin signo)
- `nombre` (cadena de caracteres)
- `apellido1` (cadena de caracteres)
- `apellido2` (cadena de caracteres)
- `nota` (número real)

Una vez creada la tabla escriba **dos triggers** con las siguientes características:

- Trigger 1: `trigger_check_nota_before_insert`
  - Se ejecuta sobre la tabla `alumnos`.
  - Se ejecuta antes de una operación de inserción.
  - Si el nuevo valor de la nota que se quiere insertar es negativo, se guarda como 0.
  - Si el nuevo valor de la nota que se quiere insertar es mayor que 10, se guarda como 10.
- Trigger2 : `trigger_check_nota_before_update`
  - Se ejecuta sobre la tabla `alumnos`.
  - Se ejecuta antes de una operación de actualización.
  - Si el nuevo valor de la nota que se quiere actualizar es negativo, se guarda como 0.
  - Si el nuevo valor de la nota que se quiere actualizar es mayor que 10, se guarda como 10.

Una vez creados los triggers escriba varias sentencias de inserción y actualización sobre la tabla `alumnos` y verifica que los triggers se están ejecutando correctamente.

8)Crea una base de datos llamada `bloque3` que contenga una tabla llamada `alumnos` con las siguientes columnas.

Tabla `datosalumn`:

- `id` (entero sin signo)
- `nombre` (cadena de caracteres)
- `apellido1` (cadena de caracteres)
- `apellido2` (cadena de caracteres)
- `email` (cadena de caracteres)

Escriba una **función** llamado **crear_email** que dados los parámetros de entrada: `nombre`, `apellido1`, `apellido2` y `dominio`, cree una dirección de email y la devuelva como salida.

- Procedimiento: `crear_email`
- Entrada:
  - `nombre` (cadena de caracteres)
  - `apellido1` (cadena de caracteres)
  - `apellido2` (cadena de caracteres)
  - `dominio` (cadena de caracteres)
- Salida:
  - `email` (cadena de caracteres)

devuelva una dirección de correo electrónico con el siguiente formato:

- El primer carácter del parámetro `nombre`.
- Los tres primeros caracteres del parámetro `apellido1`.
- Los tres primeros caracteres del parámetro `apellido2`.

- El carácter `@`.
- El dominio pasado como parámetro.

Una vez creada la tabla escriba **un trigger** con las siguientes características:

- Trigger: `trigger_crear_email_before_insert`
  - Se ejecuta sobre la tabla `alumnos`.
  - Se ejecuta antes de una operación de inserción.
  - Si el nuevo valor del email que se quiere insertar es `NULL`, entonces se le creará automáticamente una dirección de email y se insertará en la tabla.
  - Si el nuevo valor del email no es `NULL` se guardará en la tabla el valor del email.

Nota: Para crear la nueva dirección de email se deberá hacer uso de la función `crear_email`.

9) Modifica el ejercicio anterior y añade un nuevo trigger que tenga las siguientes características:

Trigger: `trigger_guardar_email_after_update`:

- Se ejecuta sobre la tabla `alumnos`.
- Se ejecuta después de una operación de actualización.
- Cada vez que un alumno modifique su dirección de email se deberá insertar un nuevo registro en una tabla llamada `log_cambios_email`.

La tabla `log_cambios_email` contiene los siguientes campos:

- `id`: clave primaria (entero autonumérico)
- `id_alumno`: id del alumno (entero)
- `fecha_hora`: marca de tiempo con el instante del cambio (fecha y hora)
- `old_email`: valor anterior del email (cadena de caracteres)
- `new_email`: nuevo valor con el que se ha actualizado

Baremo de calificaciones:

Bloque 1: 1,5 punto (0,75 cada ejercicio)
Bloque 2: 1,5 puntos (0,5 cada ejercicio)
Bloque 3: 7 puntos (1,75 cada ejercicio)

Para puntuar cada ejercicio deberá resolver correctamente el enunciado planteado, con las especificaciones descritas y devolviendo el resultado correcto.

No está permitido el uso de internet salvo para la entrega del examen, cualquier uso que exceda de esto supondrá la retirada del examen y por consiguiente el suspenso del mismo.

Con este examen se superarán los resultados de aprendizaje R5 y R6 descritos en la programación.