Evaluación Técnica de Programación

AUTOR	VERSIÓN
Alberto Salazar Trejo	Versión 24.4

PUNTUACIÓN:

La organización de la evaluación y la distribución de puntos en los diferentes ejercicios propuestos se resume en la siguiente tabla:

EJERCICIOS	PUNTOS
1. Conocimientos SQL	10
2. Ejercicio práctico: BD	15
3. Ejercicio práctico: Desarrollo	50
4. Documentación	25
TOTAL	**100**

NOTA: A partir de la recepción de la evaluación contará con 72 horas para enviarla.

1. CONOCIMIENTOS SQL

1.1) Describe el funcionamiento general de la sentencia **JOIN**.
Un JOIN es la operación usada para combinar registros entre diversas tablas de una base de datos, esta combinación es lograda mediante las relaciones existentes entre las tablas, es decir las claves primarias y las claves foráneas.

1.2) ¿Cuáles son los tipos de **JOIN** y cuál es el funcionamiento de los mismos?
Actualmente tengo conocimiento de 4 tipos; INNER JOIN aquel que devuelve solo las filas relacionadas entre ambas tablas (dando el ejemplo de solo dos tablas), LEFT JOIN de forma muy específica devuelve las filas de la tabla de la izquierda con los datos relacionados de la tabla de la derecha, RIGHT JOIN prácticamente cumple la misma funcionalidad que el anterior a la inversa, es decir, esta vez devuelve todas las filas de la tabla de la derecha con los datos relacionados de la tabla de la izquierda y por ultimo pero no menos importante el FULL JOIN, este devuelve todas las filas de ambas tablas cuando existe una relación entre sí.

1.3) ¿Cuál el funcionamiento general de los **TRIGGER** y qué propósito tienen?
Los TRIGGERS cumplen con la función de ejecutar algún bloque de código tras un evento dentro de la base de datos, dicho disparador puede ejecutarse ya sea antes, después o bien la sustitución del evento con una acción diferente, por ejemplo, en algún punto un desarrollador hace un insert en una tabla, un disparador podría ejecutar un procedimiento almacenado para actualizar mas registros en otras tablas tomando como detonante el insert anteriormente mencionado.

1.4) ¿Qué es y para qué sirve un STORED PROCEDURE?

Un STORED PROCEDURE es un conjunto de acciones o instrucciones que pueden ser ejecutadas cuando sean necesarias, este permite la ejecución de múltiples bloques de código con diversos fines pero que cumplen una función específica, dichos procedimientos pueden contener parámetros, lo cual facilita su utilización en distintos escenarios tomándolos como referencia.

Considerando que las siguientes tablas:

productos	
PK	idProducto INT(6)
	nombre VARCHAR(40)
	precio DECIMAL(16,2)

ventas	
PK	idVenta INT(6)
FK1	idProducto INT(6)
	cantidad INT(6)

Tienen los siguientes datos insertados:

idProducto	nombre	precio
1	LAPTOP	3000.00
2	PC	4000.00
3	MOUSE	100.00
4	TECLADO	150.00
5	MONITOR	2000.00
6	MICROFONO	350.00
7	AUDIFONOS	450.00

idVenta	idProducto	cantidad
1	5	8
2	1	15
3	6	13
4	6	4
5	2	3
6	5	1
7	4	5
8	2	5
9	6	2
10	1	8

Hacer las consultas necesarias para:

1.5) Traer todos los productos que tengan una venta.

Utilizare JOINS:

```
SELECT p.idProducto, p.nombre, p.precio FROM productos p
INNER JOIN ventas v ON p.idProducto = v.idProducto
```

1.6) Traer todos los productos que tengan ventas y la cantidad total de productos vendidos.

Utilizare JOINS:

```
SELECT p.idProducto, p.nombre, p.precio, SUM(v.cantidad) as ProductosVendidos FROM productos p
INNER JOIN ventas v ON p.idProducto = v.idProducto
```

1.7) Traer todos los productos (independientemente de si tienen ventas o no) y la suma total ($) vendida por producto.

Este resultado mostrara lo esperado pero cuando no existan ventas mostrara ‘null’.

```
SELECT p.idProducto, p.nombre, p.precio, SUM(v.cantidad * p.precio) FROM productos p
INNER JOIN ventas v ON p.idProducto = v.idProducto
```

Este resultado mostrara lo esperado remplazando null por 0.

```
SELECT p.idProducto, p.nombre, p.precio, COALESCE(SUM(v.cantidad * p.precio)) FROM productos p
INNER JOIN ventas v ON p.idProducto = v.idProducto
```

ESCENARIO: Leer el siguiente caso detenidamente para trabajar los puntos 2, 3 y 4 de tu evaluación.

Una empresa necesita un sistema para administrar su inventario en el almacén. Este sistema debe tener las siguientes características:

- Debe tener un **inicio de sesión**.

- Debe tener un **módulo** para ver el **inventario** de la empresa.

◦ Se debe poder agregar nuevos productos al inventario. Al agregar un producto, la cantidad inicial será 0.
◦ Se debe poder aumentar el inventario de los productos (**Entrada de productos**). Si intentas disminuir la cantidad de inventario actual, mostrará un mensaje de error.
◦ Se debe poder dar de baja un producto. La baja no elimina el registro, solo actualiza el estatus. Los productos dados de baja se pueden activar nuevamente. ◦ Se pueden ver los productos activos e inactivos.

- Debe haber un **módulo** para sacar(restar) inventario del almacén (**Salida de productos**).

◦ Solo se pueden ver los productos activos.
◦ No se puede sacar una cantidad mayor de un producto de la que está en inventario. Si se intenta hacer, mostrará un mensaje de error.

- Debe haber un **módulo** con el **historial** de movimientos de "Entrada" y "Salida" de productos.

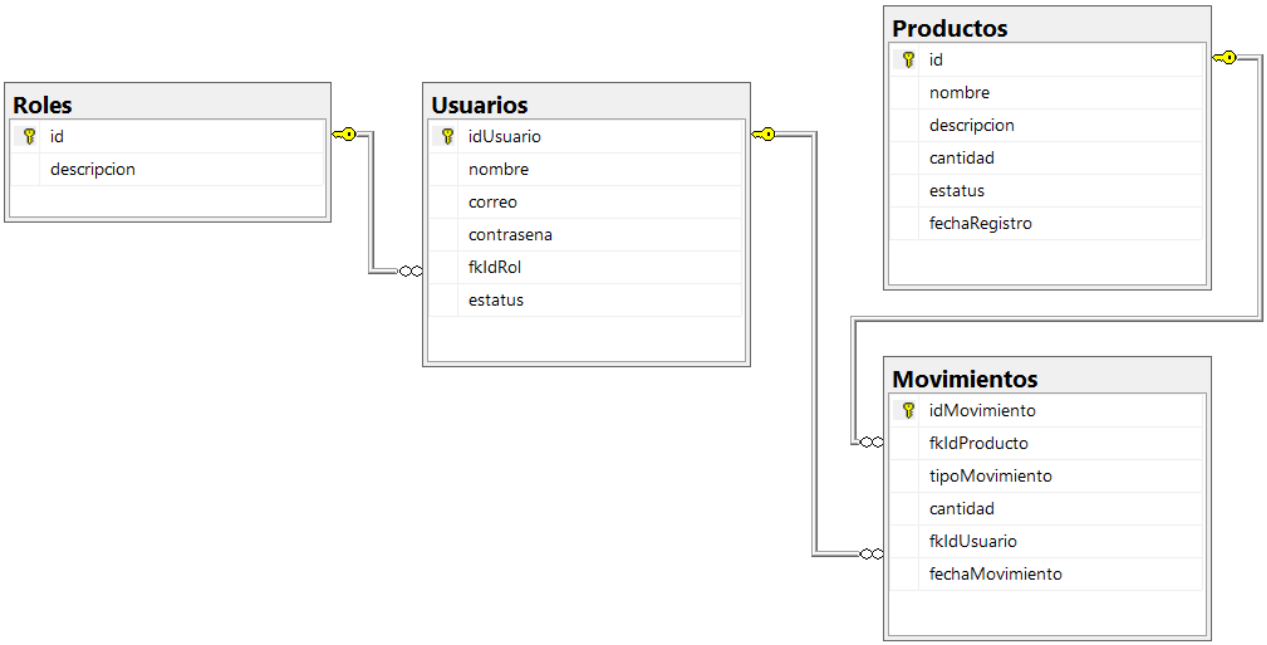
◦ El listado de movimientos se debe poder filtrar por tipo de movimiento (entrada o salida).
◦ Cada movimiento debe registrar quién lo realizó.
◦ Cada movimiento debe tener la fecha y hora en que se realizó el movimiento.

- El sistema tendrá 2 roles y los permisos de los mismos se describen a continuación:

Permiso	Administrador	Almacenista
Ver módulo inventario	✓	✓
Agregar nuevos productos	✓	✕
Aumentar inventario	✓	✕
Dar de baja/reactivar un producto	✓	✕
Ver módulo para Salida de productos	✕	✓
Sacar inventario del almacén	✕	✓
Ver módulo del histórico	✓	✕

2. EJERCICIO PRÁCTICO: BD

2.1) Crea un diagrama relacional de BD para el escenario descrito anteriormente.



2.2) Hacer el script para crear las tablas del punto anterior (en el punto 4.2 se especificará dónde debe anexarsedicho script).

NOTA: Actualmente solo se cuenta con la tabla de "usuarios". Falta considerar las tablas para saber identificar el tipo de rol, los productos y para el histórico.

3. EJERCICIO PRÁCTICO: DESARROLLO

3.1) Crear una aplicación **WEB** para el escenario que se planteó previamente.

- Preferiblemente utilizar el diseño **MVC** para su desarrollo.
- Preferiblemente utilizar **JAVA**.

- Utilizar **MySQL** o **SQLServer**.

4. DOCUMENTACIÓN:

4.1) Se necesita que el código se suba en un repositorio de **GITHUB** y esté público para que pueda ser descargado.

https://github.com/GabrielLiraChaires/EvaluacionTecnicaProgramacion-LCJG.git

4.2) Los scripts generados para la evaluación deben encontrarse en una carpeta llamada **SCRIPTS**, dentro del repositorio de GITHUB.

4.2) Dentro del repositorio, debe haber un **README.md** donde se especifiquen los datos relevantes para el desarrollo:

- IDE utilizado.
- Versión del lenguaje de programación utilizado.
- DBMS utilizado y su versión.
- Lista de pasos para correr su aplicación.

4.3) Haga un **video** donde se muestren los siguientes escenarios dentro de su aplicación:
Mostrar que los usuarios con el rol de **Administrador** pueden:

- Iniciar de sesión.
- Visualizar el histórico y filtrarlo.
- Registrar productos nuevos.
- Dar de baja productos.
- Reactivar productos.
- Agregar existencias a los productos.
- No pueden acceder al módulo para salida de material.

Mostrar que los usuarios con el rol de **Almacenista** pueden:

- Iniciar de sesión.
- No pueden visualizar el histórico.
- Solo pueden visualizar el módulo de inventario.
- Puede acceder al módulo para salida de material y sacar material.

Mostrar los mensajes de error que se consideraron en el escenario planteado.

NOTA: Dicho video se puede compartir a través de Google Drive.

https://drive.google.com/file/d/1VVKzFUh9hezJ43mu4VNUsYXjC-MG6KKy/view?usp=sharing