Examen-Laboratorio-IISSI-I-23-24...

YooAlixa

Introducción a la Ingeniería del Software y los Sistemas de Información I

2º Grado en Ingeniería Informática - Tecnologías Informáticas

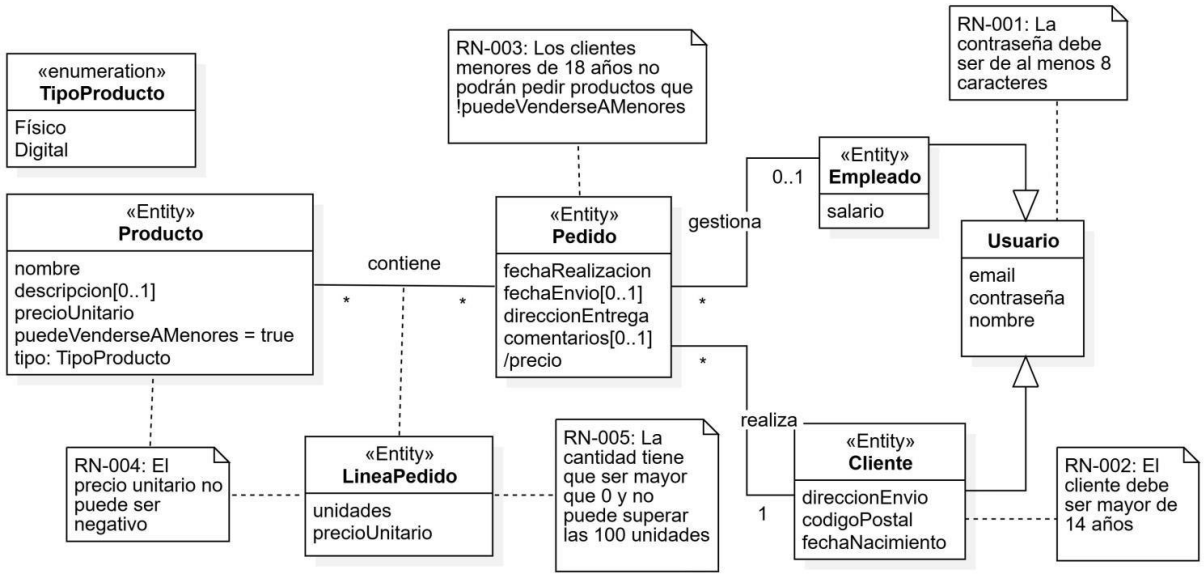
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla

Enunciado Evaluación Individual de Laboratorio Modelo C

Si usted entrega sin haber sido verificada su identidad no podrá ser evaluado. El procedimiento de entrega se encuentra al final de este documento.

Tienda Online

Partiendo de la `tiendaOnline` vista durante los laboratorios descrita en el modelado conceptual siguiente:



Las tablas y datos de prueba iniciales se encuentran en los ficheros `0.creacionTablas.sql` y `0.poblarBd.sql`.

Realice los siguientes ejercicios:

1. Creación de tabla. (1,5 puntos)

Incluya su solución en el fichero `1.solucionCreacionTabla.sql`.

Necesitamos conocer la opinión de nuestros clientes sobre nuestros productos. Para ello se propone la creación de una nueva tabla llamada `Valoraciones`. Cada valoración versará sobre un producto y será realizada por un solo cliente. Cada producto podrá ser valorado por muchos clientes. Cada cliente podrá realizar muchas valoraciones. Un cliente no puede valorar más de una vez un mismo producto.

Para cada valoración necesitamos conocer la puntuación de 1 a 5 (sólo se permiten enteros) y la fecha en que se realiza la valoración.

2. Consultas SQL (DQL). 3 puntos

Incluya su solución en el fichero `2.solucionConsultas.sql`.

2.1. Devuelva el nombre del producto, el precio unitario y las unidades compradas para las 5 líneas de pedido con más unidades. **(1 punto)**

2.3. Devuelva el nombre del empleado, la fecha de realización del pedido, el precio total del pedido y las unidades totales del pedido para todos los pedidos que de más 7 días de antigüedad desde que se realizaron. Si un pedido no tiene asignado empleado, también debe aparecer en el listado devuelto. **(2 puntos)**

3. Procedimiento. Bonificar pedido retrasado. 3,5 puntos

Incluya su solución en el fichero `3.solucionProcedimiento.sql`.

Cree un procedimiento que permita bonificar un pedido que se ha retrasado debido a la mala gestión del empleado a cargo. Recibirá un identificador de pedido, asignará a otro empleado como gestor y reducirá un 20% el precio unitario de cada línea de pedido asociada a ese pedido. **(1,5 puntos)**

Asegure que el pedido estaba asociado a un empleado y en caso contrario lance excepción con el siguiente mensaje: **(1 punto)**

```
El pedido no tiene gestor.
```

Garantice que o bien se realizan todas las operaciones o bien no se realice ninguna. **(1 punto)**

4. Trigger. 2 puntos

Incluya su solución en el fichero `4.solucionTrigger.sql`.

Cree un trigger llamado `p_limitar_unidades_mensuales_de_productos_fisicos` que, a partir de este momento, impida la venta de más de 1000 unidades al mes de cualquier producto físico.

SOLUCIONES

1. Creación de tabla.

```sql
CREATE TABLE valoraciones(
id INT PRIMARY KEY AUTO_INCREMENT,
clienteId INT NOT NULL,
productoId INT NOT NULL,
puntuacion INT NOT NULL CHECK (puntuacion >=1 AND puntuacion<=5),
fecha DATE NOT NULL,
UNIQUE(clienteId,productoId),
FOREIGN KEY (clienteId) REFERENCES Clientes(id),
FOREIGN KEY (productoId) REFERENCES Productos(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

2. Consultas SQL (DQL).

```sql
-- 2.1 - Devuelva el nombre del producto, el precio unitario y las unidades compradas para las 5 líneas de pedido con más unidades.

SELECT p.nombre AS nombre_producto ,lp.precio AS precio_unitario ,lp.unidades AS unidades_compradas
FROM lineaspedido lp JOIN productos p ON lp.productoId=p.id
ORDER BY unidades_compradas DESC
LIMIT 5;

-- 2.3
/*Devuelva el nombre del empleado, la fecha de realización del pedido, el precio total del pedido y
las unidades totales del pedido para todos los pedidos que de más 7 días de antigüedad desde que
se realizaron. Si un pedido no tiene asignado empleado, también debe aparecer en el listado
devuelto. */
SELECT u.nombre AS nombre, p.fechaRealizacion AS fecha_realizacion, SUM(lp.unidades * lp.precio) AS precio_total, SUM(lp.unidades) AS total_unidades
FROM usuarios u JOIN empleados e ON e.usuarioId=u.id
RIGHT JOIN pedidos p ON p.empleadoId=e.id
JOIN lineaspedido lp ON p.id=lp.pedidoId
GROUP BY p.id
HAVING TIMESTAMPDIFF(DAY,fecha_realizacion,CURDATE())>7;
```

3. Procedimiento. Bonificar pedido retrasado

```
DELIMITER //

CREATE OR REPLACE PROCEDURE bonificar_pedido_retrasado(IN p_pedidoId INT)
-- incluya su solución a continuación

BEGIN
DECLARE empleado INT;
DECLARE pedido ROW TYPE OF pedidos;
DECLARE nuevoempleado INT;


 -- Manejo de errores
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
     ROLLBACK;
     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al bonificar pedido retrasado';
 END;
```

```
-- Iniciar transacción
START TRANSACTION;

SET pedido =    (SELECT *
                    FROM pedidos p
                    WHERE p.id=p_pedidoId);

SET empleado =  (SELECT e.id
                    FROM empleados e
                    WHERE e.id = pedido.empleadoId);

IF empleado IS NULL THEN
      SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'El pedido no tiene gestor';
END IF;

    SELECT e.id INTO nuevoempleado
    FROM empleados e
    WHERE e.id != empleado
    LIMIT 1;

    UPDATE pedidos
    SET empleadoId = nuevoempleado
    WHERE id = p_pedidoId;

    UPDATE lineaspedido
    SET precio = precio*0.8
    WHERE pedidoId = p_pedidoId;

 -- Confirmar transacción
COMMIT;

END //
-- fin de su solución
DELIMITER ;
```

4. Trigger.

```
DELIMITER //
-- incluya su solución a continuación

CREATE OR REPLACE TRIGGER t_limitar_unidades_mensuales_de_productos_fisicos
BEFORE INSERT ON lineaspedido
FOR EACH ROW
BEGIN

DECLARE unidades_vendidas_mes INT;
DECLARE tipo_producto INT;
DECLARE fecha DATE;

SELECT pr.tipoProductoId INTO tipo_producto
FROM productos pr
WHERE pr.id=NEW.productoId;

IF (tipo_producto = 1) THEN

SELECT p.fechaRealizacion INTO fecha
FROM pedidos p
WHERE p.id=NEW.pedidoId;


SELECT SUM(lp.unidades) INTO unidades_vendidas_mes
FROM lineaspedido lp
WHERE lp.productoId=NEW.productoId AND MONTH(CURDATE()) = MONTH(fecha) AND YEAR(CURDATE()) = YEAR(fecha);

    IF((NEW.unidades+unidades_vendidas_mes)>1000) THEN
        SIGNAL SQLSTATE '45002' SET MESSAGE_TEXT = 'No se pueden comprar tantas unidades de este producto';
    END IF;


END IF;
END //
-- fin de su solución
DELIMITER ;
```