Procedimientos almacenados y disparadores, ¿para qué son necesarios?

NOMBRE Y APELLIDOS: _____

La empresa Industria del Cubito ya dispone de la base de datos que hemos construido durante el transcurso de la asignatura **Bases de datos para *Data Warehousing***. Tras poner en funcionamiento su sistema y hacer pruebas con los distintos escenarios que se pueden dar (procesos), se ha puesto nuevamente en contacto con nosotros para que implementéis una serie de mejoras, sobre las que nos ha enviado los requisitos.

Para la propuesta de solución de esta PEC, debéis de crear una base de datos nueva denominada **dbdw_pec3.** Primero ejecutar el script adjunto **BBDD_ERP_structure.sql**. A continuación, ejecutar el script **BBDD_ERP_data.sql.** Estos dos scripts SQL crearán la estructura y el conjunto de datos necesarios para el desarrollo de esta PEC.

Consideraciones para la entrega y realización de la PEC:

- Todo lo que se pide en esta PEC está explicado en los bloques didácticos 2 y 3 (salvo que se trate de un ejercicio de investigación, cuyo enunciado lo especificará). No es necesario adelantar el estudio del material de otros bloques didácticos para la realización de esta PEC.
- En esta PEC trabajaremos procedimientos/funciones y disparadores. Al tratarse de objetos más complejos, debéis de asegurar una correcta ejecución de estos. Se recomienda que probéis de forma exhaustiva los componentes programados y os creéis vuestros casos de prueba utilizando la base de datos proporcionada.
- Se recomienda la utilización de **pgAdmin** para la implementación de toda la PEC. Existe otra alternativa que es ***psql*** (línea de comandos), pero es preferible que utilicéis pgAdmin ya que es una interfaz gráfica que os permitirá editar y crear sentencias SQL (así como mostrar los resultados) de forma más sencilla que *psql*.
- Tal y como se indica en el enunciado, cada respuesta a los ejercicios ha de entregarse en un fichero *.sql* diferente, con el nombre correspondiente. Se evaluará el código entregado en estos ficheros *.sql* y **NO el código que aparezca en el documento o en los pantallazos adjuntos**.
- Las capturas de pantalla de los ejercicios (y explicaciones pertinentes) han de proporcionarse en un documento aparte (se proporciona una plantilla para el caso, **indicad vuestro nombre en el documento**, por favor).
- Se debe de realizar la entrega de todos los ficheros de la PEC (tanto los ficheros *.sql* como el documento con explicaciones y capturas de pantalla) en un fichero comprimido *.zip*.

Consideraciones para la evaluación del ejercicio:

- Se tendrá en cuenta la aplicación de las buenas prácticas de codificación en SQL, de consultas y de programación de procedimientos y disparadores. Es decir: código con sangrado, uso de cláusulas SQL de forma correcta, comentarios, cabeceras en el procedimiento, etc.
- Los *scripts* proporcionados por el estudiante con las soluciones de los ejercicios han de ejecutarse correctamente. El estudiante ha de asegurarse de que lanzando el *script* completo de cada ejercicio no produzca ningún error.
- **Importante**: Las sentencias SQL proporcionadas en los *scripts* han de ser creadas de forma manual y no mediante asistentes que PostgreSQL/pgAdmin puedan proporcionar. Se pretende aprender SQL y no la utilización de asistentes.
- Las sentencias SQL proporcionadas en los ejercicios han de ser **solamente** aquellas que pide el enunciado y ninguna otra más. Cualquier sentencia añadida a mayores, si está mal o provoca que el *script* no se ejecute correctamente a la hora de corregirlo, penalizará el ejercicio.

EJERCICIO 1 (35%)

Desde la empresa Industria del Cubito se nos ha pedido una serie de mejoras sobre el esquema ***erp*** para tener controlados los cambios que se produzcan en algunas tablas:

a) (5%) En la tabla de surtidores (*tb_pump*) se quiere tener controlada la fecha en la que se produzca cualquier cambio. Para ello necesitaremos añadir un nuevo campo "updated_dt_tm" de tipo *timestamp,* que nos permita almacenar el valor de la fecha y hora del cambio.
Este valor debe actualizarse de forma automática (es decir, por defecto) cada vez que se inserte una nueva fila de información y no debe permitir almacenar valores nulos.

b) (10%) Para la tabla de líneas de factura también se pretende capturar el momento en que se crea o modifica el registro, pero con un formato específico.
Para ello, primero se pide crear el campo "line_updated_dt_tm" sobre la tabla de líneas de factura (*tb_lines_invoice*), de tipo *varchar(20)*.
Y, en segundo lugar, se tiene que implementar la función *fn_line_inserted* para rellenar el atributo "line_updated_dt_tm" con la fecha y hora actual del sistema, pero siguiendo el **formato "yyyy-mm-dd hh:mm"** (por ejemplo, si el cambio se produce el 15/Oct/2022 a las 11:40h, se guardará el valor "2022-10-15 11:40").
Esta función será ejecutada por el disparador *tg_line_inserted* siempre que se introduzca un nuevo registro o se modifique alguno existente, el cual tendréis que crear.

c) (20%) Por cumplir con los requerimientos de una auditoría externa, se pide crear la función *fn_invoice_updated* con el objetivo de actualizar de forma automática el valor de nuevos atributos a añadir en la tabla de facturas (*tb_invoice*), siempre que se produzca un cambio en cualquier línea de esa factura o se añada una nueva línea de factura para esa factura.

Para ello, primero será necesario crear tres nuevos atributos en la tabla de facturas (*tb_invoice*), con la siguiente lógica:

- "inv_updated_dt": de tipo date y que permita almacenar valores nulos. Será una copia del campo "line_updated_dt_tm" creado en el apartado anterior para la tabla *tb_lines_invoice*. Fijaros que cambia el tipo de dato, ahora queremos *date* (solo fecha).

- "inv_update_counter": de tipo entero y que permita almacenar valores nulos, con valor 0 por defecto. El valor de esta columna será un contador de las veces que se haya modificado la misma factura (misma fila de la tabla *tb_invoice*).

- "inv_insert_counter": de tipo entero y que permita almacenar valores nulos, con valor 0 por defecto. El valor de esta columna será un contador de las veces que se haya insertado una línea para la misma factura (misma fila de la tabla *tb_invoice*). Es decir, deberá coincidir con el número de líneas de la factura.

Es necesario que la función actualice los nuevos atributos mediante sentencias *update*.

Esta función será ejecutada por el disparador *tg_invoice_updated*, el cual tendréis que implementar. Recordad que se tendrá que ejecutar siempre que se inserte o actualice una fila en la tabla *tb_lines_invoice*.

Nota: utilizad cuando sea necesario las denominadas variables especiales de PostgreSQL, que están disponibles para los *trigger procedures*.

El código de este ejercicio se tiene que entregar en un fichero llamado **PEC3_Ej1.sql.**

EJERCICIO 2 (55%)

La empresa nos ha solicitado también nuevas funcionalidades para poder hacer un mejor seguimiento y control del gasto, pudiendo así planificar de forma más adecuada la demanda de combustible que requerirá para su flota.

a) (10%) En primer lugar se requiere añadir una columna calculada "rf_cost" para almacenar el coste de cada repostaje en la tabla *tb_refueling*. Para tal fin, tendremos que buscar el precio del litro para el día del repostaje en la tabla de precios (*tb_fuel_price*), teniendo en cuenta el tipo de motor del vehículo (diésel o gasolina).
Para ello habrá que crear la función "fn_calc_cost" que se encargará de hacer el *update* tras calcular el coste de ese repostaje, siguiendo la fórmula:

[coste = litros X precio del gasoil/gasolina del día]

Así mismo se requiere la creación de un trigger "tg_refueling_cost" para calcular automáticamente el valor de "rf_cost" en cada nuevo repostaje (es decir, en cada fila nueva que se inserte en esta tabla).

b) (20%) Implementa una nueva función "*fn_get_cost_by_car_type"* en la que se obtengan el número de litros y el coste asociado a los repostajes de un tipo de vehículo en un rango de días.

El procedimiento tiene las siguientes características:

- Debe recibir como parámetros:
  - Fechas de inicio y fin del rango a estudiar (nombres de las variables: *initial_date* and *final_date*)
  - Tipo de vehículo (nombre de la variable: *car_function*)

- Debe devolver todos los repostajes realizados por el tipo de vehículo seleccionado en el periodo solicitado. Será una tabla con los siguientes campos:
  - Tipo vehículo (cadena de 20 caracteres)
  - Fecha repostaje (formato fecha)
  - Cantidad de litros repostados (número entero)
  - Coste total del repostaje (número decimal)

- Respecto al control de posibles errores del procedimiento, hay que informar claramente con el siguiente mensaje cuando se produzca un error al introducir el valor del parámetro "car_function", por superar los 20 caracteres:
  - Longitud excesiva del valor introducido para el tipo de vehículo

c) (25%) Con el fin de mejorar la planificación de la demanda de litros que requerirá la flota de vehículos de nuestra empresa, así como el coste que conllevará, se nos pide crear un procedimiento que inserte en una nueva tabla "tb_invoice_cost_summary" un listado agrupado para cada vehículo con la siguiente información:

- Matrícula del vehículo
- Tipo de fuel (gasolina o diésel)
- Año (*inv_date_start*): 1900…2200
- Trimestre (*inv_date_start*): 1, 2, 3 o 4
- Mes (*inv_date_start*): 1...12 (formato año/mes. Por ejemplo: 2022/8)
- Litros repostados
- Coste
- Número de líneas de factura
- Fecha de la inserción (*timestamp*)

Esto se traduce en la siguiente tabla, cuya estructura completa se especifica a continuación:

Tabla: tb_invoice_cost_summary **Esquema:** erp				
Nombre columna	**Tipo de datos**	**Acepta Nulos**	**Clave primaria**	**Valores a guardar (ejemplos)**
cars_registration	Cadena de 10 caracteres variable	No	No	0019GVM
cars_fuel	Cadena de 10 caracteres variable	No	No	Gasolina o gasoil
invoice_year	Entero	No	No	2016
invoice_quarter	Entero	No	No	1, 2, 3, 4
invoice_month	Cadena de 10 caracteres variable	No	No	2016/1
total_liters	Entero	No	No	*394 (suma del total de litros de todas las líneas)*
total_cost	Número de 10 dígitos y 2 decimales.	No	No	787.19 *(suma del coste de todas las líneas)*
number_of_lines	Entero	No	No	6 *(cantidad de líneas de factura)*
timestamp	Fecha y hora	No	No	2022-09-02 21:42:23.741903 (momento en que se ha realizado la inserción)

La creación del procedimiento ha de cumplir los siguientes requisitos:

- El procedimiento no recibirá ningún parámetro ni se requiere control de excepciones.
- Se ha de utilizar un bucle FOR para obtener cada uno de los diferentes vehículos de las facturas *cars_registration* del detalle de las facturas (*tb_lines_invoice*).
- El año, trimestre y mes han de obtenerse mediante cálculos en el procedimiento a partir de la fecha *inv_date_start* (podéis revisar la documentación de PostgreSQL en línea)
- El número de litros repostados, el coste y el número de líneas de factura (*total_liters, cost* y *number_of_lines*) han de obtenerse mediante una consulta que devuelva dichos valores para los valores que estamos iterando en el bucle FOR: coche, tipo de fuel y fecha factura

El código de este ejercicio se tiene que entregar en un fichero llamado **PEC3_Ej2.sql.**

EJERCICIO 3 (10%)

En este ejercicio nos vamos a centrar en revisar e investigar conceptos que pueden ser de utilidad a la hora de gestionar nuestra base de datos.

Se pide dar respuesta a las siguientes dos cuestiones:

a) **(5%)** A medida que va creciendo nuestra base de datos nos damos cuenta de que es necesario tener monitorizados los objetos que vamos añadiendo sobre ella. Centrándonos en los *triggers*, queremos saber cuántos tenemos definidos en un momento concreto. ¿Cómo podemos obtener esta información?

   Comenta brevemente la respuesta y acompáñala del código necesario y una captura de los *triggers* que tienes creados en tu base de datos.

b) **(5%)** Investiga sobre la vistas y tablas internas de PostgreSQL. Explica brevemente su significado general, y detalla un par de tablas y/o consultas que te resulten de utilidad relacionadas con el “Statistics Collector” (**máximo 1 página**)

Criterios de valoración

En el enunciado se indica el peso/valoración de cada ejercicio.

Para conseguir la puntuación máxima en los ejercicios, es necesario explicar con claridad la solución que se propone.

Formato y fecha de entrega

Tenéis que enviar la PEC al buzón de Entrega y registro de EC disponible en el aula (apartado Evaluación). El formato del archivo que contiene vuestra solución puede ser **.pdf, .doc y .docx**. **Para otras opciones, por favor, contactar previamente con vuestro consultor**. El nombre del fichero debe contener el código de la asignatura, vuestro apellido y vuestro nombre, así como el número de actividad (PEC3).

La fecha límite para entregar la PEC3 es el **16/12/2022**.

Nota: **Propiedad intelectual**

Al presentar una práctica o PEC que haga uso de recursos ajenos, se tiene que presentar junto con ella un documento en que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL etc.). El estudiante tendrá que asegurarse que la licencia que sea no impide específicamente su uso en el marco de la práctica o PEC. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por el copyright.

Será necesario, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente, si así corresponde.