Universidad Nacional Autónoma de México
Facultad de Ciencias

Fundamentos de Bases de Datos

Práctica 6

Funciones, Procedimientos Almacenados, Triggers y Vistas

Autor: Arroyo Martínez Erick Daniel

Introducción

En esta práctica, los estudiantes tendrán la oportunidad de profundizar en el uso de funciones, procedimientos almacenados, triggers y vistas dentro de un sistema de bases de datos relacional. Utilizando el esquema definido, los alumnos deberán implementar diferentes funcionalidades avanzadas que permitirán automatizar procesos, validar datos y garantizar la integridad referencial en la base de datos.

Además, se introducirá el concepto de vistas, que permitirá a los estudiantes abstraer y simplificar consultas complejas mediante la creación de vistas personalizadas. Estas vistas ayudarán a organizar y visualizar la información de manera más eficiente, permitiendo una mejor manipulación de los datos.

El esquema 1 utilizado se centra en la gestión de oficinas, empleados, clientes, órdenes, pagos y productos. Cada tabla está interrelacionada mediante claves foráneas que aseguran la consistencia de los datos a lo largo de todo el sistema. Los alumnos deberán utilizar estas relaciones para crear vistas que permitan consultar la información de manera eficiente, además de implementar procedimientos, funciones y triggers para garantizar la integridad de la base de datos.

Objetivos

- Implementar funciones, procedimientos almacenados y triggers en un entorno de base de datos realista y complejo.
- Introducir y manejar vistas para simplificar el acceso a información derivada de varias tablas.
- Automatizar la validación de datos y las reglas de negocio mediante triggers.
- Manipular y presentar datos utilizando funciones que realicen cálculos específicos sobre las tablas.
- Garantizar la consistencia y coherencia de los datos mediante la correcta implementación de integridad referencial.

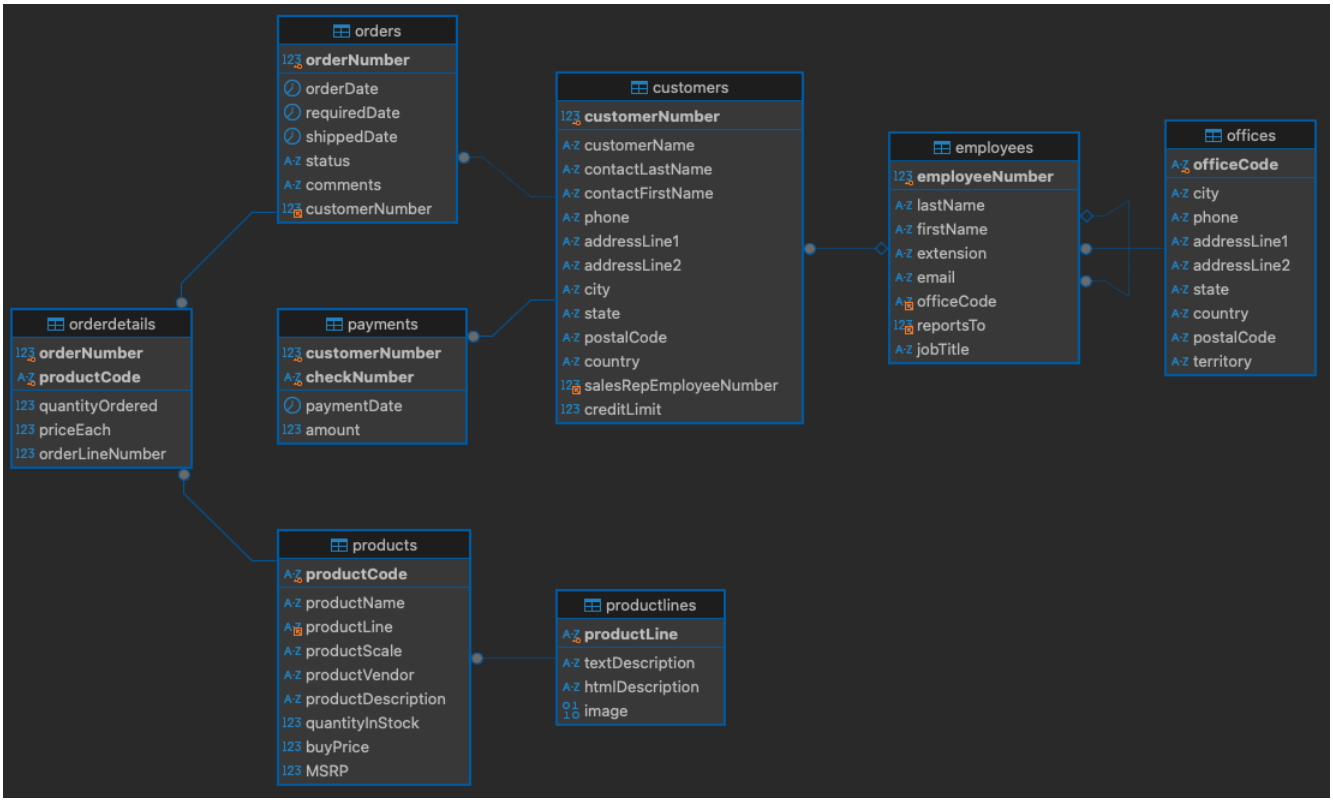

Figura 1: Esquema Relacional

Vistas

¿Qué es una vista?

Una **vista** es una tabla virtual basada en el resultado de una consulta a una o más tablas. Aunque una vista no almacena datos por sí misma, se comporta como una tabla cuando se consulta, y presenta los datos de manera simplificada o filtrada. Las vistas son útiles para proporcionar un acceso controlado y organizado a los datos, permitiendo una abstracción de la complejidad subyacente del esquema de la base de datos.

Objetivo de las vistas

El uso de vistas tiene varios objetivos:

- Simplificar el acceso a datos complejos mediante la combinación de varias tablas.
- Restringir el acceso a determinadas columnas o filas de una tabla para usuarios con permisos limitados.
- Mejorar la seguridad y el control del acceso a los datos.
- Facilitar la reusabilidad de consultas complejas, eliminando la necesidad de repetir las mismas consultas en diferentes partes del sistema.
- Mejorar el rendimiento de consultas frecuentes que no requieran modificaciones frecuentes de los datos subyacentes.

Sitaxis básica

```
CREATE VIEW nombre_de_la_vista AS
SELECT columnas
FROM tabla(s)
WHERE condiciones;
```

Donde:

- `nombre_de_la_vista`: Es el nombre que se le da a la vista, que se utilizará posteriormente para referirse a ella.
- `columnas`: Especifica las columnas que aparecerán en la vista.
- `tabla(s)`: Indica la tabla o tablas de donde se obtienen los datos.
- `condiciones`: Define las condiciones que filtran los datos que se mostrarán en la vista.

A continuación se muestra un ejemplo de cómo crear una vista para listar todos los empleados junto con la ciudad y el país de la oficina en la que trabajan:

Ejemplo Concreto

```
CREATE VIEW EmployeeOfficeInfo AS
SELECT e.firstName, e.lastName, o.city, o.country
FROM employees e
JOIN offices o ON e.officeCode = o.officeCode;
```

En este ejemplo:

- La vista `EmployeeOfficeInfo` muestra los nombres de los empleados junto con la ciudad y el país de la oficina a la que están asignados.
- Se utiliza una `JOIN` para unir las tablas `employees` y `offices` mediante la columna `officeCode`.

Al consultar esta vista con la instrucción `SELECT`, el resultado sería similar a una tabla real, pero basada en los datos de las dos tablas subyacentes.

Especificaciones de Desarrollo

Para esta práctica, se desarrollarán cuatro funciones, cuatro procedimientos almacenados y cuatro vistas y triggers. Cada uno de estos elementos deberá cumplir con los siguientes requisitos y estar interrelacionados para simular un entorno de negocio realista y complejo. Los alumnos deberán implementar cada elemento utilizando el esquema proporcionado de la base de datos.

Funciones

1. **TotalOrders**. Deberá devolver el total de órdenes realizadas por ese cliente en el año proporcionado.
2. **TotalOrdersDiscount**: Esta función recibirá como parámetros el `orderNumber` y calculará el descuento total basado en la diferencia entre el precio sugerido (`MSRP`) y el precio real de cada producto ordenado en esa orden.
3. **IsOffManager**: Devolverá un valor booleano indicando si ese empleado es el gerente de la oficina asociada.

4. **GetTotalPayments**: Devolverá el total de pagos realizados por ese cliente durante el rango de fechas proporcionado.

Procedimientos Almacenados

1. **Procedimiento para registrar una nueva orden (RegisterNewOrder)**: Este procedimiento recibirá los detalles necesarios para registrar una nueva orden, incluyendo el `customerNumber`, el `orderDate`, los productos y la cantidad de cada uno. El procedimiento debe crear la orden y actualizar el inventario de los productos involucrados.

2. **Procedimiento para actualizar la el estado de un cliente (UpdateCustomerState)**: El procedimiento debe recibir el `customerNumber` y actualizar el estado del cliente a **Deactivate** sino tiene ordenes pendientes asociadas, además sino ha realizado ordenes en el último año debe actualizar su representante de ventas a **NULL**, asegurando que las actualizaciones no rompan la integridad referencial con otros registros.

3. **Procedimiento para asignar un empleado como representante de ventas (AssignSalesRepToCustomer)**: Este procedimiento recibirá el `employeeNumber` y el `customerNumber`, y asignará ese empleado como representante de ventas de dicho cliente. Si el cliente ya tiene un representante de ventas asignado, deberá actualizarse al nuevo empleado.

4. **Procedimiento para calcular el total de ventas por empleado en un rango de fechas (TotalSalesByEmployee)**: Este procedimiento recibirá un rango de fechas y un `employeeNumber`, y devolverá el total de ventas que ese empleado ha generado en ese período.

Triggers

1. **Trigger para evitar la eliminación de productos con órdenes pendientes (PreventProductBD)**: Este trigger debe activarse antes de intentar eliminar un producto y verificar si ese producto está involucrado en alguna orden que aún no ha sido enviada. Si es así, debe lanzar un error y cancelar la eliminación.

2. **Trigger para actualizar la cantidad de productos en inventario después de una venta (ProductInventorySaleAU)**: Este trigger se debe activar después de insertar una nueva orden. El propósito es actualizar la cantidad disponible de los productos en el inventario tras la venta de esos productos.

3. **Trigger para validar el límite de crédito de un cliente (ValidateCustomerCreditLimitBI)**: Este trigger se debe ejecutar antes de registrar una nueva orden. Si la suma del monto de la nueva orden más las órdenes previas excede el límite de crédito del cliente, el trigger debe cancelar la inserción de la orden.

4. **Trigger para actualizar el estado de un cliente a Activate (CustomerStatusBI)**: Antes de que se realice una orden, deberá verificar que si ya está activo no haga actualización.

5. **(+1pt) Trigger para actualizar el estado de los clientes a Suspended cuando todas sus ordenes han sido entregadas o su última orden fue hace más de un mes.**

Vistas

1. **Vista para mostrar el historial de órdenes de cada cliente (CustomerOrderHistoryView)**: Esta vista debe incluir el nombre del cliente, la cantidad de órdenes realizadas, el total de pagos recibidos, y el estado actual de cada una de sus órdenes.

2. **Vista para consultar el inventario de productos (ProductInventoryView)**: Debe mostrar el `productCode`, `productName`, `quantityInStock`, y el precio de cada producto disponible. Esta vista se usará para generar reportes de inventario.

3. **Vista para consultar las ventas por empleado (EmployeeSalesView)**: Esta vista debe mostrar el nombre completo del empleado, el número de clientes que tiene asignados, y el total de ventas generadas por ese empleado.

4. **Vista para consultar el detalle de órdenes (OrderDetailsView)**: Debe incluir información sobre cada orden, como el número de orden, la fecha de la orden, el cliente que la realizó, el total de productos ordenados, y el monto total de la orden.

Entregables

Los alumnos deberán entregar los siguientes elementos:

- Implementación de las cuatro funciones, procedimientos almacenados, triggers y vistas descritos.
- Un archivo SQL con la implementación completa del esquema y los elementos requeridos.
- Un **breve** informe que explique brevemente la interrelación entre las funciones, procedimientos, triggers y vistas, y cómo estos garantizan la consistencia e integridad de la base de datos.

Rúbrica de Evaluación

Criterio	**Puntos**
Implementación correcta de las funciones	20 %
Implementación correcta de los procedimientos almacenados	20 %
Implementación correcta de los triggers	20 %
Implementación correcta de las vistas	20 %
Claridad y presentación del código/documentación	20 %

Recursos

- Live SQL
- SQL Language Reference
- SQL Tutorial
- mockaroo
- MySQL Documentation