Administrador de base de datos [Nivel 2]
Lección 3 / Actividad 1

Procedimientos almacenados

IMPORTANTE

Para resolver tu actividad, **guárdala** en tu computadora e **imprímela**.

Si lo deseas, puedes conservarla para consultas posteriores ya que te sirve para reforzar tu aprendizaje. No es necesario que la envíes para su revisión.

Propósito de la actividad

Aplicar la sintaxis de procedimientos almacenados y de transacciones para mejorar el rendimiento y seguridad de una base de datos.

Practica lo que aprendiste

I. Para realizar esta actividad instala en tu servidor de SQL Server la base de datos llamada **AdventureWorks** siguiendo estos pasos:

1. Descarga el archivo con extensión **.bak** de la siguiente liga y guárdalo: https://fsvc.capacitateparaelempleo.org/CapacitateFS/storage/b69c2287-83db-4f76-b4a9-20c5eb78b547.zip
2. Abre el gestor de base de datos, da clic derecho sobre **Base de datos** y selecciona **Restaurar base de datos**.
3. En el campo llamado **A una base de datos**, escribe **AdventureWorks.**
4. Selecciona la opción **Desde dispositivo** y haz clic en el botón con los tres puntos **(…)**
5. Haz clic en el botón **Agregar** y localiza el archivo que descargaste con la extensión **.bak**
6. Una vez seleccionado, marca el cuadro de **Restaurar** y presiona **Aceptar.**

II. Escribe el código para que crees los siguientes procedimientos almacenados.

a) Crea un procedimiento almacenado que devuelva todos los empleados (nombre y apellidos), sus puestos y el nombre del departamento al que corresponden a partir de una vista. No utilices ningún parámetro de entrada.

```
CREATE PROCEDURE EmployeeBasicData
AS BEGIN
    SELECT p.FirstName, p.MiddleName, p.LastName, e.JobTitle, d.Name AS Department
    FROM HumanResources.Employee e
    JOIN Person.Person p ON e.BusinessEntityID = p.BusinessEntityID
    JOIN HumanResources.EmployeeDepartmentHistory ed ON e.BusinessEntityID = ed.BusinessEntityID
    JOIN HumanResources.Department d ON ed.DepartmentID = d.DepartmentID
    ORDER BY Department ASC
   END
GO
```

b) Crea un procedimiento almacenado que sólo devuelva el empleado especificado (nombre y apellidos), su puesto y el nombre del departamento que corresponde a partir de una vista. Utiliza parámetros de entrada.

```
CREATE PROCEDURE EmployeeBasicDataByID @BusinessEntityID INT
AS BEGIN
   SELECT e.BusinessEntityID AS ID, p.FirstName, p.MiddleName, p.LastName, e.JobTitle, d.Name AS
Department
   FROM HumanResources.Employee e
   JOIN Person.Person p ON e.BusinessEntityID = p.BusinessEntityID
   JOIN HumanResources.EmployeeDepartmentHistory ed ON e.BusinessEntityID = ed.BusinessEntityID
   JOIN HumanResources.Department d ON ed.DepartmentID = d.DepartmentID
   WHERE e.BusinessEntityID = @BusinessEntityID
   ORDER BY Department ASC
  END
GO
```

c) Crea el procedimiento almacenado que devuelva una lista de productos con precios que no superen un importe especificado.

```
CREATE PROCEDURE ProductsLessEquals @MaximunPrice MONEY
AS BEGIN
   SELECT * FROM Production.Product p WHERE p.ListPrice <= @MaximunPrice
  END
GO
```

III. Escribe el código para que crees las siguientes transacciones.

a) Crea una transacción que haga un **INSERT** y un **UPDATE** a las tablas **HumanResourses.Employee**, agrega una condicional de error que entregue mensajes de **ERROR** o de **BIEN** si se hizo un **ROLLBACK** o un **COMMIT** respectivamente.

```
BEGIN TRANSACTION
  INSERT INTO HumanResources.Employee (BusinessEntityID, NationalIDNumber, LoginID,
OrganizationNode, JobTitle, BirthDate, MaritalStatus, Gender, HireDate, SalariedFlag,VacationHours,
SickLeaveHours, CurrentFlag, rowguid)
  VALUES (1, '295847284', 'adventure-works\ken0', '/', 'Executive Officer', '1984-03-02', 'S', 'M', '1999-02-20',
1, 99, 69, 1, 'f01251e5-96a3-448d-981e-0f99d789110d')
  IF (@@error <> 0) BEGIN
    ROLLBACK
    PRINT ('ERROR')
  END
  ELSE BEGIN
    COMMIT
    PRINT ('BIEN')
  END
GO
```

b) Crea una transacción dentro de un procedimiento almacenado que realice un **INSERT** a la tabla de **Production.Products**, el procedimiento debe tener parámetros de entrada y la transacción debe entregar mensajes de **ERROR** o de **BIEN** si se hizo un **ROLLBACK** o un **COMMIT** respectivamente.