Evaluación de Aprendizaje N°2

Base de datos 2

Alumno: Marcos Cabral

DNI: 42684487

Profesores: Osores Hernán. Pardieux Eliana.

Fecha de Entrega: 14/10/2020.

1)

Listar todos los préstamos realizados. Mostrar datos del préstamo, nombre y apellido de la persona, datos del libro, datos de la editorial y datos del autor.

```
create view v_listaPrestamosRealizados
as
select fechaPrestamo,fechaDevolucion, c.nombres  as 'Nombres Cliente',c.apellido as
'Apellido Cliente',
l.titulo as 'Libro',e.nombre as 'Editorial',a.nombres as 'Nombre Autor',a.apellido
as 'Apellido Autor'
from Biblioteca.Prestamo as p
inner join Biblioteca.Cliente as c
on c.codigoCliente=p.codigoCliente
inner join Biblioteca.LibroPrestamo as lp
on lp.numeroPrestamo=p.numeroPrestamo
inner join Biblioteca.Libro as l
on l.codigoLibro=lp.codigoLibro
inner join Biblioteca.Editorial as e
on e.codigoEditorial=l.codigoLibro
inner join Biblioteca.LibroAutor as la
on la.codigoLibro=l.codigoLibro
inner join Biblioteca.Autor as a
on a.codigoAutor=la.codigoAutor
;
```

Listar todos los libros que alguna vez fueron prestados.

```
create view v_LibrosPrestados
as
select l.titulo,l.importe,l.stock from Biblioteca.Libro as l
where l.codigoLibro in (select codigoLibro from Biblioteca.LibroPrestamo);
```

Listar los libros con sus respectivos autores

```
create view v_LibroYAutores
as
select titulo,importe,stock,nombres,apellido from Biblioteca.Libro as l
inner join Biblioteca.LibroAutor as la
on l.codigoLibro=la.codigoAutor
inner join Biblioteca.Autor as a
on la.codigoAutor=a.codigoAutor;
```

Listar los libros que fueron escritos por dos o más autores

```
create view v_LibrosEscritorPorDosOMasAutores
as
select * from Biblioteca.Libro
where codigoLibro =(
       select la.codigoLibro from Biblioteca.LibroAutor as la
       group by la.codigoLibro
       having count(la.codigoAutor)>=2
);
```

Listar los clientes que pidieron la mayor cantidad de libros y su apellido contenga la letra 'u'.

```
create view v_ClientesConMayorCantidadDeLibros
```

```
as
select c.codigoCliente,c.nombres,c.apellido,count(lp.codigoLibro) as 'Libros
retirados' from Biblioteca.Cliente as c
inner join Biblioteca.Prestamo as p
on p.codigoCliente=c.codigoCliente
inner join Biblioteca.LibroPrestamo as lp
on p.numeroPrestamo=lp.numeroPrestamo
where c.apellido like '%u%'
group by c.codigoCliente,c.nombres,c.apellido
having count(lp.codigoLibro)=(
       select max(maximo.cantidadLibros)  from(
              select count(codigoLibro) as 'cantidadLibros' from
Biblioteca.LibroPrestamo as lp
                     inner join Biblioteca.Prestamo as p
                     on p.numeroPrestamo=lp.numeroPrestamo
                     inner join Biblioteca.Cliente as c
                     on p.codigoCliente=c.codigoCliente
                     where c.apellido like '%u%'
                     group by c.codigoCliente
              ) as maximo
);
```

Listar los autores cuyos libros han sido prestados por más de dos días.

```
create view v_autoresConLibrosPrestadosMasDeDosDias
as
select distinct a.nombres, a.apellido from Biblioteca.Prestamo as p
inner join Biblioteca.LibroPrestamo as lp
on lp.numeroPrestamo=p.numeroPrestamo
inner join Biblioteca.LibroAutor as la
on lp.codigoLibro=la.codigoLibro
inner join Biblioteca.Autor as a
on a.codigoAutor=la.codigoAutor
where DATEDIFF(day,p.fechaPrestamo,p.fechaDevolucion) >=2;
```

2)

```
CREATE PROC p_ValidaLibros
as
SELECT titulo,codigoLibro,Validacion=(
CASE WHEN titulo not like '%[^a-zA-Z0-9]%' THEN 1
        WHEN importe <50.5 or importe > 15500 THEN 2
        WHEN exists (
              SELECT titulo
              FROM Biblioteca.Libro
              GROUP BY titulo
              HAVING COUNT (titulo)>1) THEN 3
        ELSE 0
END)
FROM Biblioteca.Libro;
```

3) Para crear el procedimiento genere constraints en las tablas Préstamo, y LibroPrestamo para borrar en cascada a los préstamos en los que se encuentre el cliente, y posteriormente la relación nn de LibroPrestamo, donde se encuentran estos códigos de préstamos mencionados anteriormente.

Las sentencias fueron:

```
ALTER TABLE Biblioteca.LibroPrestamo
ADD CONSTRAINT cascada_prestamo FOREIGN KEY(numeroPrestamo)
REFERENCES Biblioteca.Prestamo (numeroPrestamo)
ON DELETE CASCADE
GO
ALTER TABLE Biblioteca.Prestamo
ADD  CONSTRAINT cascada_cliente FOREIGN KEY(codigoCliente)
REFERENCES Biblioteca.CLiente (codigoCliente)
ON DELETE CASCADE
```

-Yo asumo que si un cliente realizó 2 préstamos, uno hace 7 años, y otro el mes pasado, no será eliminado. Esto pasaría solo para los clientes que están en esta situación antes de ejecutar el procedimiento, con los nuevos que lleven más de 6 años se eliminarían.

```
CREATE PROC p_DardeBajaClientes
AS
BEGIN
       UPDATE Biblioteca.Cliente
              SET fecha_baja = GETDATE()
              WHERE codigoCliente not in(
                     SELECT codigoCliente FROM Biblioteca.Prestamo
                     WHERE DATEDIFF(year,fechaPrestamo,GETDATE()) <=2
              ) and fecha_baja is null;

       DELETE FROM Biblioteca.Cliente
       WHERE codigoCliente not in(
              SELECT c.codigoCliente
              FROM Biblioteca.Prestamo as p
              inner join Biblioteca.cliente as c
              on p.codigoCliente = c.codigoCliente
                     WHERE DATEDIFF(year,fechaPrestamo,GETDATE()) < 6
              )
              or codigoCliente in (
              SELECT codigoCliente
```

```
            FROM Biblioteca.Cliente
            WHERE DATEDIFF(year,fecha_baja,GETDATE()) > 1
            )
END;
```

4)

```
CREATE PROC p_TotalPrestamos @fecha date, @cantidad int OUTPUT
as
SELECT @cantidad=count(*)
FROM Biblioteca.Prestamo as p
WHERE fechaPrestamo = @fecha;
---ver resultados, devuelve 1 préstamo ese día ---
DECLARE @cant int
EXEC p_TotalPrestamos '2020-10-01',@cant OUTPUT
SELECT @cant
```

Funciones

5)

```
CREATE FUNCTION Biblioteca.f_NumeroPrestamo()
RETURNS int
as BEGIN
      DECLARE @num int
      SELECT @num= MAX(numeroPrestamo)
      FROM Biblioteca.Prestamo;
      RETURN @num
END;

--resultado--

SELECT Biblioteca.f_NumeroPrestamo();
```

6)

```
CREATE FUNCTION Biblioteca.f_cuit(@cuit BIGINT)
RETURNS VARCHAR(40)
as BEGIN
      DECLARE @cuitFormateado VARCHAR(40)
      SELECT @cuitFormateado = FORMAT(@cuit,'##-########-#')
      RETURN @cuitFormateado
end;
--resultado--

SELECT Biblioteca.f_cuit(20426844878);
```

7)

```
CREATE FUNCTION Biblioteca.f_soloLetras(@texto varchar(100))
RETURNS varchar(100)
as
BEGIN
      DECLARE @textoADevolver varchar(100)
      SET @textoADevolver='%[^a-z]%'
       WHILE PatIndex(@textoADevolver, @texto) > 0
          SET @texto = Stuff(@texto, PatIndex(@textoADevolver, @texto), 1, '')
```

```
	RETURN @texto
END;

--resultado--

SELECT Biblioteca.f_soloLetras('hola es´te es + de un texto {} de prueba !! 12 -.');
8)

CREATE FUNCTION Biblioteca.f_quitarEspacios(@texto varchar(100))

RETURNS varchar(100)
as
BEGIN
	DECLARE @txt varchar(100)
	SET @txt=replace(@texto,' ','')
	RETURN @txt
END;

--resultado--

SELECT Biblioteca.f_quitarEspacios('    hola estoy usando muhcos es   pa ci o   s');


9)

CREATE FUNCTION Biblioteca.f_cantidadPrestamos(@codigo int)
RETURNS int
as BEGIN
	DECLARE @cant int
		SELECT @cant=count(*)
		FROM Biblioteca.Prestamo as p
		WHERE p.codigoCliente = @codigo
	RETURN @cant
END;

--resultado--

SELECT Biblioteca.f_cantidadPrestamos(5);

10)

use Biblioteca;

create login sis_biblio with password='sis_biblio';
create user sis_biblio for login sis_biblio;
CREATE APPLICATION ROLE [sis_biblio_role] WITH DEFAULT_SCHEMA = [Biblioteca],
PASSWORD = N'sis_biblio_role'
grant select on Biblioteca.Prestamo to  sis_biblio_role
grant exec on p_DardeBajaClientes to sis_biblio_role;
grant select,update, delete on object::v_LibrosPrestados to sis_biblio_role;

use Biblioteca;
exec sp_setapprole 'sis_biblio_role', 'sis_biblio_role';
select * from Biblioteca.Prestamo;

11)
```

```
use Biblioteca;
create login sis_biblio_dba with password='sis_biblio_dba';
use master;
deny view any database to sis_biblio_dba;
use Biblioteca;
alter authorization on database:: Biblioteca to sis_biblio_dba;
12)

use master;
create login sis_dba with password='sis_dba';
use Biblioteca;
create user sis_dba for login sis_dba;
ALTER AUTHORIZATION ON SCHEMA::[db_accessadmin] TO [sis_dba]
ALTER ROLE [db_accessadmin] ADD MEMBER [sis_dba]
```

13)

Validación del punto 10 – sis_biblio:

Si ejecuto la vista, el procedimiento y la tabla préstamo nos muestra correctamente. En cambio sí accedo a otra vista, tabla o procedimiento tira error, como las últimas imágenes.

```
select * from Biblioteca.Prestamo;
select * from v_LibrosPrestados;
exec p_darDeBajaClientes;
```

	numeroPrestamo	fechaPrestamo	fechaDevolucion	codigoCliente
1	2	2020-09-19	2020-09-29	2
2	3	2020-10-01	2020-11-01	3
3	6	2017-05-06	2017-05-12	5
4	7	2019-07-12	2019-08-01	5
5	8	2010-01-22	2010-01-31	6
6	9	2019-07-10	2019-07-30	6

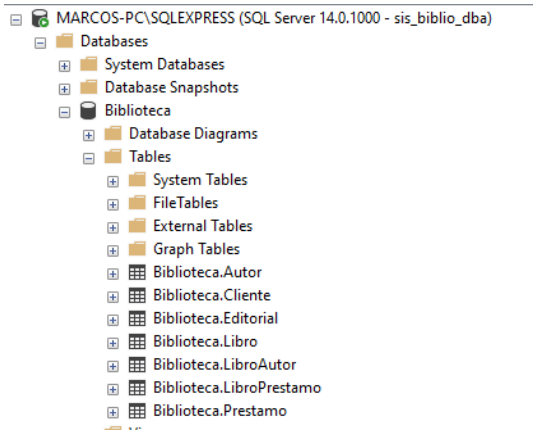
	titulo	importe	stock
1	Libro de cortazar 1	400	570

```
select * from Biblioteca.Libro;
select * from v_autoresConLibrosPrestadosMasDeDosDias
```

```
The SELECT permission was denied on the object 'Libro', database 'Biblioteca', schema 'Biblioteca'.
Msg 229, Level 14, State 5, Line 4
The SELECT permission was denied on the object 'v_autoresConLibrosPrestadosMasDeDosDias', database 'Biblioteca', schema 'dbo'.
```

Validación del punto 11 – sis_biblio_dba:

Como se observa, al iniciar sesión con el usuario **sis_biblio_dba** la única base de datos que aparece es Biblioteca.

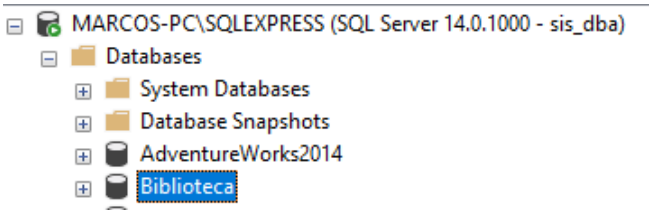


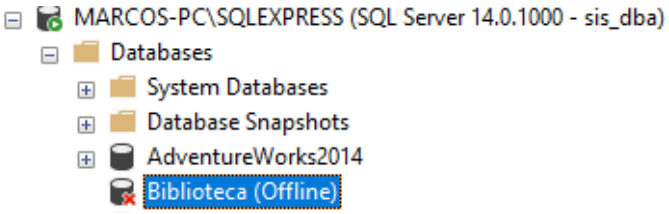
Validación del punto 12 – sis_dba:

Para validar un db_accesadmin utilice las mismas consultas dadas en las prácticas:

alter database Biblioteca set offline

alter database Biblioteca set online





14. Si tuviera que conectarme a una base de datos desde mi código, lo haría a través de procedimientos. Esto lo haría debido a que los procedimientos tienen un mejor manejo de errores, y lleva a cabo el uso de las transacciones. Lo cual permite que si hay una excepción generada por un error al momento de conectarse a la base de datos.

De esta manera se podría capturar el error y manejarlo de forma transaccional, sin proseguir con una acción concreta en el código si algo salió mal, pero sin tener que cerrar el programa.

Además si necesitamos de lógica para la conexión, las funciones no tienen la capacidad de manipular datos de forma estructurada o en bucle (IF, WHILE).

15) Yo recomendaría utilizar procedimientos almacenados para esta situación, ya que nos permite tener una capa adicional de seguridad. Esto permite que si tenemos un usuario para la aplicación, y solo le asignamos el permiso de ejecutar el procedimiento, dicho usuario no podrá acceder directamente a las tablas que tenemos por debajo o a la estructura completa de la base de datos.

Otros puntos para resaltar son que el procedimiento controla que procesos y actividades se llevan a cabo. Esto elimina la necesidad de conceder permisos en cada nivel de objetos, simplificando así los niveles de seguridad.

Y finalmente el uso de parámetros ayuda a protegernos contra ataques por inyección de SQL.

Por otro lado cabe mencionar que se puede utilizar vistas, ya que estas permiten a los usuarios obtener acceso a los datos mediante esta, PERO no les conceden el permiso de obtener acceso directo a las tablas base de la vista. Permiten emular una tabla que existía pero esquema haya cambiado