UNIVERSIDAD ESTATAL A DISTANCIA

ESCUELA DE CIENCIAS EXACTAS Y NATURALES

CÁTEDRA DE INGENIERÍA DE SOFTWARE

Base de datos

Código: 00826

Centro Universitario: San José

Grupo: 01

Estudiante: Andrés Eduardo Balladares Flores

Cédula: 1-1852-0229

Primer cuatrimestre, 2022

Contenido

Introducción

Anteriormente, en la base de datos de la taxonomía de seres vivos, se observó cómo crear las tablas, ingresar datos a estas y manipular dichos datos, por medio de select, delete y update. Estas son las funciones básicas de SQL, con las cuales se pueden crear bases de datos muy grandes. Sin embargo, mientras más grande es la cantidad de información, menos efectivo es utilizar solo estas sentencias. Para ello, se tienen las funciones y los procedimientos almacenados.

Por definición, tanto las funciones como los procedimientos almacenados son un conjunto de instrucciones que realizan una tarea específica de manera automática. Por ejemplo, si tiene que escribir repetidamente grandes scripts SQL para realizar la misma tarea, se puede crear una función que realice esa tarea. Por ello, la próxima vez, en lugar de reescribir el SQL, de manera repetida lo que se puede hacer es simplemente llamar a esa función.

En ello radica la importancia de las funciones y procedimientos, estos facilitan la manipulación de las tablas y, en caso de necesitarse, solo se llaman y realizan la función necesaria. En este trabajo, se realizará una función y un procedimiento almacenado en la base de datos de la taxonomía de los seres vivos, con los cuales se simplificarán los procesos que se realizaron en el trabajo anterior. La función se encargará de obtener el reino de cualquier ser vivo que se ingrese, mientras que el procedimiento ingresará nuevos seres vivos a la base de datos.

Al finalizar este trabajo, se podrá aplicar funciones y procedimientos en cualquier base de datos en SQL para poder crear código de programación que permita ejecutar diferentes tareas a nivel de la base de datos.

Desarrollo

Una función es una operación denotada por un nombre, seguido por uno o más operandos que se incluyen entre paréntesis, con las cuales se ingresan argumentos y se retorna un valor. Se tienen las funciones incorporadas en el sistema y aquellas definidas por el usuario

Los procedimientos almacenados son muy similares en concepto. Según Álvarez (2019), estos son instrucciones que se almacenan en la base de datos para ser ejecutadas por otras consultas. También se presentan dos tipos, aquellos incorporados en el sistema y los generados por el usuario. Implementar este tipo de sentencias en SQL presenta las siguientes ventajas:

- Fomentan la reutilización de código.
- Mejoran la velocidad de consultas.
- Aumentan la seguridad.
- Optimizan el mantenimiento.
- Incrementan el rendimiento.

A continuación, se creará la base de datos con sus respectivas tablas y se insertará información a estas, para poder crear la función y el procedimiento respectivos.

Creación de las tablas e inserción de datos

El primer paso es la creación de las tablas y la inserción de datos, debido a que no se podría realizar las siguientes funciones sin la estructura inicial. El script de esta sección está basado en los trabajos anteriores y es el siguiente:

Script de creación de tablas e inserción de datos

```
create database AndresBalladaresProyecto3

use AndresBalladaresProyecto3



--PARTE 1: creación de las tablas e inserción de datos.

--Esta parte inserta en la base de datos la información creada en el proyecto anterior.



create table Reino(

	IDReino varchar(30) primary key NOT NULL
```

```
)


create table Phylum(
        IDPhylum varchar(30) primary key NOT NULL,
        Reino varchar(30),
        CONSTRAINT fk_Reino FOREIGN KEY (Reino) REFERENCES Reino(IDReino)
)


create table Clase(
        IDClase varchar(30) primary key NOT NULL,
        Phylum varchar(30),
        CONSTRAINT fk_Phylum FOREIGN KEY (Phylum) REFERENCES Phylum(IDPhylum)
)


create table Orden(
        IDOrden varchar(30) primary key NOT NULL,
        Clase varchar(30),
        CONSTRAINT fk_Clase FOREIGN KEY (Clase) REFERENCES Clase(IDClase)
)


create table Familia(
        IDFamilia varchar(30) primary key NOT NULL,
        Orden varchar(30),
```

```
	CONSTRAINT fk_Orden FOREIGN KEY (Orden) REFERENCES Orden(IDOrden)

)



create table Genero(

	IDGenero varchar(30) primary key NOT NULL,

	Familia varchar(30),

	CONSTRAINT fk_Familia FOREIGN KEY (Familia) REFERENCES Familia(IDFamilia)

)



create table Especie(

	IDEspecie varchar(30) primary key NOT NULL,

	Genero varchar(30),

	Nombre_Cientifico varchar(45) NOT NULL,

	Nombre_Comun varchar(45) NOT NULL,

	CONSTRAINT fk_Genero FOREIGN KEY (Genero) REFERENCES Genero(IDGenero)

)



--Si se desea eliminar las tablas, usar este código una vez:

--drop table Especie

--drop table Genero

--drop table Familia

--drop table Orden

--drop table Clase

--drop table Phylum
```

```
--drop table Reino



insert into Reino values
        ('Animal'),
        ('Vegetal');

insert into Phylum values
        ('Chordata', 'Animal'),
        ('Tracheophyta', 'Vegetal');

insert into Clase values
        ('Mammalia', 'Chordata'),
        ('Angiosperma', 'Tracheophyta');

Insert into Orden values
        ('Artiodactyla', 'Mammalia'),
        ('Primate', 'Mammalia'),
        ('Carnívora', 'Mammalia'),
        ('Glumifloral', 'Angiosperma'),
        ('Cetacea', 'Mammalia');

Insert into Familia values
        ('Bovidae', 'Artiodactyla'),
        ('Hominidae', 'Primate'),
```

```
        ('Canidae', 'Carnívora'),
        ('Felidae', 'Carnívora'),
        ('Gramínea', 'Glumifloral'),
        ('Balaenopteridae', 'Cetacea');

Insert into Genero values
        ('Bos', 'Bovidae'),
        ('Homo', 'Hominidae'),
        ('Canis', 'Canidae'),
        ('Felis', 'Felidae'),
        ('Zea', 'Gramínea'),
        ('Megaptera', 'Balaenopteridae');

Insert into Especie values
        ('Taurus', 'Bos', 'Bos Taurus', 'Vaca'),
        ('Sapiens', 'Homo', 'Homo Sapiens', 'Hombre/Mujer'),
        ('Familiaris', 'Canis', 'Canis Familiaris', 'Perro'),
        ('Silvestris', 'Felis', 'Felis Silvestris', 'Gato'),
        ('Maíz', 'Zea', 'Zea Maíz', 'Maíz'),
        ('Novaeangliae', 'Megaptera', 'Megaptera Novaeangliae', 'Ballena
Jorobada');

--Si se desea eliminar los datos ingresados, usar este código una vez:
--delete from Especie;
--delete from Genero;
--delete from Familia;
```

```
--delete from Orden;

--delete from Clase;

--delete from Phylum;

--delete from Reino;
```

Función para obtener el reino

El siguiente paso es crear una función llamada "fnc_get_reino", la cual reciba como parámetro de entrada el nombre común de un ser vivo, y devuelva como resultado el reino al cual pertenece.

Para ello, se utiliza una función. Según Richardson (2019), se tienen tres tipos de funciones: escalares, de valor de tabla y de valor de tabla múltiple. La función por realizar pertenece al segundo grupo, donde se obtiene una tabla. Para obtener el reino, se utiliza un select con los inner joins necesarios, donde se conecta el parámetro de entrada por definir, llamado @nombre en la función, con el reino.

Para ejecutar esta función, se usa otro select, con el cual se llama la función y se coloca el parámetro. Se ha utilizado el parámetro "gato", con el cual se obtendrá el reino animal. El script correspondiente es el siguiente:

Función fnc_get_reino

```
--Para crear la función:

GO

create or alter function fnc_get_reino (@Nombre VARCHAR(30))

returns table

as

return

        select Reino.IDReino as Reino_Correspondiente from Especie

                inner join Genero on Especie.Genero=Genero.IDGenero

                inner join Familia on Genero.Familia=Familia.IDFamilia

                inner join Orden on Familia.Orden=Orden.IDOrden
```

```
            inner join Clase on Orden.Clase=Clase.IDClase
            inner join Phylum on Clase.Phylum=Phylum.IDPhylum
            inner join Reino on Phylum.Reino=Reino.IDReino
        where Nombre_Comun=@Nombre;
GO


--Para utilizar la función:
SELECT * FROM fnc_get_reino('Gato');
```

Esta sentencia también se puede utilizar para mostrar el reino de los registros nuevos que serán agregados por medio del procedimiento almacenado para insertar registros, que se realizará a continuación.

Procedimiento para insertar registros

El último paso es generar un procedimiento llamado "sp_inserta_registro", el cual reciba los siguientes parámetros: nombre común, nombre científico, especie, género, familia, orden, clase, phylum y reino, e inmediatamente crear un nuevo registro en las tablas de la base de datos, con los datos recibidos. El procedimiento debe ser capaz de determinar que, si ya hay información existente en la base de datos, solo debe utilizarla, de lo contrario deberá crear lo que sea nuevo.

Para ello, se utilizará un procedimiento almacenado. Según Calbimonte (2019), los administradores de bases de datos suelen preferir los procedimientos a las funciones, debido a que las segundas son menos flexibles. Sin embargo, el código de ambos es muy similar, con parámetros, el código por ejecutar y la línea que llama al procedimiento.

Como parámetros, se colocará toda la taxonomía que se debe añadir, para agregar especies desde un solo procedimiento en lugar de añadir datos a las tablas individualmente. El siguiente paso es añadir los inserts de cada tabla, relacionando las llaves primarias y foráneas con los parámetros por obtener.

Sin embargo, el procedimiento debe poder determinar que la información por añadir sea distinta a la información presente en las tablas, para evitar errores en las llaves primarias. Esto se consigue por medio de sentencias if, con las cuales el sistema podrá detectar datos repetidos y solo almacenará datos nuevos, evitando fallos. Así, si se añade un ser vivo del reino animal, el sistema

no intentará crear un nuevo reino, sino que utilizará el insertado anteriormente por medio de las llaves foráneas. El script es el siguiente:

Procedimiento so_inserta_registro

```
GO
create or alter procedure sp_inserta_registro
	@Nombre_Comun varchar(30),
	@Nombre_Cientifico varchar(30),
	@Especie varchar(30),
	@Genero varchar(30),
	@Familia varchar(30),
	@Orden varchar(30),
	@Clase varchar(30),
	@Phylum varchar(30),
	@Reino varchar(30)
as
	if not exists(select * from Reino where IDReino=@Reino)
		insert into Reino (IDReino) values (@Reino);
	if not exists(select * from Phylum where IDPhylum=@Phylum)
		insert into Phylum (IDPhylum, Reino) values (@Phylum, @Reino);
	if not exists(select * from Clase where IDClase=@Clase)
		insert into Clase (IDClase, Phylum) values (@Clase, @Phylum);
	if not exists(select * from Orden where IDOrden=@Orden)
		insert into Orden (IDOrden, Clase) values (@Orden, @Clase);
	if not exists(select * from Familia where IDFamilia=@Familia)
		insert into Familia (IDFamilia, Orden) values (@Familia, @Orden);
```

```
        if not exists(select * from Genero where IDGenero=@Genero)

                insert into Genero (IDGenero, Familia) values (@Genero, @Familia);

        if not exists(select * from Especie where IDEspecie=@Especie)

                insert into Especie (IDEspecie, Genero, Nombre_Cientifico, Nombre_Comun) values (@Especie, @Genero, @Nombre_Cientifico, @Nombre_Comun);

GO


--Para utilizar el procedimiento:

exec sp_inserta_registro 'Ardilla gris',
                                        'Sciurus Aureogaster',
                                        'Aureogaster',
                                        'Sciurus',
                                        'Sciuridae',
                                        'Rodentia',
                                        'Mammalia',
                                        'Chordata',
                                        'Animal';


exec sp_inserta_registro 'Limonero',
                                        'Citrus x Limon',
                                        'Limon',
                                        'Citrus',
                                        'Rutaceae',
                                        'Sapindales',
                                        'Magnoliopsida',
```

```
                                        'Magnoliophyta',
                                        'Vegetal';

exec sp_inserta_registro 'Ameba',
                                        'Amoeba Agilis',
                                        'Agilis',
                                        'Amoeba',
                                        'Amoebidae',
                                        'Euamoebida',
                                        'Tubulinea',
                                        'Amoebozoa',
                                        'Protista';

--Para observar todos los registros ingresados:
select Reino.IDReino, Phylum.IDPhylum, Clase.IDClase, Orden.IDOrden,
        Familia.IDFamilia, Genero.IDGenero, Especie.IDEspecie,
        Especie.Nombre_Cientifico, Especie.Nombre_Comun from Especie
                inner join Genero on Especie.Genero=Genero.IDGenero
                inner join Familia on Genero.Familia=Familia.IDFamilia
                inner join Orden on Familia.Orden=Orden.IDOrden
                inner join Clase on Orden.Clase=Clase.IDClase
                inner join Phylum on Clase.Phylum=Phylum.IDPhylum
                inner join Reino on Phylum.Reino=Reino.IDReino
                order by Nombre_Comun;
```

Conclusión

Durante la realización de este trabajo, se ha investigado sobre las funciones y procedimientos almacenados en SQL, desde sus definiciones, hasta sus distintos tipos y en qué se diferencian. De esa manera, se pueden implementar ambos dependiendo de la situación en una base de datos.

Para demostrar el manejo de las funciones, se ha implementado una en la base de datos de la taxonomía de distintos animales, llamado "fnc_get_reino". Con esta, se ingresa un nombre común de un animal como parámetro y la función obtiene el reino al que pertenece. De esta forma, si se ingresa el maíz, la función retornará el reino vegetal, mientras que, si se ingresa la ameba, se obtendrá el reino protista.

Por último, se ha implementado un procedimiento almacenado que inserta registros nuevos, llamado "sp_inserta_registro". Debido a que se tienen múltiples tablas, insertar un registro taxonómico nuevo requiere insertar manualmente la categoría en cada tabla. Este procedimiento se encarga de realizar este proceso de forma automática, con solo ingresar las categorías del ser vivo. Además, si la categoría ya existe, este proceso utiliza el ya existente en lugar de duplicarlo. De esta forma, se facilita enormemente el ingreso de nuevos registros a la base de datos.

Así, se pueden implementar funciones y procedimientos almacenados en las bases de datos en SQL. Con cualquier proyecto, se pueden simplificar las tareas por realizar de esta manera, en lugar de depender solamente de las sentencias insert, select, update y delete.

Bibliografía

Álvarez, G. (2019). Procedimientos almacenados en SQL Server. *Kyocode*. https://www.kyocode.com/2019/07/procedimientos-almacenados-en-sql-server/

Calbimonte, D. (2019). Funciones frente a los procedimientos almacenados en SQL Server. SQLShack. https://www.sqlshack.com/es/funciones-frente-a-los-procedimientos-almacenados-en-sql-server/

Richardson, B. (2019). Cómo utilizar las funciones integradas de SQL Server y crear funciones escalares definidas por el usuario. *SQLShack*. https://www.sqlshack.com/es/como-utilizar-las-funciones-integradas-de-sql-server-y-crear-funciones-escalares-definidas-por-el-usuario/