**Evaluación de Aprendizaje N°1**

Base de datos 2

Alumno: Marcos Cabral

DNI: 42684487

Profesores: Osores Hernán. Pardieux Eliana.

Fecha de Entrega: 14/10/2020.

Las tablas dadas anteriormente se encuentran en primera forma normal, ya que ambas están divididas en dos tablas, hay repetición de casi todos los campos, por ejemplo toda la información del cliente en la tabla Préstamo, o la editorial del libro.

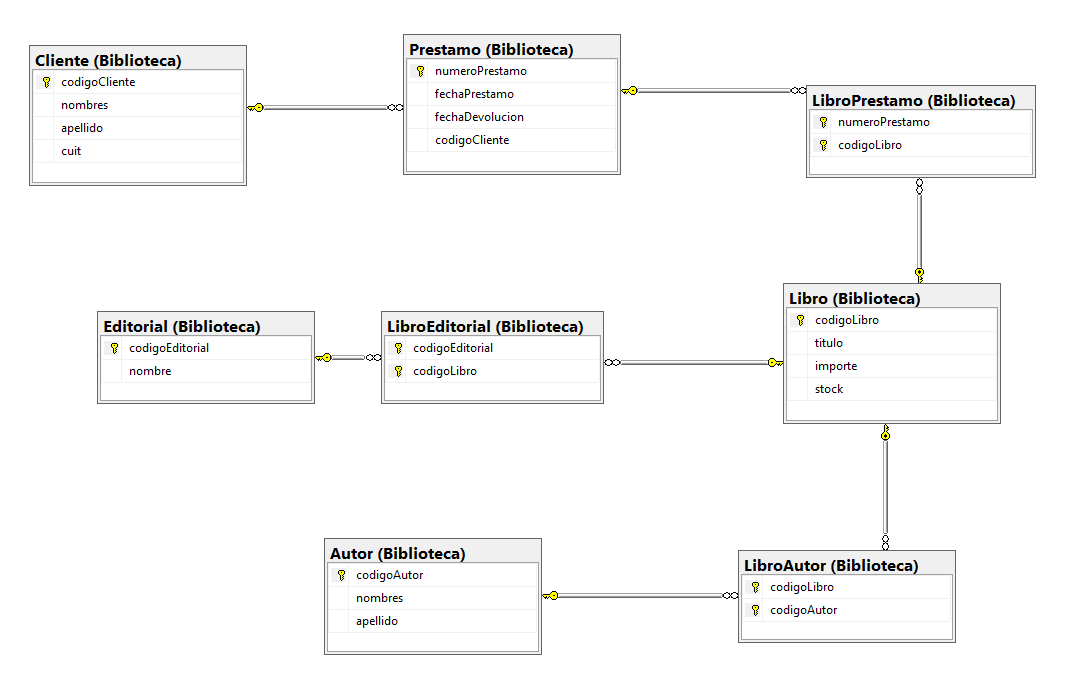
Las ventajas de tener una base de datos normalizada son:

La menor repetición de datos en una misma tabla, por ende logra mayor facilidad a la hora de entender y leer la base de datos.

Previene errores con los datos, tales como escribir múltiples veces campos que podrían ser propios de una tabla (Países) como ‘Argentina’, dando la libre opción que se ingrese ‘Arg’.

Y es más escalable, permite agregar columnas sin romper relaciones o el esquema actual.

Luego de normalizar la tabla, mi base de datos quedó de la siguiente forma:



En el mismo diagrama, asumo como nuevas tablas la de libroPrestamo, ya que un préstamo puede tener muchos libros y un libro puede estar en muchos préstamos, de esta manera con una relación nn y no una 1-n, no estaríamos repitiendo los grupos del préstamo cada vez que un libro sea pedido, es decir no estaríamos cargando información tal como las fechas y el cliente, donde me pareció mas organizado tener una tabla que se encargue de mantener al préstamo con el libro. A nivel de querys, es un poco más pesado por el hecho de tener una nn, pero en mi opinión permite ser más funcional que tener todo dentro del préstamo. LibroEditorial, asumiendo que un libro puede estar en muchas editoriales y una editorial posee muchos libros. LibroAutor, asumiendo que un libro puede tener uno o más autores, y un autor puede tener muchos libros.

1.

create database Biblioteca on primary

( NAME = 'biblio\_data1',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\biblio\_data1.mdf' ,

SIZE = 20MB ,

MAXSIZE = 20MB ,

FILEGROWTH = 20MB

),

( NAME = 'biblio\_data2',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\biblio\_data2.mdf' ,

SIZE = 20MB ,

MAXSIZE = 20MB ,

FILEGROWTH = 20MB

)

LOG ON

( NAME = 'biblio\_log1',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\biblio\_log1.ldf' ,

SIZE = 10MB ,

MAXSIZE = 10MB ,

FILEGROWTH = 10MB

),

( NAME = 'biblio\_log2',

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\biblio\_log2.ldf' ,

SIZE = 10MB ,

MAXSIZE = 10MB ,

FILEGROWTH = 10MB

)

2. create schema Biblioteca;

CREATE TABLE Biblioteca.Cliente(

codigoCliente int NOT NULL,

nombres varchar(30) not null,

apellido varchar(30) not null,

primary key (codigoCliente)

);

CREATE TABLE Biblioteca.Autor(

codigoAutor int NOT NULL,

nombres varchar(30) not null,

apellido varchar(30) not null,

primary key (codigoAutor)

);

CREATE TABLE Biblioteca.Editorial(

codigoEditorial int NOT NULL,

nombre varchar(30) not null,

primary key (codigoEditorial)

);

CREATE TABLE Biblioteca.Libro(

codigoLibro int NOT NULL,

titulo varchar(30) not null,

primary key (codigoLibro)

);

CREATE TABLE Biblioteca.LibroEditorial(

codigoEditorial int NOT NULL,

codigoLibro int NOT NULL,

primary key (codigoEditorial,codigoLibro),

foreign key (codigoEditorial) references Biblioteca.Editorial(codigoEditorial),

foreign key (codigoLibro) references Biblioteca.Libro(codigoLibro)

);

CREATE TABLE Biblioteca.LibroAutor(

codigoLibro int NOT NULL,

codigoAutor int NOT NULL,

primary key(codigoLibro,codigoAutor),

foreign key (codigoLibro) references Biblioteca.Libro(codigoLibro),

foreign key (codigoAutor) references Biblioteca.Autor(codigoAutor)

);

CREATE TABLE Biblioteca.Prestamo(

numeroPrestamo int NOT NULL,

fechaPrestamo date,

fechaDevolucion date,

codigoCliente int NOT NULL,

primary key(numeroPrestamo),

foreign key (codigoCliente) references Biblioteca.Cliente(codigoCliente),

);

CREATE TABLE Biblioteca.LibroPrestamo(

numeroPrestamo int NOT NULL,

codigoLibro int NOT NULL,

primary key(numeroPrestamo,codigoLibro),

foreign key (numeroPrestamo) references Biblioteca.Prestamo(numeroPrestamo),

foreign key (codigoLibro) references Biblioteca.Libro(codigoLibro)

);

**3.**

insert into Biblioteca.Autor values(1,'JJ','Benitez'),(2,'Julio','Cortazar'),(3,'Julio','Cortazar');

insert into Biblioteca.Libro values(1,'Caballo de troya 1');

insert into Biblioteca.Libro values(2,'Caballo de troya 2');

insert into Biblioteca.Libro values(3,'Libro de cortazar 1');

insert into Biblioteca.Libro values(4,'Cortazar ft Borges');

insert into Biblioteca.LibroAutor values(1,1);

insert into Biblioteca.LibroAutor values(2,1);

insert into Biblioteca.LibroAutor values(3,2);

insert into Biblioteca.LibroAutor values(4,2);

insert into Biblioteca.LibroAutor values(4,3);

insert into Biblioteca.Editorial values(1,'Editorial 1'),(2,'Editorial 2'),(3,'Editorial 3');

insert into Biblioteca.LibroEditorial values(1,1),(3,1),(1,2),(2,2),(3,2),(1,3),(2,4),(3,4);

insert into Biblioteca.Cliente values(1,'Marcos','Cabral'),(2,'German','Romero'),(3,'Sofia','Aguero');

insert into Biblioteca.Prestamo values(1,'2020-10-10','2020-10-17',1),(2,'2020-09-19','2020-09-28',2),

(3,'2020-10-01','2020-11-01',3),(4,'2020-10-12','2020-10-30',1);

insert into Biblioteca.LibroPrestamo values(1,1),(1,2),(2,3),(3,4),(4,1),(4,2),(4,3),(4,4);

4.

ALTER TABLE Biblioteca.Libro

ADD importe float not null

CONSTRAINT importe\_cero default (0);

ALTER TABLE Biblioteca.Libro

ADD stock int not null

CONSTRAINT stock\_libro default (0);

ALTER TABLE Biblioteca.Cliente

ADD cuit varchar(25) not null

CONSTRAINT cuit\_void default ('');

ALTER TABLE Biblioteca.Cliente

ADD CONSTRAINT cuit\_unique UNIQUE (cuit);

7. 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.

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.LibroEditorial as le

on le.codigoLibro=l.codigoLibro

inner join Biblioteca.Editorial as e

on e.codigoEditorial=le.codigoEditorial

inner join Biblioteca.LibroAutor as la

on la.codigoLibro=l.codigoLibro

inner join Biblioteca.Autor as a

on a.codigoAutor=la.codigoAutor;

8. Listar todos los libros que alguna vez fueron prestados.

select l.titulo,l.importe,l.stock

from Biblioteca.Libro as l

where l.codigoLibro in (select codigoLibro from Biblioteca.LibroPrestamo);

9. - Listar los libros con sus respectivos autores.

select titulo,importe,stock,nombres,apellido

from Biblioteca.Libro as l

inner join Biblioteca.LibroAutor as la

on l.codigoLibro=la.codigoLibro

inner join Biblioteca.Autor as a

on la.codigoAutor=a.codigoAutor;

10. Listar los libros que fueron escritos por dos o más autores

select \* from Biblioteca.Libro

where codigoLibro =(

select codigoLibro from Biblioteca.LibroAutor as la

having count(la.codigoAutor)>=2

);

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

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

);

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

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;

13. La principal diferencia entre ambos índices es que en un índice agrupado los registros están ordenados y almacenados de forma secuencial en función de su clave, además de definirse uno por tabla y agrupar físicamente los registros. Por otra parte, los índices no agrupados utilizan referencias, al momento de consultarlas no devuelven registros en sí, sino más bien la ubicación real del registro a consultar.

Otras diferencias a destacar es que los índices agrupados modifican el orden físico de los registros, ya que ordena secuencialmente, y es por esta razón que deben ser creados primero.

Y por último, entre otras diferencias, es que un índice agrupado suele ser utilizado para campos en los que se realizan búsquedas con frecuencia, mientras que para un índice no agrupado es más bien para cuando se realizan distintos tipos de búsquedas frecuentemente.

Si no definimos o no se especifica un tipo de índice, el modo predeterminado que tomará el índice será no agrupado.

14. Al momento de crear una clave primaria lo que sucede es que SQL Server crea o genera de manera automática un índice en una tabla.

Los datos residen en una página de datos, ya que al momento de la creación de un índice se genera una página y la misma es asignada al índice. En los índices no agrupados, por ejemplo, se encuentran punteros que apuntan a la página de datos donde reside el registro.

Cabe mencionar que los índices son llamados heap, es decir son una estructura de datos que residen en memoria, y son del tipo árbol, por lo tanto la manera en la que acceden a los datos y las búsquedas que realizan son eficientes y rápidas.