

Documentación proyecto final taller de bd

Docente: Rubén Miguel Riojas Rodríguez



14 de junio de 2021

tecnm monclova coahuila

6ATM ING. Informática

Alumno: Andrés Iván Rodríguez Hernández

Contenido

[Diagrama entidad relacion 2](#_Toc74660785)

[Modelo relacional 2](#_Toc74660786)

[Diccionario de datos 3](#_Toc74660787)

[Creación de la BD 4](#_Toc74660788)

[Creación de las tablas 5](#_Toc74660789)

[Creación de las tablas históricas 9](#_Toc74660790)

[Creación de los índices 10](#_Toc74660791)

[Relaciones entre las entidades 11](#_Toc74660792)

[Creación de los triggers 13](#_Toc74660793)

[Procedimientos almacenados 17](#_Toc74660794)

[Funciones de TSQL 20](#_Toc74660795)

[vistas 22](#_Toc74660796)

[Población de las tablas 24](#_Toc74660797)

[Select de todas las tablas 26](#_Toc74660798)

# Redacción

Se requiere de una base de datos que emplee tablas para representar los datos almacenados en el sistema de un banco, los, las sucursales del banco tienen cajeros automáticos, 1 o más empleados, y cada uno de estos un determinado puesto, al cual pueden pertenecer uno o más empleados, la sucursal se encuentra en una ciudad y esta puede pertenecer a un estado, las sucursales producen productos, los cuales solo pueden ser creados en una sucursal, cada producto es adquirido por uno o más clientes y cada uno de estos tener en su posesión uno o más productos, cada uno de estos pueden tener una o más comisiones o viceversa y cada cliente vive también en una ciudad, además de que cada cliente puede uno o más préstamos, existen servicios como luz, agua, gas, etc. Que pueden ser domiciliados con un producto, cada uno de estos últimos puede tener una o más domiciliaciones, mientras que cada una de estas solo pueden ligarse a un producto en específico, y de igual forma cada domiciliación solo pueden tener un servicio, mientras que estos tienen multitud de domiciliaciones asignadas. El tipo de producto corresponde a uno o más productos mientras que al revés, como es de esperarse, no, ya que el producto solo puede tener un solo tipo de producto. Finalmente existen las autorizaciones, en las que cada una puede servir para uno o más movimientos, sin embargo, cada movimiento solo tiene una autorización en específico. Y de igual forma cada producto puede tener varios movimientos, sin embargo, cada movimiento solo puede ser efectuado con un solo producto en específico.

# Diagrama entidad relación

Diagrama

Descripción generada automáticamente

# Modelo relacional

**ciudad**: idCiudad,nombreCiudad,idEstado,estatus

**estado**: idEstado,nombreEstado,estatus

**sucursal**: idSucursal,nombreSucursal,direccionSucursal,idCiudad,estatus

**tipoProducto**: idTipoProducto,nombreTipoProducto,estatus

**producto**: idProducto,codigoProducto,idTipoProducto,idSucursal,estatus

**cliente**: idCliente,nombreCliente,fechaNacimiento,direccionCliente,idCiudad,estatus

**clienteProducto**: idClienteProducto,idCliente,idProducto,estatus

**movimiento**: idMovimiento,montoMovimiento,idProducto,idAutorizacion,estatus

**comision**: idComision,nombreComision,montoComision,estatus

**comisionProducto**: idComisionProducto,idComision, idProducto,estatus

**atm**: idAtm,codigoAtm,fechaUltimoMantenimiento,dinero,idSucursal,estatus

**puesto**: idPuesto,nombrePuesto,estatus

**empleado**: idEmpleado,nombreEmpleado,idSucursal,idPuesto,estatus

**autorizacion**: idAutorizacion,descripcionAutorizacion,estatus

**domiciliacion**: idDomiciliacion,descripcionDomiciliacion,montoDomiciliacion,idServicio,idProducto,estatus

**servicio**: idServicio,descripcionServicio,estatus

**prestamo**: idPrestamo,fechaPrestamo,cantidadPrestamo,idCliente,estatus

**tablas nm ----- tablas 1N**

# Diccionario de datos

Una captura de pantalla de un videojuego

Descripción generada automáticamente con confianza baja

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Imagen que contiene gabinete, muebles, llenado, entero

Descripción generada automáticamente

Imagen que contiene gabinete, llenado, entero, mucho

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Imagen que contiene gabinete

Descripción generada automáticamente

# Creación de la BD

El nombre que recibe es “Banco”, se verifica que no exista anteriormente y se ser así es dropeada, y seguidamente se crean los archivos mdf y ldf.

USE master;

GO

IF DB\_ID (N'Banco') IS NOT NULL

DROP DATABASE Banco

GO

CREATE DATABASE Banco

ON

( NAME =Banco\_dat,

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\Banco.mdf',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5 )

LOG ON

( NAME = Banco\_log,

FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\Banco.ldf',

SIZE = 5MB,

MAXSIZE = 25MB,

FILEGROWTH = 5MB ) ;

GO

USE Banco

GO

# Creación de las tablas

----------------------------------Tablas--------------------------------------

CREATE TABLE ciudad

(

   idCiudad int IDENTITY (1,1),

   CONSTRAINT PK\_Ciudad PRIMARY KEY (idCiudad),

   nombreCiudad varchar(20),

   idEstado int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE estado

(

   idEstado int IDENTITY (1,1),

   CONSTRAINT PK\_Estado PRIMARY KEY (idEstado),

   nombreEstado varchar(20),

   estatus bit NOT NULL,

)

GO

Se emplea el idSucursal como identificador único auto incrementable, un nombre que lo diferencia de los demás datos, una dirección, una llave foránea idCiudad para tener una relación con la tabla ciudad y un estatus para tener la vigencia del dato.

CREATE TABLE sucursal

(

   idSucursal int IDENTITY (1,1),

   CONSTRAINT PK\_Sucursal PRIMARY KEY (idSucursal),

   nombreSucursal varchar(20),

   direccionSucursal varchar(40),

   idCiudad int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE tipoProducto

(

   idTipoProducto int IDENTITY (1,1),

   CONSTRAINT PK\_TipoProducto PRIMARY KEY (idTipoProducto),

   nombreTipoProducto varchar(20),

   estatus bit NOT NULL,

)

GO

CREATE TABLE producto

(

   idProducto int IDENTITY (1,1),

   CONSTRAINT PK\_Producto PRIMARY KEY (idProducto),

   codigoProducto varchar(20),

   idTipoProducto int NOT NULL,

   idSucursal int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE cliente

(

   idCliente int IDENTITY (1,1),

   CONSTRAINT PK\_Cliente PRIMARY KEY (idCliente),

   nombreCliente varchar(20),

   fechaNacimiento datetime,

   direccionCliente varchar(30),

   idCiudad int NOT NULL,

   estatus bit NOT NULL,

)

GO

Se tiene un idClienteProducto como identificador único auto incrementable. Se crea un idCliente que es con el que se tiene una relación, lo mismo con la tabla producto y el campo estatus.

CREATE TABLE clienteProducto

(

   idClienteProducto int IDENTITY (1,1),

   CONSTRAINT PK\_ClienteProducto PRIMARY KEY (idClienteProducto),

   idCliente int NOT NULL,

   idProducto int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE movimiento

(

   idMovimiento int IDENTITY (1,1),

   CONSTRAINT PK\_Movimiento PRIMARY KEY (idMovimiento),

   montoMovimiento decimal,

   idProducto int NOT NULL,

   idAutorizacion int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE comision

(

   idComision int IDENTITY (1,1),

   CONSTRAINT PK\_Comision PRIMARY KEY (idComision),

   nombreComision varchar(20),

   montoComision decimal NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE comisionProducto

(

   idComisionProducto int IDENTITY (1,1),

   CONSTRAINT PK\_ComisionProducto PRIMARY KEY (idComisionProducto),

   idComision int NOT NULL,

   idProducto int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE atm

(

   idAtm int IDENTITY (1,1),

   CONSTRAINT PK\_Atm PRIMARY KEY (idAtm),

   codigoAtm varchar(20),

   fechaUltimoMantenimiento datetime,

   dinero decimal,

   idSucursal int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE puesto

(

   idPuesto int IDENTITY (1,1),

   CONSTRAINT PK\_Puesto PRIMARY KEY (idPuesto),

   nombrePuesto varchar(20),

   estatus bit NOT NULL,

)

GO

CREATE TABLE empleado

(

   idEmpleado int IDENTITY (1,1),

   CONSTRAINT PK\_Empleado PRIMARY KEY (idEmpleado),

   nombreEmpleado varchar(20),

   idSucursal int NOT NULL,

   idPuesto int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE autorizacion

(

   idAutorizacion int IDENTITY (1,1),

   CONSTRAINT PK\_Autorizacion PRIMARY KEY (idAutorizacion),

   descripcionAutorizacion varchar(20),

   estatus bit NOT NULL,

)

GO

CREATE TABLE domiciliacion

(

   idDomiciliacion int IDENTITY (1,1),

   CONSTRAINT PK\_Domiciliacion PRIMARY KEY (idDomiciliacion),

   descripcionDomiciliacion varchar(20),

   montoDomiciliacion decimal,

   idServicio int NOT NULL,

   idProducto int NOT NULL,

   estatus bit NOT NULL,

)

GO

CREATE TABLE servicio

(

   idServicio int IDENTITY (1,1),

   CONSTRAINT PK\_Servicio PRIMARY KEY (idServicio),

   descripcionServicio varchar(20),

   estatus bit NOT NULL,

)

GO

CREATE TABLE prestamo

(

   idPrestamo int IDENTITY (1,1),

   CONSTRAINT PK\_Prestamo PRIMARY KEY (idPrestamo),

   fechaPrestamo datetime,

   cantidadPrestamo decimal,

   idCliente int NOT NULL,

   estatus bit NOT NULL,

)

GO

## Creación de las tablas históricas

CREATE TABLE cliente\_historico

(

   idCliente\_historico int,

   CONSTRAINT PK\_Cliente\_historico PRIMARY KEY (idCliente\_historico),

   nombreCliente varchar(20),

   fechaNacimiento datetime,

   direccionCliente varchar(30),

   idCiudad int NOT NULL,

   fechaUltimaMod datetime,

   estatus bit NOT NULL,

)

GO

CREATE TABLE producto\_historial

(

   idProducto\_historial int,

   CONSTRAINT PK\_Producto\_historial PRIMARY KEY (idProducto\_historial),

   codigoProducto varchar(20),

   anteriorCodigo varchar(20) default 0000,

   idTipoProducto int NOT NULL,

   idSucursal int NOT NULL,

   situacion varchar(20),

   fechaUltimaMod varchar(20) default '12/31/99',

   estatus bit NOT NULL,

)

GO

# Creación de los índices

El nombre del índice consta en el de la tabla, precedido por “IX\_” y la que corresponde con el índice mencionado y el id de la tabla que corresponde entre paréntesis + ; y GO. Lo mismo con las tablas NM

---------------------------------INDEX-------------------------

CREATE INDEX IX\_Ciudad ON Ciudad(idCiudad);

GO

CREATE INDEX IX\_Estado ON Estado(idEstado);

GO

CREATE INDEX  IX\_Sucursal ON Sucursal (idSucursal);

GO

CREATE INDEX  IX\_Producto ON Producto (idProducto);

GO

CREATE INDEX  IX\_TipoProducto ON TipoProducto (idTipoProducto);

GO

CREATE INDEX IX\_Cliente ON Cliente(idCliente);

GO

CREATE INDEX IX\_ClienteProducto ON ClienteProducto(idClienteProducto);

GO

CREATE INDEX IX\_Movimiento ON Movimiento(idMovimiento);

GO

CREATE INDEX  IX\_Comision ON Comision (idComision);

GO

CREATE INDEX  IX\_ComisionProducto ON ComisionProducto (idComisionProducto);

GO

CREATE INDEX  IX\_Atm ON Atm (idAtm);

GO

CREATE INDEX  IX\_Puesto ON Puesto (idPuesto);

GO

CREATE INDEX  IX\_Empleado ON Empleado (idEmpleado);

GO

CREATE INDEX IX\_Autorizacion ON Autorizacion(idAutorizacion);

GO

CREATE INDEX  IX\_Domiciliacion ON Domiciliacion (idDomiciliacion);

GO

CREATE INDEX  IX\_Servicio ON Servicio (idServicio);

GO

CREATE INDEX  IX\_Prestamo ON Prestamo (idPrestamo);

GO

# Relaciones entre las entidades

---------------------------------RELACIONES--------------------

ALTER TABLE Ciudad

ADD CONSTRAINT FK\_CiudadEstado

FOREIGN KEY (idEstado) REFERENCES Estado(idEstado);

GO

ALTER TABLE Sucursal

ADD CONSTRAINT FK\_SucursalCiudad

FOREIGN KEY (idCiudad) REFERENCES Ciudad(idCiudad);

GO

ALTER TABLE Producto

ADD CONSTRAINT FK\_ProductoTipoProducto

FOREIGN KEY (idTipoProducto) REFERENCES TipoProducto(idTipoProducto);

GO

ALTER TABLE Producto

ADD CONSTRAINT FK\_ProductoSucursal

FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal);

GO

En las relaciones 1N la tabla que tiene la “N” es la que en su creación contiene la llave foránea correspondiente con la tabla con la que se relaciona. Y en la creación del constraint es sobre la que se crea este, nombrándolo como “FK\_’tablaN’+’tabla1’” y creando la foreign key con los datos de la ‘tabla1’

ALTER TABLE Cliente

ADD CONSTRAINT FK\_ClienteCiudad

FOREIGN KEY (idCiudad) REFERENCES Ciudad(idCiudad);

GO

En las relaciones NM se crean dos constraints para la misma tabla NM, una con fk de la tabla “N” y lo mismo con la “M”, nombrando el constraint como “FK\_’tablaNM’+’tablaN’” y el otro constraint con “FK\_’tablaNM’+’tablaM’”

ALTER TABLE ClienteProducto

ADD CONSTRAINT FK\_ClienteProductoCliente

FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente);

GO

ALTER TABLE ClienteProducto

ADD CONSTRAINT FK\_ClienteProductoProducto

FOREIGN KEY (idProducto) REFERENCES Producto(idProducto);

GO

ALTER TABLE Movimiento

ADD CONSTRAINT FK\_MovimientoProducto

FOREIGN KEY (idProducto) REFERENCES Producto(idProducto);

GO

ALTER TABLE Movimiento

ADD CONSTRAINT FK\_MovimientoAutorizacion

FOREIGN KEY (idAutorizacion) REFERENCES Autorizacion(idAutorizacion);

GO

ALTER TABLE ComisionProducto

ADD CONSTRAINT FK\_ComisionProductoComision

FOREIGN KEY (idComision) REFERENCES Comision(idComision);

GO

ALTER TABLE ComisionProducto

ADD CONSTRAINT FK\_ComisionProductoProducto

FOREIGN KEY (idProducto) REFERENCES Producto(idProducto);

GO

ALTER TABLE Atm

ADD CONSTRAINT FK\_AtmSucursal

FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal);

GO

ALTER TABLE Empleado

ADD CONSTRAINT FK\_EmpleadoSucursal

FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal);

GO

ALTER TABLE Empleado

ADD CONSTRAINT FK\_EmpleadoPuesto

FOREIGN KEY (idPuesto) REFERENCES Puesto(idPuesto);

GO

ALTER TABLE Domiciliacion

ADD CONSTRAINT FK\_DomiciliacionServicio

FOREIGN KEY (idServicio) REFERENCES Servicio(idServicio);

GO

ALTER TABLE Domiciliacion

ADD CONSTRAINT FK\_DomiciliacionProducto

FOREIGN KEY (idProducto) REFERENCES Producto(idProducto);

GO

ALTER TABLE Prestamo

ADD CONSTRAINT FK\_PrestamoCliente

FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente);

GO

# Creación de los triggers

Seguida de la creación de cada uno de los triggers, aparecen comentadas las líneas que los invocan

Seguido de insertar un dato en la tabla cliente, inserta esa misma información en la tabla cliente\_historico

 IF OBJECT\_ID ('trigger1', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger1;

END

go

    CREATE or ALTER TRIGGER trigger1

    ON cliente

    AFTER INSERT

   AS

    BEGIN

insert into cliente\_historico(idCliente\_historico,nombreCliente, fechaNacimiento, direccionCliente, idCiudad,fechaUltimaMod,estatus) (select idCliente,nombreCliente, fechaNacimiento, direccionCliente, idCiudad,'12/31/99', 1 from inserted)

--UPDATE cliente\_historico SET fechaUltimaMod = '12/31/99';

--insert into cliente\_historico(fechaUltimaMod) values ('12/31/99')

    END

GO

Después de actualizar un dato en la tabla cliente, actualiza también la fecha de la ultima modificación y la fecha de nacimiento

 IF OBJECT\_ID ('trigger2', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger2;

END

go

    CREATE or ALTER TRIGGER trigger2

    ON cliente

    AFTER UPDATE

   AS

    BEGIN

UPDATE cliente\_historico SET fechaUltimaMod = GetDate() WHERE idCliente\_historico=(SELECT idCliente FROM inserted);

UPDATE cliente\_historico SET fechaNacimiento = (select fechaNacimiento from inserted) WHERE idCliente\_historico=(SELECT idCliente FROM inserted);

--UPDATE cliente\_historico SET fechaUltimaMod = GETDATE() where idCliente\_historico = ;

--insert into cliente\_historico(fechaUltimaMod) values ('12/31/99')

    END

GO

--select \* from cliente; select \* from cliente\_historico;

--UPDATE cliente set fechaNacimiento = '03/13/03' where idCliente = 3

--select \* from cliente; select \* from cliente\_historico;

Después de eliminar un registro de cliente, también se elimina de cliente\_histórico

 IF OBJECT\_ID ('trigger3', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger3;

END

go

    CREATE or ALTER TRIGGER trigger3

    ON cliente

    AFTER DELETE

   AS

    BEGIN

DELETE cliente\_historico  WHERE idCliente\_historico=(SELECT idCliente FROM deleted);

    END

GO

--select \* from cliente;select \* from cliente\_historico;

--delete cliente where idCliente = 6

--select \* from cliente;select \* from cliente\_historico;

Despues de eliminar un registro de producto, se inserta en producto\_historial

 IF OBJECT\_ID ('trigger4', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger4;

END

go

    CREATE or ALTER TRIGGER trigger4

    ON producto

    AFTER delete

   AS

    BEGIN

   --INSERT producto\_eliminado select \* from producto where idProducto=old.idProducto

   --INSERT INTO producto\_eliminado(idProducto\_eliminado, codigoProducto) (SELECT idProducto, codigoProducto FROM deleted);

   insert into producto\_historial (idProducto\_historial,codigoProducto,idTipoProducto,idSucursal,situacion,estatus)(select idProducto,codigoProducto,idTipoProducto,idSucursal,'eliminado',estatus from deleted)

--DELETE producto\_eliminado  WHERE idProducto\_eliminado=(SELECT idProducto FROM deleted);

    END

GO

--select \* from producto; select \* from producto\_eliminado;

--delete producto where idProducto = 6;

--select \* from producto; select \* from producto\_eliminado;

Después de insertar un registro en producto, se registra también en producto historial

 IF OBJECT\_ID ('trigger5', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger5;

END

go

    CREATE or ALTER TRIGGER trigger5

    ON producto

    AFTER insert

   AS

    BEGIN

   insert into producto\_historial (idProducto\_historial,codigoProducto,idTipoProducto,idSucursal,situacion,estatus) (select idProducto,codigoProducto,idTipoProducto,idSucursal,'agregado',estatus from inserted)

    END

GO

--select \* from producto; select \* from producto\_historial;

--INSERT INTO producto (codigoProducto, idTipoProducto, idSucursal, estatus) VALUES (234,5,4,1);

--select \* from producto; select \* from producto\_historial;

Cuando se actualiza un dato en producto, también se actualizan producto historial en fechaUltimaMod para poner la fecha actual y cambian de situación a actualizado en vez de agregado

Este trigger, luego de actualizar un dato en La tabla producto, actualiza también de la tabla nombrada producto\_historial, el campo fechaUltimaMod por la fecha actual y el campo situación, a “actualizado” ambos campos donde el id sea igual al del id del registro actualizado de la tabla para la cual fue creado el trigger.

 IF OBJECT\_ID ('trigger6', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger6;

END

go

    CREATE or ALTER TRIGGER trigger6

    ON producto

    AFTER update

   AS

    BEGIN

   UPDATE producto\_historial SET fechaUltimaMod = GetDate() WHERE idProducto\_historial=(SELECT idProducto FROM inserted);

   UPDATE producto\_historial SET situacion = 'actualizado' WHERE idProducto\_historial=(SELECT idProducto FROM inserted);

    END

GO

--select \* from producto; select \* from producto\_historial;

--UPDATE producto set codigoProducto = 0963 where idProducto = 5

--select \* from producto; select \* from producto\_historial;

Luego de actualizer en producto, se actualiza el código de producto en producto historial y en la columna o campo anterior código cambia de 0 al código que acaba de ser reemplazado por el nuevo

 IF OBJECT\_ID ('trigger7', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger7;

END

go

    CREATE or ALTER TRIGGER trigger7

    ON producto

    after update

   AS

    BEGIN

   UPDATE producto\_historial set anteriorCodigo = (select codigoProducto from deleted) where idProducto\_historial = (select idProducto from inserted)

   UPDATE producto\_historial SET codigoProducto = (select codigoProducto from inserted) where idProducto\_historial = (SELECT idProducto FROM inserted);

    END

GO

--select \* from producto; select \* from producto\_historial;

--UPDATE producto set codigoProducto = 0963 where idProducto = 5

--select \* from producto; select \* from producto\_historial;

Después de inserter en empleado, se invoca un mensaje que contiene los datos del ingreso

 IF OBJECT\_ID ('trigger8', 'TR') IS NOT NULL

BEGIN

   DROP TRIGGER trigger8;

END

go

    CREATE or ALTER TRIGGER trigger8

    ON empleado

    after insert

   AS

    BEGIN

   declare @Mensaje varchar(200)

   declare @Dato numeric

   declare @Nom varchar(20)

   select @Dato = (select idSucursal from inserted)

   select @Nom = (select nombreEmpleado from inserted)

   select @Mensaje = 'Nuevo empleado en el banco en sucursal: ' + @Nom + ' Sucursal No. ' + convert(varchar(10),@Dato)

   raiserror(@Mensaje,0,1)

    END

GO

Después de actualizar la tabla empleado, se muestran los registros de esta

IF OBJECT\_ID ('trigger9', 'TR') IS NOT NULL

BEGIN

DROP TRIGGER trigger9;

END

go

CREATE or ALTER TRIGGER trigger9

ON empleado

after update

AS

BEGIN

select \* from empleado;

END

GO

--select \* from empleado;

--UPDATE empleado set nombreEmpleado = 'Amairami Rangel' where idEmpleado = 3

Después de eliminar un registro en empleado, se muestran los datos almacenados en esta tabla y en producto historial

IF OBJECT\_ID ('trigger10', 'TR') IS NOT NULL

BEGIN

DROP TRIGGER trigger10;

END

go

CREATE or ALTER TRIGGER trigger10

ON empleado

after delete

AS

BEGIN

select \* from empleado; select \* from producto\_historial;

END

GO

--delete empleado where idEmpleado = 6;

# Procedimientos almacenados

Después de cada procedimiento aparecen comentadas las líneas que permiten ver los procedimientos en acción al ser invocados

---------------------------------Procedimientos almacenados--------

Envía un mensaje al usuario

IF OBJECT\_ID ('Saludar', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Saludar;

END

go

CREATE or ALTER PROCEDURE Saludar

AS

PRINT 'Hola, Como estas?';

GO

EXECUTE Saludar;

Nos muestra de la tabla cliente los campos nombre y fecha de nacimiento donde este primero sea igual al nombre pasado como parámetro

IF OBJECT\_ID ('Mostrar', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Mostrar;

END

go

CREATE or ALTER PROCEDURE Mostrar

@Nombre varchar(30)

AS

begin

select nombreCliente,fechaNacimiento from cliente where nombreCliente = @Nombre;

end

go

--EXECUTE Mostrar 'Juan Perez';

--EXECUTE Mostrar 'Maria Maldonado';

--EXECUTE Mostrar 'Abelardo Ramirez';

Nos muestra de cliente los campos nombre, fecha de nacimiento y dirección donde este primero sea igual al nombre y fecha recibidos como parámetro

IF OBJECT\_ID ('Procedimiento3', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento3;

END

go

CREATE PROCEDURE Procedimiento3

@NombreCliente nvarchar(50),

@FechaNacimiento datetime

AS

SET NOCOUNT ON;

SELECT nombreCliente, fechaNacimiento, direccionCliente

FROM cliente

WHERE nombreCliente = NombreCliente AND fechaNacimiento = @FechaNacimiento

AND direccionCliente IS NOT NULL;

GO

--execute Procedimiento3 N'Jose Esquivel','04/20/01';

Inserta en empleado los valores proporcionados en los parámetros

IF OBJECT\_ID ('Procedimiento4', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento4;

END

go

CREATE PROCEDURE Procedimiento4

@nombre VARCHAR(50),

@sucursal int,

@puesto int

AS

INSERT INTO empleado (nombreEmpleado,idSucursal,idPuesto,estatus)

VALUES (@nombre, @sucursal, @puesto,1)

GO

--select \* from empleado;

--exec Procedimiento4 'juana juana',4,3;

--select \* from empleado;

Muestra el registro del empleado donde el idEmpleado coincida con el parámetro

IF OBJECT\_ID ('Procedimiento5', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento5;

END

go

CREATE PROCEDURE Procedimiento5

@id INT

AS

SELECT \* FROM empleado WHERE idEmpleado = @id

GO

--select \* from empleado;

--exec Procedimiento5 6;

--select \* from empleado;

Actualiza los campos de la tabla empleado con los datos dados en los parámetros

Este procedimiento actualiza en la tabla empleado los campos nombreEmpleado, idSucursal, idPuesto por los datos guardados en los parámetros @nombre, @sucursal y @puesto, el campo estatus por 1 donde el idEmpleado sea igual al parámetro @id. Todos los parámetros son dados por el usuario.

IF OBJECT\_ID ('Procedimiento6', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento6;

END

go

CREATE PROCEDURE Procedimiento6

@id int,

@nombre varchar(20),

@sucursal int,

@puesto int

AS

UPDATE empleado SET

nombreEmpleado = @nombre,

[idSucursal] = @sucursal,

[idPuesto] = @puesto,

[estatus] = 1

WHERE idEmpleado= @id

GO

--select \* from empleado;

--exec Procedimiento6 4,'vasco de mata',2,3;

--select \* from empleado;

Elimina de la tabla empleado el dato cuyo id halla sido proporcionado al método

IF OBJECT\_ID ('Procedimiento7', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento7;

END

go

CREATE PROCEDURE Procedimiento7

@id INT

AS

DELETE FROM empleado WHERE idEmpleado = @id

GO

--select \* from empleado;

--exec Procedimiento7 2;

--select \* from empleado;

Inserta en comisión los valores datos por el usuario a los parámetros

IF OBJECT\_ID ('Procedimiento8', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento8;

END

go

CREATE PROCEDURE Procedimiento8

@nombre VARCHAR(50),

@monto int

AS

INSERT INTO comision(nombreComision,montoComision,estatus)

VALUES (@nombre, @monto,1)

GO

--select \* from comision;

--exec Procedimiento8 'mantenimiento',1100;

--select \* from comision;

Muestra el empleado cuyo id halla sido proporcionado

IF OBJECT\_ID ('Procedimiento9', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento9;

END

go

CREATE PROCEDURE Procedimiento9

@id int

AS

SELECT \* FROM empleado WHERE idEmpleado = @id

GO

--exec Procedimiento9 5;

Elimina al empleado solicitado al usuario por el id

IF OBJECT\_ID ('Procedimiento10', 'PR') IS NOT NULL

BEGIN

DROP PROCEDURE Procedimiento10;

END

go

CREATE PROCEDURE Procedimiento10

@id INT

AS

DELETE FROM empleado WHERE idEmpleado = @id

GO

--select \* from empleado;

--exec Procedimiento10 5;

--select \* from comision;

# Funciones de TSQL

-------------------------------Funciones de TSQL

La primera función cuenta el número de registros donde el montoComision sea > a 15, la segunda da la sumatoria de la misma columna, las 2 últimas funciones dan el máximo y mínimo de la columna.

------AGREGATE---------

**COUNT**() es una **función** agregada de **SQL Server** que devuelve el número de elementos encontrados en un conjunto

select count(nombreComision) from comision where montoComision > 15

Devuelve la suma de todos los valores o solo de los valores DISTINCT de la expresión. SUM solo puede utilizarse con columnas numéricas. Se omiten los valores NULL.

select sum(montoMovimiento) from movimiento

------GROUPING---------

----MAX

Las sentencias SQL MIN y MAX nos ayudan a obtener los valores mínimos y máximos de una columna en una tabla especifica de SQL Server

SELECT MAX(montoComision)

FROM comision;

GO

---MIN

SELECT MIN(montoComision)

FROM comision;

GO

Las funciones proporcionan la fecha del sistema, la hora, establecen una variable @datetime2 y obtiene el resultado de la suma de la fecha establecida mas fracciones de fecha

La primera función nos muestra el valor ASCII de A, B, a, b, 1 y 2. La siguiente muestra la posición que ocupa TEST en el texto del 2do parámetro, la última concatena los valores en una cadena

------DATETIME---------

--Obtener la fecha actual del sistema

SELECT CONVERT (DATE, SYSDATETIME())

,CONVERT (DATE, SYSDATETIMEOFFSET())

,CONVERT (DATE, SYSUTCDATETIME())

,CONVERT (DATE, CURRENT\_TIMESTAMP)

,CONVERT (DATE, GETDATE())

,CONVERT (DATE, GETUTCDATE());

--Obtener la hora actual del sistema

SELECT CONVERT (TIME, SYSDATETIME())

,CONVERT (TIME, SYSDATETIMEOFFSET())

,CONVERT (TIME, SYSUTCDATETIME())

,CONVERT (TIME, CURRENT\_TIMESTAMP)

,CONVERT (TIME, GETDATE())

,CONVERT (TIME, GETUTCDATE());

DECLARE @datetime2 datetime2 = '2007-01-01 13:10:10.1111111';

-----obtiene el resultado de la suma de la fecha establecida mas fracciones de fecha

SELECT 'year', DATEADD(year,1,@datetime2)

UNION ALL

SELECT 'quarter',DATEADD(quarter,1,@datetime2)

UNION ALL

SELECT 'month',DATEADD(month,1,@datetime2)

UNION ALL

SELECT 'dayofyear',DATEADD(dayofyear,1,@datetime2)

UNION ALL

SELECT 'day',DATEADD(day,1,@datetime2)

UNION ALL

SELECT 'week',DATEADD(week,1,@datetime2)

UNION ALL

SELECT 'weekday',DATEADD(weekday,1,@datetime2)

UNION ALL

SELECT 'hour',DATEADD(hour,1,@datetime2)

UNION ALL

SELECT 'minute',DATEADD(minute,1,@datetime2)

UNION ALL

SELECT 'second',DATEADD(second,1,@datetime2)

UNION ALL

SELECT 'millisecond',DATEADD(millisecond,1,@datetime2)

UNION ALL

SELECT 'microsecond',DATEADD(microsecond,1,@datetime2)

UNION ALL

SELECT 'nanosecond',DATEADD(nanosecond,1,@datetime2);

------STRING---------

La **función ASCII**() acepta una expresión de caracteres y devuelve el valor del código **ASCII** del carácter más a la izquierda de la expresión de caracteres

SELECT ASCII('A') AS A, ASCII('B') AS B,

ASCII('a') AS a, ASCII('b') AS b,

ASCII(1) AS [1], ASCII(2) AS [2];

GO

Devuelve el valor del código ASCII del carácter más a la izquierda de una expresión de caracteres.

SELECT CHARINDEX ( 'TEST',

'This is a Test'

COLLATE Latin1\_General\_CI\_AS);

GO

Sirve para concatenar valores proporcionados

SELECT CONCAT( nombreCliente, fechaNacimiento, direccionCliente ) AS Result

FROM cliente;

GO

------CONVERT---------

Tanto parse y convert son útiles para convertir un dato de un determinado tipo a uno de otro determinado tipo

Las 2 primeras funciones convierten el valor datetime a datetime2 y de datetime2 a datetime respectivamente.

La penúltima función convierte montoComision a varchar

La última función convierte el idMovimiento a varchar y el montoMovimiento a varchar.

Las 4 funciones muestran al usuario el resultado mediante select y se especifican el nombre de las tablas y columnas de donde vienen los datos originales

SELECT PARSE('Monday, 13 December 2010' AS datetime2 USING 'en-US') AS Result;

SELECT PARSE('Monday, 13 December 2010' AS datetime USING 'en-US') AS Result;

select 'El monto de comision de ' + nombreComision + ' es ' + cast(montoComision as varchar(15)) as textoprecio from comision

select 'El monto del movimiento ' + cast(idMovimiento as varchar(15)) + ' es ' + convert(varchar(15), montoMovimiento) as textoprecio from movimiento

go

# vistas

Después de cada vista están comentados los select que dan la salida correspondiente al invocar las vistas

Muestra los campos monto y id de la tabla movimiento

---------------------------------------------------views

DROP VIEW IF EXISTS view\_movimiento

GO

CREATE VIEW view\_movimiento AS

SELECT montoMovimiento, idProducto

FROM movimiento

WHERE idMovimiento > 0 and idMovimiento <6;

go

--select \* from view\_movimiento;

--go

Muestra de la tabla producto los campos id, código y id de tipo donde el id del producto sea mayor que 0

DROP VIEW IF EXISTS view\_producto

GO

CREATE VIEW view\_producto

AS

select p.idProducto AS [ProductIdNumber] ,

p.codigoProducto AS [ProductName],

p.idTipoProducto [ProductTypeNumber] from [producto] p

WHERE idProducto >0

go

--select \* from view\_producto

--go

Muestra de la tabla producto los campos de id, código, tipo y sucursal

DROP VIEW IF EXISTS view\_producto

GO

CREATE VIEW view\_producto

AS

select p.idProducto AS [ProductIdNumber] ,

p.codigoProducto AS [ProductName],

p.idTipoProducto [ProductMainNumber],

p.idSucursal as [ProductModelName] from producto p

go

--select \* from view\_producto

--go

De la tabla cliente muestra el idCliente y el idCiudad de los registros que tengan una diferencia horaria mayor o igual a 8030 días entre la fecha de nacimiento y el día actual para comprobar la edad de las personas

DROP VIEW IF EXISTS vista4

GO

CREATE VIEW vista4

AS SELECT idCliente, idCiudad

FROM cliente

WHERE (SELECT DATEDIFF\_BIG(day, fechaNacimiento, GETDATE())) >= 8030

go

-- select \* from vista4

--go

De la tabla atm muestra el id y el código donde la diferencia entre la fecha del último mantenimiento y de la fecha actual sea menor o igual a 150 días

DROP VIEW IF EXISTS vista5

GO

CREATE VIEW vista5

AS SELECT idAtm, codigoAtm

FROM atm

WHERE (SELECT DATEDIFF\_BIG(day, fechaUltimoMantenimiento, GETDATE())) <= 150

go

-- select \* from vista5

--go

De la tabla préstamo muestra el idPrestamo y cantidadPrestamo de los registros que tengan una diferencia mayor o igual 45 en cuanto a la diferencia entre la fecha del préstamo y la fecha actual

DROP VIEW IF EXISTS vista6

GO

CREATE VIEW vista6

AS SELECT idPrestamo, cantidadPrestamo

FROM prestamo

WHERE (SELECT DATEDIFF\_BIG(day, fechaPrestamo, GETDATE())) <= 45

go

select \* from vista6

go

De la tabla succursal muestra el id y el nombre donde la longitud en caracteres de la dirección sucursal sea menor o igual a 20

DROP VIEW IF EXISTS vista7

GO

CREATE VIEW vista7

AS SELECT idSucursal, nombreSucursal

FROM sucursal

WHERE (select len(direccionSucursal)) <= 20

go

-- select \* from vista7

--go

De la tabla sucursal selecciona el id y el nombre donde la longitud de la dirección sea mayor o igual a 20

La siguiente vista nos muestra los campos de la tabla sucursal idSucursal y nombreSucursal donde la longitud en caracteres del campo direccionSucursal sea igual o menor que 20. Además cada vista cuenta con go al final para finalizar el batch y en caso de que la view ya exista es dropeada para ser creada nuevamente.

DROP VIEW IF EXISTS vista8

GO

CREATE VIEW vista8

AS SELECT idSucursal, nombreSucursal

FROM sucursal

WHERE (select len(direccionSucursal)) >= 20

go

-- select \* from vista8

--go

Muestra el id y código del producto cuyo código supere el número 1000

DROP VIEW IF EXISTS vista9

GO

CREATE VIEW vista9

AS SELECT idProducto, codigoProducto

FROM producto

WHERE codigoProducto >= 1000

go

-- select \* from vista9

--go

Muestra el id, código y fecha del último mantenimiento del atm cuya mencionada fecha se encuentre entre 12/31/17 y 01/01/20

DROP VIEW IF EXISTS vista10

GO

CREATE VIEW vista10

AS SELECT idAtm, codigoAtm,fechaUltimoMantenimiento

FROM atm

WHERE fechaUltimoMantenimiento between '12/31/17' and '01/01/20'

go

-- select \* from vista10

--go

# Población de las tablas

------------------------------//POBLACION//------------------------------------

INSERT INTO estado (nombreEstado, estatus)

VALUES ('Coahuila',1),

      ('Durango',1),

      ('Zacatecas',1),

      ('Chihuahua',1),

      ('Hidalgo',1);

INSERT INTO ciudad (nombreCiudad, idEstado, estatus)

VALUES ('monclova',1, 1),

      ('Lerdo',2, 1),

      ('Parral',4,1),

      ('Fresnillo',3, 1),

      ('Pachuca',5, 1);

En el caso de esta tabla se ingresan el nombre de la sucursal, su dirección, ambos en formato varchar, el id de la ciudad ya que es una llave foránea, y por consiguiente debe ser de una ciudad ya existente, y el valor del campo estatus para observar la vigencia del dato.

INSERT INTO sucursal (nombreSucursal, direccionSucursal,idCiudad, estatus)

VALUES ('pape','juan alvarez 14 Jardines de guadalupe',1,1),

      ('sarabia','alberto rodriguez 178 Centro', 2,1),

      ('desierto','Aguacate 3 Pemex', 3,1),

      ('oro','manzana 85 Zapata', 4,1),

      ('tula','ahuitzol 15 Tlaloc', 5,1);

INSERT INTO tipoProducto (nombreTipoProducto, estatus)

VALUES ('cuentaCorriente', 1),

       ('cuentaAhorro', 1),

      ('tarjetaDebito', 1),

      ('tarjetaCredito', 1),

      ('cuentaNomina', 1);

INSERT INTO producto (codigoProducto, idTipoProducto, idSucursal, estatus)

VALUES (0546,2,5,1),

      (2985,1,4,1),

      (3542,3,2,1),

      (8654,4,1,1),

      (5974,5,3,1),

      (0123,1,2,1);

INSERT INTO cliente (nombreCliente, fechaNacimiento, direccionCliente, idCiudad, estatus)

VALUES ('Juan Perez','09/13/94','roble 89 Fovisste',3,1),

       ('Maria Maldonado','02/28/95','gonzalez 97 aviacion',5,1),

      ('Jose Esquivel','04/20/01','principal 667 refugio',4,1),

      ('Antonieta Mendoza','11/14/64','francisco mina 24 heroes',1,1),

      ('Eleazar Arevalo','07/22/89','primo de verdad 997 americas',2,1),

      ('Filomena Ruiz','01/02/03','hermano del mar 123 san jose',4,1);

Se realiza el mismo procedimiento, con la diferencia de que se ingresan los valores que corresponderían con los de la tabla “N”, los de la tabla “M”, valores que deben existir en estas dos últimas tablas 1N anteriormente mencionadas, seguido del valor del campo estatus

INSERT INTO clienteProducto (idCliente, idProducto, estatus)

VALUES (2,3,1),

       (1,5,1),

      (3,4,1),

      (5,2,1),

      (4,1,1);

INSERT INTO autorizacion (descripcionAutorizacion, estatus)

VALUES ('cargo',1),

       ('abono',1),

      ('transferencia',1),

      ('denegado',1),

      ('espera',1);

INSERT INTO movimiento (montoMovimiento, idProducto, idAutorizacion, estatus)

VALUES (9000,1,4,1),

       (20000,2,3,1),

      (4000,3,2,1),

      (10000,5,1,1),

      (31000,4,5,1);

INSERT INTO comision (nombreComision, montoComision,estatus)

VALUES ('apertura',0,1),

       ('anualidad',100,1),

      ('renovacion',5,1),

      ('mantenimiento',2000,1),

      ('interes',200,1);

INSERT INTO comisionProducto (idComision, idProducto, estatus)

VALUES (1,1,1),

       (2,4,1),

      (3,3,1),

      (4,2,1),

      (5,5,1);

INSERT INTO atm (codigoAtm, fechaUltimoMantenimiento, dinero, idSucursal, estatus)

VALUES (3425,'12/23/20',100500,5,1),

       (5432,'05/31/19',50600,3,1),

      (7654,'03/02/21',90300,2,1),

      (4567,'06/12/18',110700,4,1),

      (8765,'04/17/21',30600,1,1);

INSERT INTO puesto (nombrePuesto, estatus)

VALUES ('guardia',1),

       ('intendente',1),

      ('cajero',1),

      ('secretario',1),

      ('ejecutivo',1);

INSERT INTO empleado (nombreEmpleado, idSucursal, idPuesto, estatus)

VALUES ('Eduardo Jimenez',2,3,1),

       ('Araceli Mendoza',3,4,1),

      ('Juana de la Garza',4,5,1),

      ('Luisa de las Nieves',5,1,1),

      ('Alejandro Moreno',1,2,1);

INSERT INTO servicio(descripcionServicio, estatus)

VALUES ('gas',1),

       ('agua',1),

      ('luz',1),

      ('cable',1),

      ('internet',1),

      ('telefono',1);

INSERT INTO domiciliacion (descripcionDomiciliacion, montoDomiciliacion, idServicio, idProducto, estatus)

VALUES ('pagado',1500,2,4,1),

       ('pagado',3000,2,4,1),

      ('adeudo',4500,1,5,1),

      ('adeudo',1500,3,3,1),

      ('pagado',2700,4,1,1),

      ('adeudo',1000,6,2,1),

      ('entram',5000,5,2,1),

      ('adeudo',2400,6,2,1),

      ('pagado',3300,2,2,1),

      ('adeudo',1100,4,2,1);

INSERT INTO prestamo (fechaPrestamo, cantidadPrestamo, idCliente, estatus)

VALUES ('01/09/14',4700,1,1),

       ('09/12/15',1500,5,1),

      ('08/11/21',5400,4,1),

      ('04/21/19',2000,2,1),

      ('02/20/20',1000,1,1);

# Select de todas las tablas

Se presentó la forma en que fueron comprobados el correcto funcionamiento de los triggers, procedimientos almacenados y vistas luego de ser creados y de igual forma se presenta como visualizar los campos de cada tabla a continuación.

SELECT \* FROM Ciudad

SELECT \* FROM Estado

SELECT \* FROM Sucursal

SELECT \* FROM tipoProducto

SELECT \* FROM producto

SELECT \* FROM cliente

SELECT \* FROM clienteProducto

SELECT \* FROM movimiento

SELECT \* FROM comision

SELECT \* FROM comisionProducto

SELECT \* FROM atm

SELECT \* FROM puesto

SELECT \* FROM empleado

SELECT \* FROM autorizacion

SELECT \* FROM domiciliacion

SELECT \* FROM servicio

SELECT \* FROM prestamo

SELECT \* FROM cliente\_historico

SELECT \* FROM producto\_historial

GO;

Código usado en las pruebas

--UPDATE Estado SET nombreEstado = 'tlaxcala' where idEstado = 4;

--DELETE FROM Ciudad WHERE idCiudad=5;

----------triggers-----------------

--CREATE or ALTER TRIGGER trigger\_historico

--ON cliente

--AFTER INSERT

--AS

--BEGIN

----lineas de c�gido SQL que se ejecutar�n

--INSERT INTO cliente\_historico(nombreCliente,fechaNacimiento,direccionCliente)

--VALUES(NEW.nombreCliente,NEW.fechaNacimiento,NEW.direccionCliente,CURDATE());

-- Borramos el Trigger si existise

--IF OBJECT\_ID ('StatusChangeDateTrigger', 'TR') IS NOT NULL

--BEGIN

--   DROP TRIGGER StatusChangeDateTrigger;

--END;

--GO -- Necesario

-- IF OBJECT\_ID ('trigger1', 'TR') IS NOT NULL

--BEGIN

--   DROP TRIGGER trigger1;

--END

--go

--    CREATE or ALTER TRIGGER trigger1

--    ON cliente

--    AFTER INSERT

-- AS

--    BEGIN

--insert into cliente\_historico(nombreCliente, fechaNacimiento, direccionCliente, idCiudad,estatus) (select nombreCliente, fechaNacimiento, direccionCliente, idCiudad, 1 from inserted)

--    END

--GO

--INSERT INTO cliente (nombreCliente, fechaNacimiento, direccionCliente, idCiudad, estatus)

--VALUES ('Alejandro Barboza','10/14/95','pino 90 astillero',4,1);

--select \* from cliente;

--select \* from cliente\_historico;

--GO