Clase 2 de Octubre.

UNO. Construcción de base de datos física

Ver video de la clase, se desarrolla un ejemplo construyendo un modelo físico, para el caso de Cuentas de Ahorro, a partir de un modelo conceptual.

DOS. Construcción de SP para Insertar un Cliente.

```
ALTER PROCEDURE [dbo].[InsertaCliente]
        @inNombre VARCHAR(100)
        , @inValorDocumentoIdent VARCHAR(50)
        , @inTipoDocumentoIdentidadId int
        , @inInsertBy VARCHAR(100)
        , @inInsertIn VARCHAR(60)
        , @outClienteId INT OUTPUT
        , @outResultCode INT OUTPUT
AS
BEGIN
SET NOCOUNT ON;
        BEGIN TRY

        -- CODIGO PARA PROBAR

--DECLARE @RC int
--DECLARE @inNombre varchar(100) = 'Juan'
--DECLARE @inValorDocumentoIdent varchar(50) = '106070999'
--DECLARE @inTipoDocumentoIdentidadId int = 1
--DECLARE @inInsertBy varchar(100) = 'Yomismo'
--DECLARE @inInsertIn varchar(60) = ''
--DECLARE @outClienteId int
--DECLARE @outResultCode int

---- TODO: Set parameter values here.

--EXECUTE @RC = [dbo].[InsertaCliente]
--   @inNombre
--  ,@inValorDocumentoIdent
--  ,@inTipoDocumentoIdentidadId
--  ,@inInsertBy
--  ,@inInsertIn
--  ,@outClienteId OUTPUT
--  ,@outResultCode OUTPUT

--SELECT @outClienteId, @outResultCode

                -- asignar valores a variables

                Select @outResultCode = 0

                -- Valida valores de entrada y si error retornar valor en @outResultCode mayor a 50000

                IF NOT EXISTS (SELECT 1 FROM [dbo].[TipoDocumentoIdentidad] WHERE
id=@inTipoDocumentoIdentidadId)
                BEGIN
                        SET @outResultCode=50001 -- @inTipoDocumentoIdentidadId trae valor que no existe
                        RETURN
                END;

                -- insertar validaciones que faltan ...
```

```
                -- si el SP contiene un solo estatuto de actualizacion no es necesario iniciar transaccion
de BD
                INSERT dbo.Cliente (
                        [Nombre]
                        , [ValorDocumentoIdent]
                        , [TipoDocumentoIdentidadId]
                        , [InsertAt]
                        , [InsertBy]
                        , [InsertIn]
                )
                values (
                        @inNombre
                        , @inValorDocumentoIdent
                        , @inTipoDocumentoIdentidadId
                        , getdate()
                        , @inInsertBy
                        , @inInsertIn
                );
                select @outResultCode=1/0;

                set @outClienteId=SCOPE_IDENTITY();

        END TRY
        BEGIN CATCH
                INSERT INTO dbo.DBErrores        VALUES (
                        SUSER_SNAME(),
                        ERROR_NUMBER(),
                        ERROR_STATE(),
                        ERROR_SEVERITY(),
                        ERROR_LINE(),
                        ERROR_PROCEDURE(),
                        ERROR_MESSAGE(),
                        GETDATE()
                );

                Set @OutResultCode=50005;

        END CATCH;

SET NOCOUNT OFF;
END;
```

Clase Miercoles 7 de Octubre

UNO. Realización del Quiz No.1

NO SE HIZO EL QUIZ POR ERRORES EN SCHOOLOGY.COM

DOS. Inspeccionar ejemplo de SP para insertar un movimiento.

Una transacccion de BD es un conjunto de estatutos que inician con Begin Transaction y terminan con Commit Transaction o Rollback transaction, que se ejecutan bajo la filosófica de TODO o NADA siguiendo los criterios ACID (Atómico, Consistente, Isolated, Durable) respecto de la ejecución de transacciones.

La A de atomico, refiere al Todo o Nada, se ejecuta la transacción completa (todo) si no hay errores, y si los hay no se ejecuta nada.

Variables que inician con @@ son variables del sistema, ejemplo: @@TRANCOUNT que indica la cantidad de transacciones de BD activas en conexión actual.

La transacción de BD debe ir al final del código, luego veremos la razón.

Los objetos que se actualizan en una transacción deben referirse siempre en el mismo orden, en orden ascendente de importancia del objeto.

Es buena practica preprocesar cálculos antes de iniciar transacción, aunque existe riesgo que los valores base de los cálculos sean modificados en el momento que son calculados.

SET TRANSACTION ISOLATION LEVEL READ COMMITTED (O READUNCOMMITTED O SERIALIZABLE), refiere a la I, de Isolated.

```
CREATE PROCEDURE dbo.insertamovimiento
        @inCuentaId INT
        , @inTipoMovimientoId INT
        , @inMonto MONEY
        , @inFecha Date
        , @inDescripcion VARCHAR(200)
        , @outMovimientoId INT OUTPUT
        , @OutResultCode INT OUTPUT
AS
BEGIN
        -- codigo para probar el SP

        --DECLARE
        --      @inCuentaId INT = 123
        --      , @inTipoMovimiento INT = 1
        --      , @inMonto MONEY = 200
        --      , @inFecha Date = '2020-10-7'
        --      , @inDescription VARCHAR(200) = 'Deposito Inicial'
        --      , @outMovimientoId INT
        --      , @OutResultCode INT

        --EXEC dbo.CA_insertamovimiento
        --      @inCuentaId
        --      , @inTipoMovimiento
        --      , @inMonto
        --  , @infecha
        --      , @inDescription
```

```
--      , @outMoivimientoId OUTPUT
--      , @OutResultCode OUTPUT

--SELECT @outMoivimientoId, @OutResultCode

SET NOCOUNT ON;
BEGIN TRY
        -- se declaran variables
        DECLARE
                @nuevoSaldo money
        -- se inicializan variables
        SELECT
                @OutResultCode=0
                ;

        -- Validacion de paramentros de entrada

        IF NOT EXISTS(SELECT 1 FROM dbo.CuentaAhorro C WHERE C.iD=@inCuentaId)
        BEGIN
                Set @OutResultCode=50001; -- cuenta no existe
                RETURN
        END;

        IF NOT EXISTS (SELECT 1 FROM dbo.TipoMovimiento M WHERE M.ID=@inTipoMovimientoId)
        BEGIN
                Set @OutResultCode=50002; -- tipo de movimiento no existe
                RETURN
        END;

        SELECT @NuevoSaldo=CA.Saldo+@inMoNTO
        FROM dbo.CuentaAhorro CA
        where CA.Id=@inCuentaId;

        SET TRANSACTION ISOLATION LEVEL READ COMMITTED
        BEGIN TRANSACTION TSaveMov

                INSERT dbo.MOVIMIENTO (CuentaAhorroId, TipoMovimientoId, Fecha, Monto,
Descripcion, NuevoSaldo)
                VALUES (@InCuentaId, @inTipoMovimientoId, @inFecha, @inMonto, @inDescripcion,
@NuevoSaldo)

                Set @outMovimientoId = SCOPE_IDENTITY();

                UPDATE dbo.CuentaAhorro
                SET Saldo=Saldo+@inMonto
                WHERE Id=@inCuentaId

        COMMIT TRANSACTION TSaveMov;  -- asegura el TODO, que las todas actualizaciones "quedan"
en la BD.

END TRY
BEGIN CATCH
        IF @@TRANCOUNT>0 -- chequeo que el errror sucedio dentro de la transaccion
                ROLLBACK TRANSACTION TSaveMov;    -- asegura el Nada, deshace las actualizaciones
previas al error
        INSERT INTO dbo.DBErrores       VALUES (
                SUSER_SNAME(),
                ERROR_NUMBER(),
                ERROR_STATE(),
                ERROR_SEVERITY(),
                ERROR_LINE(),
                ERROR_PROCEDURE(),
                ERROR_MESSAGE(),
                GETDATE()
        );

        Set @OutResultCode=50005;

END CATCH
```

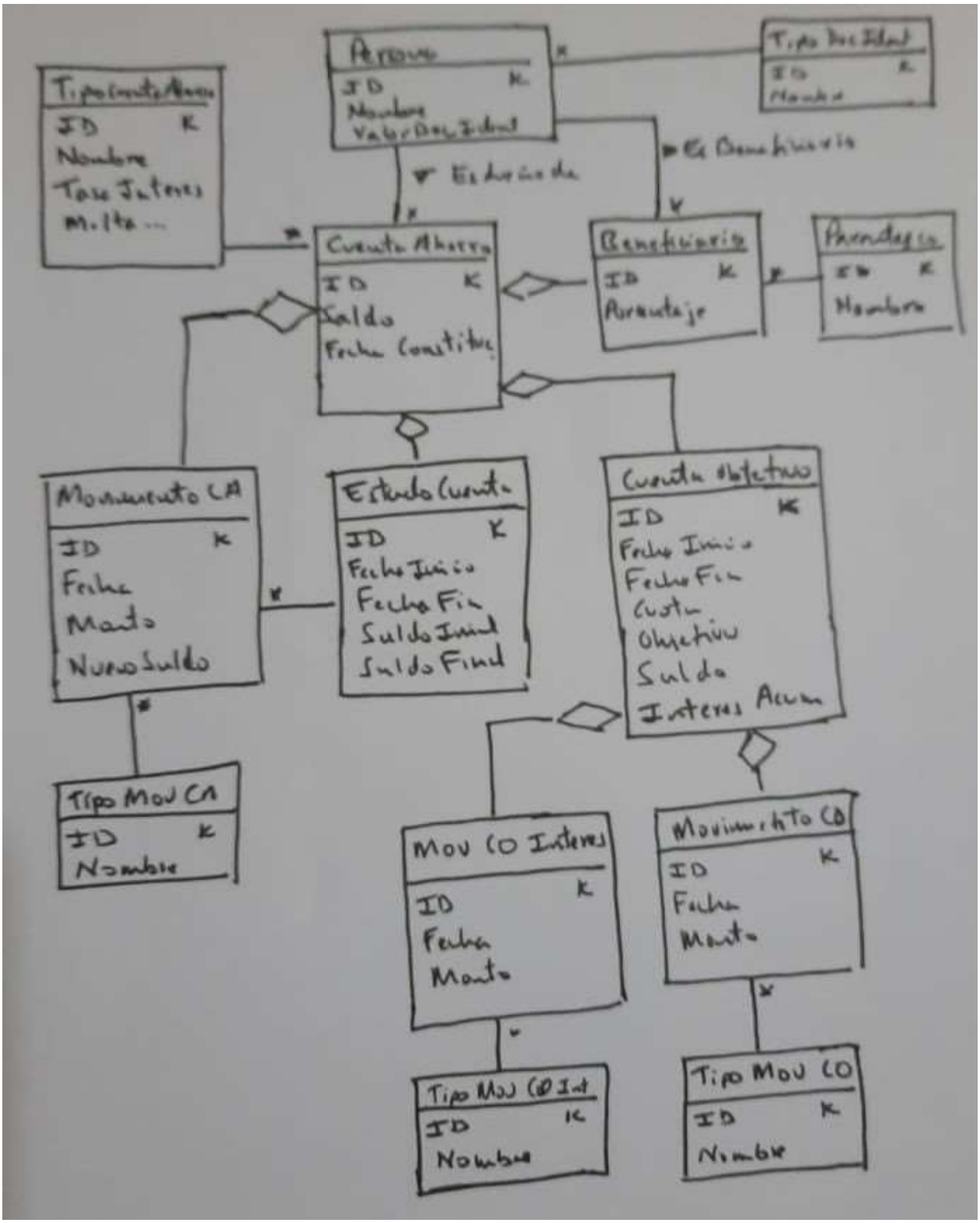
```
        SET NOCOUNT OFF
END;
```

TRES. Lectura de la especificación de la tarea.

Tarea moral: desde SQL, como extraer datos de un documento o string que represente datos en XML o Json.

CUATRO. Modelo conceptual para solución de la tarea programada.

De referencia, no esta completo, hay que hacer correcciones:


Persona
ID K
Nombre
Val. Doc. Ident
Tipo Doc Ident
ID K
Nombre
Tipo Cuenta Ahorro
ID K
Nombre
Tasa Interes
Multa ...
Es duenio de
Es Beneficiario
Cuenta Ahorro
ID K
Saldo
Fecha Constitu
Beneficiario
ID K
Porcentaje
Parentesco
ID K
Nombre
Movimiento CA
ID K
Fecha
Monto
Nuevo Saldo
Estado Cuenta
ID K
Fecha Inicio
Fecha Fin
Saldo Inicial
Saldo Final
Cuenta Objetivo
ID K
Fecha Inicio
Fecha Fin
Cuota
Objetivo
Saldo
Interes Acum
Tipo Mov CA
ID K
Nombre
Mov CO Interes
ID K
Fecha
Monto
Movimiento CO
ID K
Fecha
Monto
Tipo Mov CO Int
ID K
Nombre
Tipo Mov CO
ID K
Nombre