FUNDAMENTOS DE BASE DE DATOS.

Tema Nº10:Introducción a la programación en SQL SERVER I.

Indicador de logro Nº10:Reconoce y emplea los elementos de programación en SQL Server.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº10:**

Introducción a la programación en SQL SERVER I.

**Subtema 10.1:**

Empleo de bloques de códigos de programación en base de datos.

Páginas referenciales:

<https://www.tutorialesprogramacionya.com/sqlserverya/>

<https://www.aulaclic.es/sqlserver/>

**La programación SQL permite interactuar con una base de datos. El lenguaje de consulta estructurado (SQL) es el lenguaje de base de datos más implementado**y valioso para cualquier persona involucrada en la programación informática o que usa bases de datos para recopilar y organizar información.

**¿Para qué sirve la programación SQL?**

La programación SQL se puede usar para compartir y administrar datos, en particular la información organizada en tablas que se encuentra en los sistemas de administración de bases de datos relacionales.

Mediante el uso de SQL, se puede:

* Consultar, actualizar y reorganizar datos.
* Crear y modificar la estructura de un sistema de base de datos.
* Controlar el acceso a sus datos.

**¿Quién debería aprender SQL y con qué propósito?**

**El lenguaje de consulta estructurado y todo lo relacionado con la programación SQL son, a día de hoy, aspectos clave para las empresas**. Por ello, en cualquier organización se pueden necesitar perfiles expertos como los siguientes:

* **Gerentes de producto**: este tipo de profesionales siempre tienen que conocer toda la información accesible acerca del producto y su salud. Gracias a este lenguaje, pueden controlar lo que está sucediendo con él, tanto dentro, como fuera de la empresa.
* **Analistas de datos**: en este caso, SQL es una necesidad para poder extraer todo el valor que encierra la información.
* **Científicos de datos**: son los mayores expertos en datos que ha dado la industria y, **en su currículum suele constatarse que también son especialistas en programación SQL**.
* **Ingenieros de datos**: lo mismo que los científicos de datos y analistas de datos, **para este tipo de profesionales es una necesidad tener nociones de programación SQL**. No hay que olvidar que son ellos quienes diseñan las bases de datos y las mantienen en funcionamiento.
* **Desarrolladores back - end:** esta función es un poco similar a los ingenieros de datos en lo que respecta a la administración de bases de datos. Es difícil encontrar una aplicación back- end sin una base de datos y **ahí es donde entra en juego la** **programación SQL**.
* **Desarrolladores de aplicaciones móviles**: los que se ocupan de aplicaciones móviles, especialmente para Android, están familiarizados con SQLite, una base de datos integrada que se utiliza en proyectos en los que se necesita almacenar los datos en un dispositivo, no en un servidor. Por supuesto, ese dispositivo está alimentado por SQL.
* **Especialistas en marketing: su trabajo está impulsado por los datos y, aunque en este caso no hace falta ser un experto en** **programación SQL**, tener conocimientos de este lenguaje facilita el autoservicio de información, que conducirá a un mejor análisis del negocio.

SQL se utiliza en la industria de la salud, el retail, la educación o las telecomunicaciones. Incluso tiene aplicaciones en la industria de la defensa. Por eso, hoy día, **sectores muy diferentes buscan profesionales con conocimientos en programación SQL**, por ser considerados un activo para la organización en lo que respecta a las consultas.

**Subtema 10.2:**

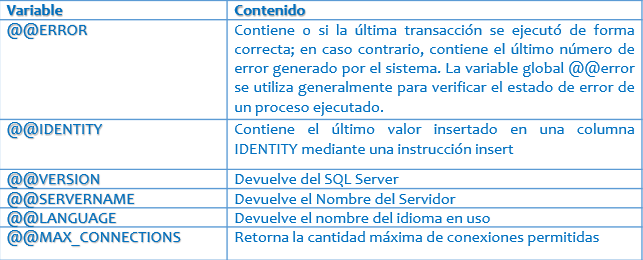
Tipos de datos, Declaración de Variables @, Bloque de códigos BEGIN – END, estructuras de control IF/THEN/ELSE estructura Bucle WHILE.

**DECLARACIÓN DE VARIABLES:**

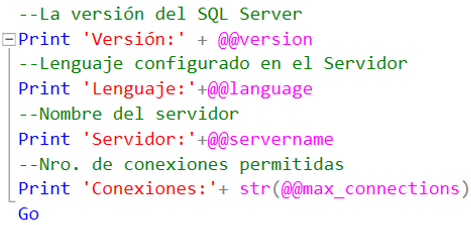
* Una variable es una entidad a la que se asigna un valor. Este valor puede cambiar durante el proceso donde se utiliza la variable. SQL Server tiene dos tipos de variables: locales y globales.
* Las variables locales están definidas por el usuario, mientras que las variables globales las suministra el sistema y están predefinidas.

**VARIABLES PUBLICAS.**

Las variables globales son variables predefinidas suministradas por el sistema. Se distinguen de las variables locales por tener dos símbolos “@”.



**EMPLEO DE ALGUNAS VARIABLES PÚBLICAS.**



**VARIABLES LOCALES (PRIVADAS)**

Las variables locales se declaran, nombran y escriben mediante la palabra clave declare, y reciben un valor inicial mediante una instrucción **Select** o **Set**.

Los nombres de las variables locales deben empezar con el símbolo “@”. A cada variable local se le debe asignar un tipo de dato definido por el usuario o un tipo de dato suministrado por el sistema distinto de text, image o sysname.

**Sintaxis:**

* Declara una variable
  + **Declare** @variable <tipo de dato>
* Asigna valor a una variable
  + **Set** @variable= valor

**/\* V a r i a b l e s L o c a l e s \*/**

**--Ejemplo 1**

Declare @v\_CodLib Varchar(5)

Set @v\_CodLib = '1000'

Select \* From LIBRO L Where L.CODLIB = @v\_CodLib

Go

**--Ejemplo 2**

Declare @mx decimal = 0

Select @mx=MAX(PRECIOLIB) from LIBRO

--imprimir los valores de las variables

print 'El Precio más alto es:'+ str(@mx)

print 'El Precio más alto es:'+ Convert(Varchar,@mx)

go

**--Ejemplo 3**

Declare @v\_mn decimal

Select @v\_mn = MIN(PRECIOLIB) From LIBRO

Print 'El precio mas bajo es:'+str(@v\_mn)

go

**--Ejemplo 4**

Declare @Base real, @Altura real, @At real

Set @Base = 100

Set @Altura = 5

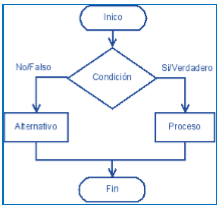
Set @At = (@Base \* @Altura)/2

print 'El Area del Triángulo es: ' + str(@At)

print 'El Area del Triángulo es: ' + Convert(Varchar,@At)

**ESTRUCTURA DE CONTROL**

* IF se utiliza para definir una condición que determina si se ejecutará la instrucción siguiente. La instrucción SQL se ejecuta si la condición se cumple, es decir, si devuelve TRUE (verdadero).
* La palabra ELSE introduce una instrucción SQL alternativa que se ejecuta cuando la condición IF devuelva FALSE.



**ESTRUCTURA DE CONTROL IF.**

**IF** (<Expresión>)

**BEGIN**

…….

**END**

**ELSE IF** <Expresión>

**BEGIN**

……..

**END**

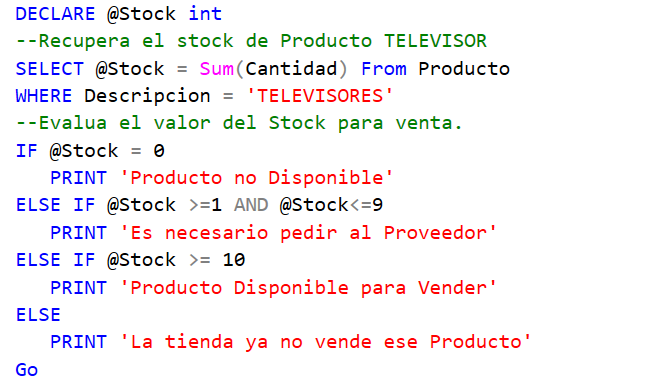
**ELSE**

**BEGIN**

……..

**END**

Ejemplo de la Estructura de control IF.



Aquí desarrollarás el contenido del subtema 1.1.

Ejemplos:

Aquí colocarás los ejemplos que facilitarán la comprensión del subtema

**ESTRUCTURA DE CONTROL CASE:**

* La estructura CASE evalúa una lista de condiciones y devuelve una de las varias expresiones de resultado posibles.
* La expresión CASE tiene dos formatos:
  + La expresión CASE sencilla compara una expresión con un conjunto de expresiones sencillas para determinar el resultado.
  + La expresión CASE buscada evalúa un conjunto de expresiones booleanas para determinar el resultado.
* Ambos formatos admiten un argumento ELSE opcional.

**SINTAXIS:**

**CASE** <expresión>

**WHEN** <valor expresión> THEN <valor devuelto>

**WHEN** <valor\_expresion1> THEN <valor\_devuelto1>

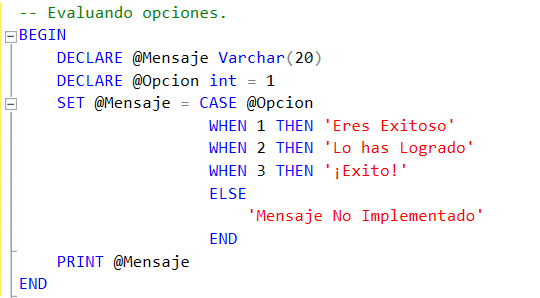
……..

……..

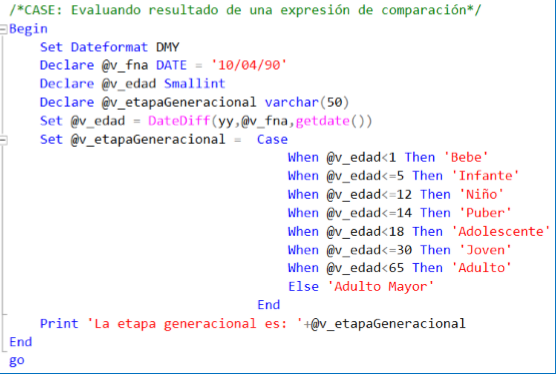
**ELSE** <valor\_devuelto2> -- Valor por defecto

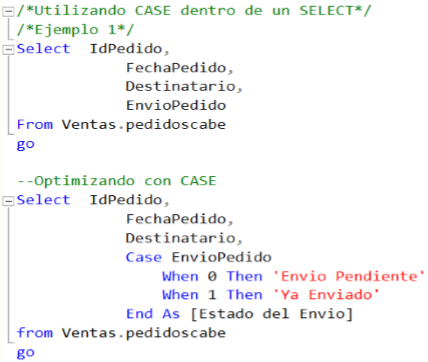
**END**

**Estructura de Control CASE.**



**Estructura de control CASE.**





**ESTRUCTURA DE CONTROL WHILE.**

* Ejecuta en forma repetitiva un conjunto o bloque de instrucciones SQL siempre que la condición especificada sea verdadera
* Se puede controlar la ejecución de instrucciones en el bucle WHILE con las palabras clave BREAK y CONTINUE.

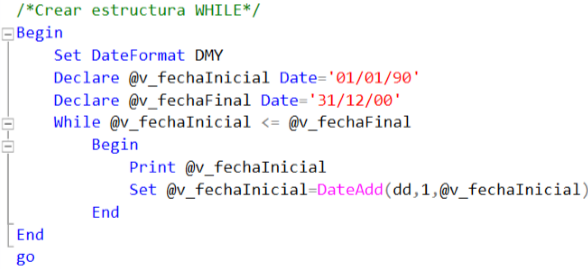
**Sintaxis:**

**WHILE** <Condición>

**BEGIN**

….. Código

**END.**



**Subtema 10.3:**

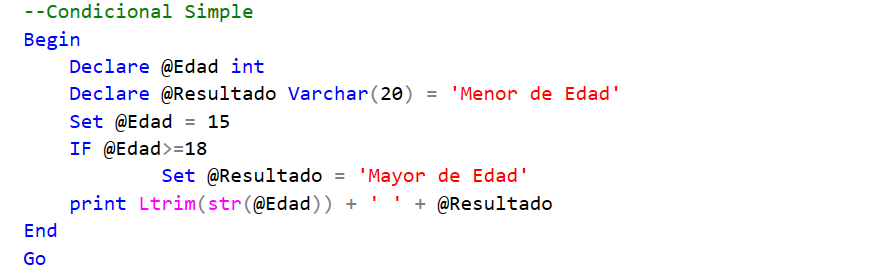
Realizando la recuperación de datos de tablas usando bloques de códigos.

La Programación en SQL Server, son capaces de realizar procesos tanto con operaciones con o sin las tablas que forman parte de una base de datos. Pero de preferencia siempre se busca realizar programación basada en datos. Aquí algunos ejemplos:

**EJEMPLOS SIN USO DE TABLAS:**

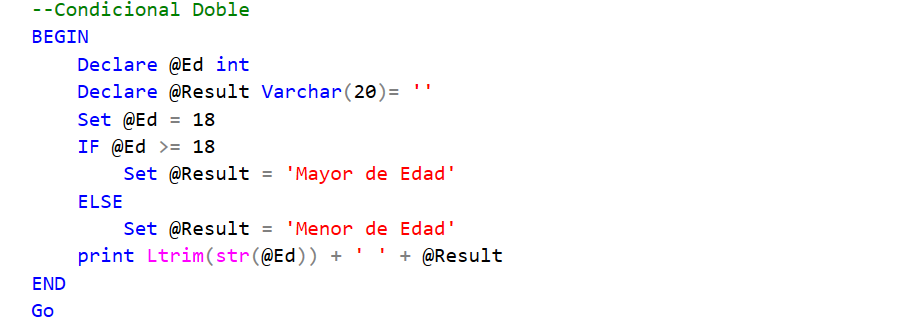
**Ejemplo 1:**

Programa que determina si es o no Mayor de Edad. Según Edad. Usando el formato de condición simple.



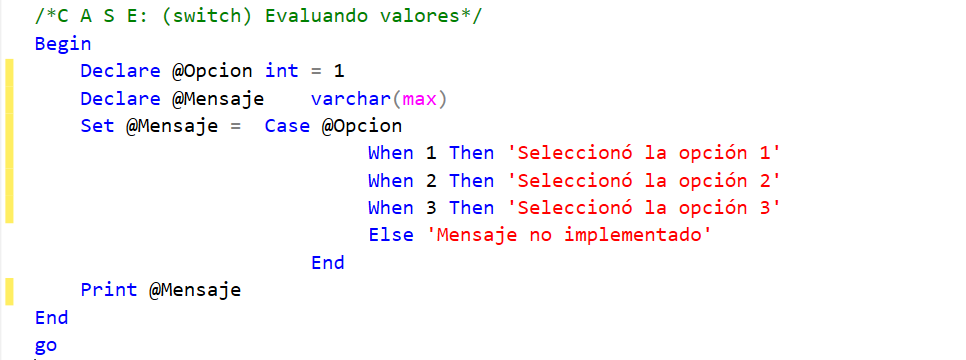
**Ejemplo 2:**

Programa que determina si es o no Mayor de Edad. Según Edad. Usando el formato de condición doble.



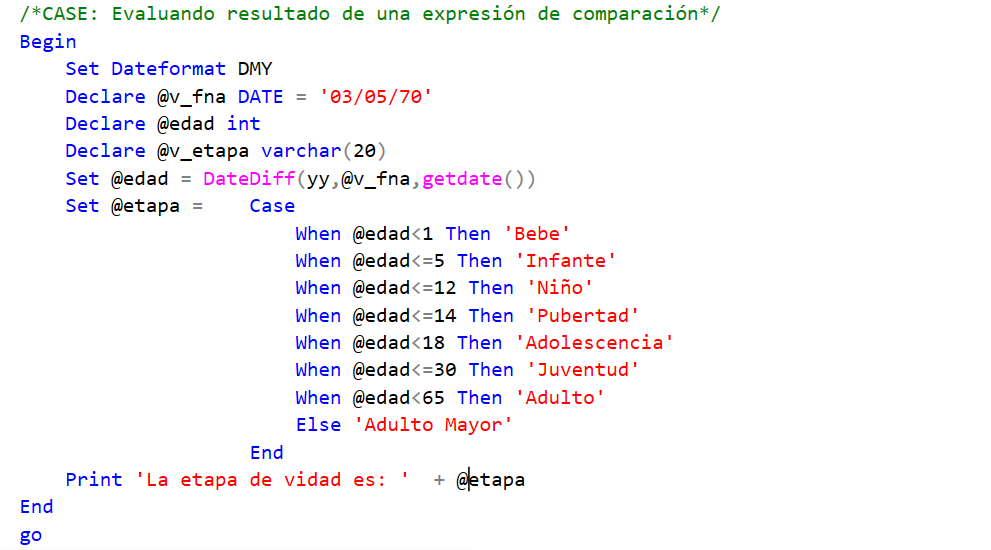
**Ejemplo 3:**

Programa que determina el mensaje a obtener según opción seleccionada.



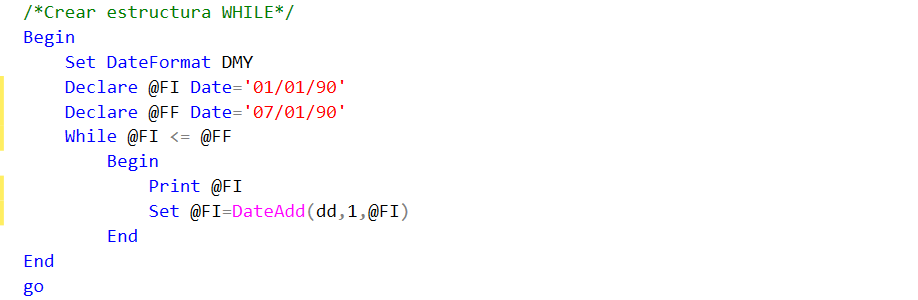
**Ejemplo 4:**

Programa que determina la etapa de ida correspondiente según edad de la persona.



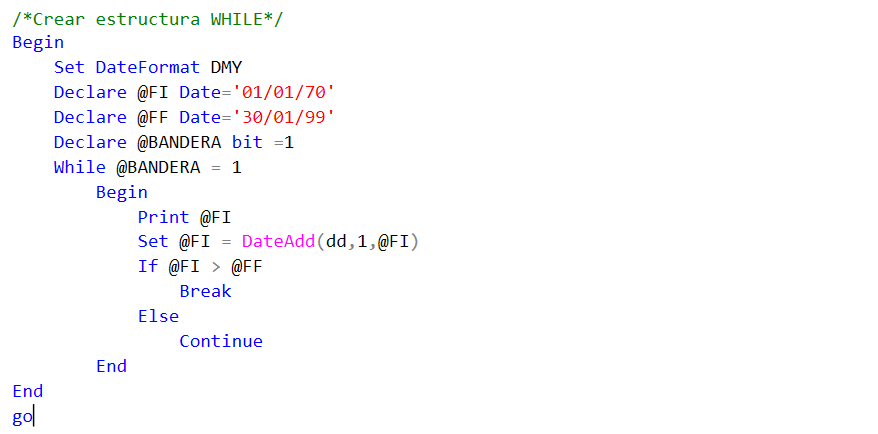
**Ejemplo 5:**

Programa que muestra las fechas a diario entre los días 01/01/90 y 07/01/90



**Ejemplo 6:**

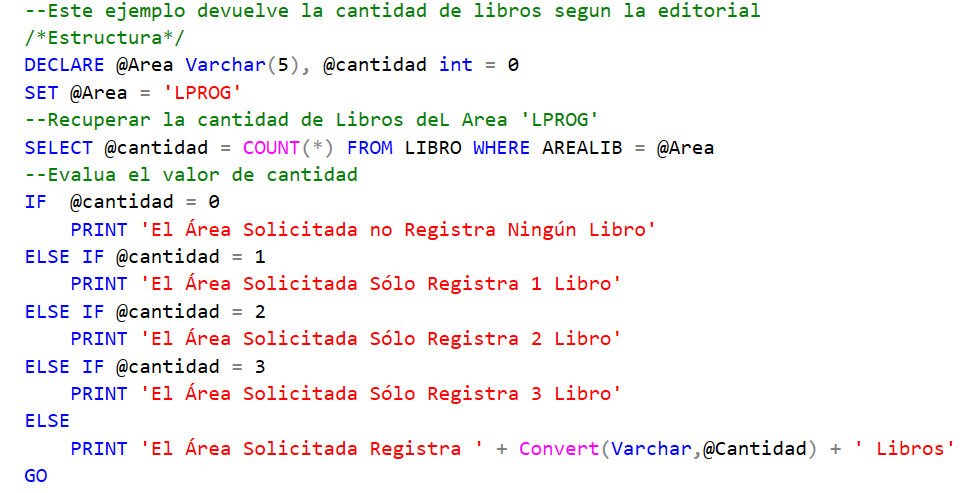
Generador de días comprendidos entre el 01/01/70 al 10/01/99



**EJEMPLOS CON USO DE TABLAS:**

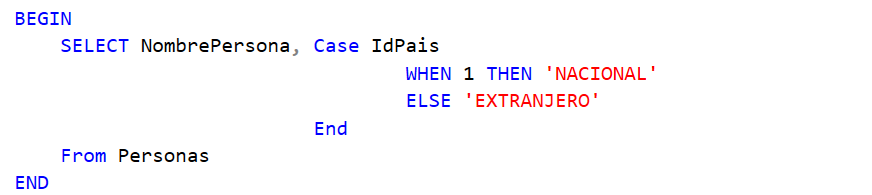
**Ejemplo 1:**

Devolver la cantidad de libros según la editorial.



**Ejemplo 2:**

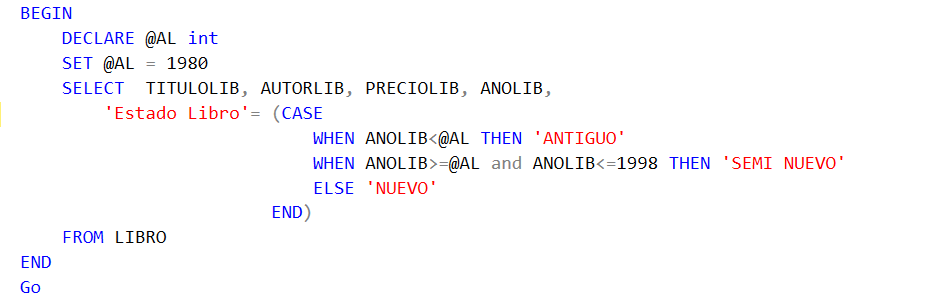
Programa que determina si una persona es de nacionalidad LOCAL o EXTRANJERA.



**Ejemplo 3:**

Cargar los libros (**TITULO**, **AUTOR**, **PRECIO** Y **AÑO DEL LIBRO**) con las siguientes especificaciones:

* Libros que sean menor al año 1980 se consideran antiguos.
* Libros que se encuentren entre los años 1980 y no pasan de los 1998 se consideran semi nuevo.
* Caso contrario a todos es nuevo.



**TEMA IMPORTANTE EN LA PROGRAMACION CON BASE DE DATOS:**

**CONTROL DE ERRORES**

SQL Server proporciona el control de errores a través de las instrucciones TRY y CATCH. Estas nuevas instrucciones suponen un gran paso adelante en el control de errores en SQL Server

La sintaxis de **TRY CATCH** es la siguiente:

**BEGIN TRY**

<EXPRESION\_SQL>

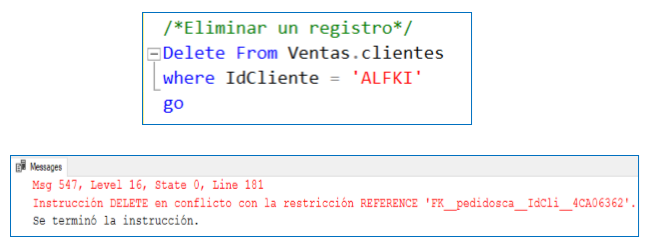
**END TRY**

**BEGIN CATCH**

<EXPRESION\_SQL>

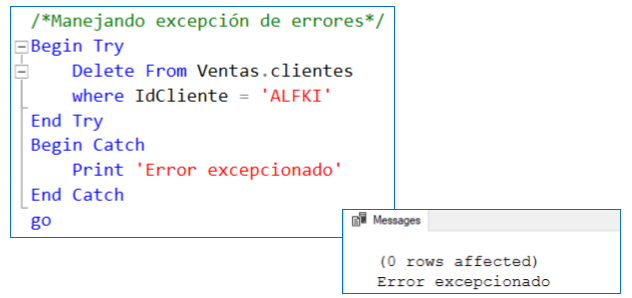
**END CATCH**

Por ejemplo, si tratamos de eliminar un registro **Cliente** que está siendo usado en la tabla **Pedidos**, nos va a provocar un error de Integridad Referencial.



**‘00001’**

La misma sentencia, pero con Excepción de Errores.



**‘00001’**

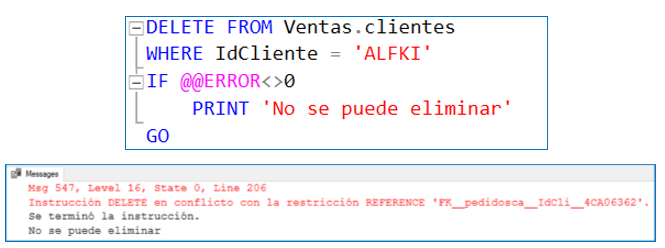
**FUNCIONES ESPECIALES DE ERRORES**

Las funciones especiales de error, están disponibles únicamente en el bloque CATCH para la obtención de información detallada del error:

|  |  |
| --- | --- |
| **ERROR** | **DESCRIPCION** |
| **ERROR\_NUMBER()** | Devuelve el número de error |
| **ERROR\_SEVERITY()** | Devuelve la severidad del error. |
| **ERROR\_STATE()** | Devuelve el estado del error. |
| **ERROR\_PROCEDURE()** | Devuelve el nombre del procedimiento almacenado que ha provocado el error. |
| **ERROR\_LINE()** | Devuelve el número de línea en la que se ha producido el error. |
| **ERROR\_MESSAGE()** | Devuelve el mensaje de error. |

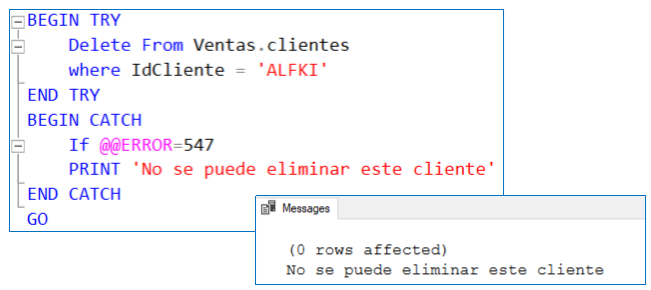
VARIABLE **@@ERROR.**

Devuelve el número de error de la última instrucción TRANSACT-SQL ejecutada; si la variable devuelve 0, la TRANSACT-SQL anterior no encontró errores.



**‘00001’**

Otra forma de implementar este proceso, es controlándolo a través del bloque TRY CATCH.

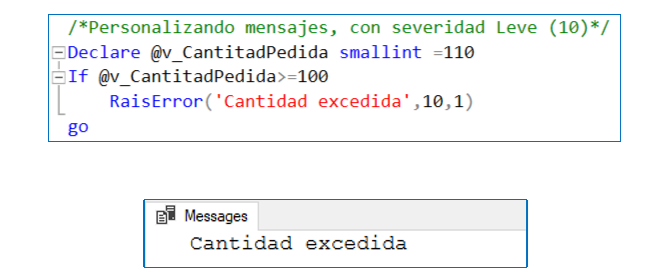


**‘00001’**

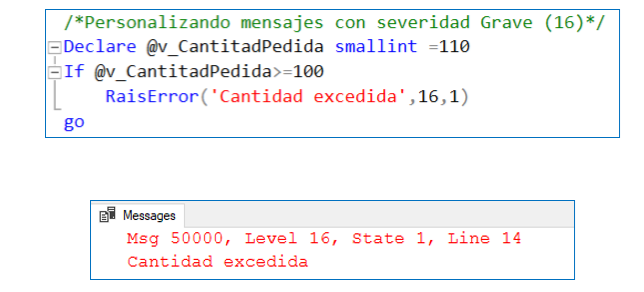
**RAISERROR**

* En ocasiones es necesario provocar voluntariamente un error, por ejemplo, nos puede interesar que se genere un error cuando los datos incumplen una regla de negocio
* Podemos provocar un error en tiempo de ejecución a través de la función RAISERROR.
* La función RAISERROR recibe tres parámetros, el mensaje del error (o código de error predefinido), la severidad y el estado.

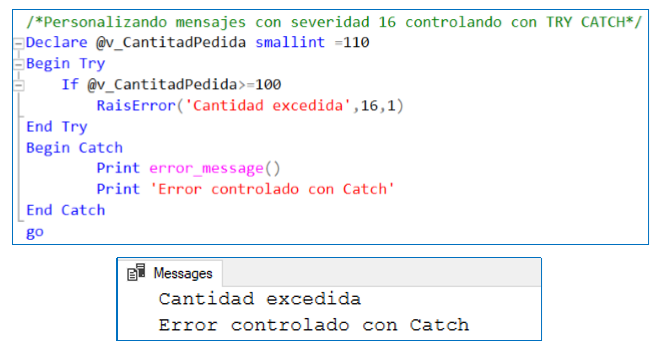
**EJEMPLO 1:**



**EJEMPLO 2:**



**EJEMPLO 3:**



**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

* La programación SQL permite interactuar con una base de datos. El lenguaje de consulta estructurado (SQL) es el lenguaje de base de datos más implementado y valioso para cualquier persona involucrada en la programación informática o que usa bases de datos para recopilar y organizar información.
* Importancia de las variables, estructuras de control y bucles, en el desarrollo de códigos del entorno de la base de datos SQL Server.
* Una transacción es un conjunto de operaciones Transact SQL que se ejecutan como un único bloque. • RaisError genera un mensaje de error e inicia el procesamiento de errores para la sesión. RAISERROR puede hacer referencia a un mensaje definido por el usuario.

**ACTIVIDAD VIRTUAL**

Analizar y revisar detenidamente este recurso de aprendizaje y luego responde las preguntas propuestas.

* ¿Qué es un bloque de programación BEGIN – END en SQL SERVER??
* Importancia de la programación en bases de datos.
* ¿Qué es una variable y que finalidad cumple??
* ¿Qué es una estructura de control?
* ¿Qué es un bucle en SQL Server?
* ¿Errores de Excepción, tipos y que utilidad tienen en los programas?