PROCEDIMIENTOS ALMACENADOS

- Un procedimiento almacenado es un conjunto de sentencias SQL y de control de flujo
- Procedimientos almacenados definidos por el usuario
  Son procedimientos definidos por el usuario que se deben llamar explícitamente

Los procedimientos almacenados son un conjunto pre compilado de instrucciones Transact-SQL (*) almacenadas bajo un solo nombre y procesadas como una unidad. Los procedimientos almacenados pueden recibir parámetros -en base a los cuales realizar distintas acciones- y devolver datos de varias formas distintas (principalmente como parámetros de salida y como conjuntos de resultados o Recordsets).

Características

Una vez creado un procedimiento almacenado, se puede invocar directamente desde una aplicación, o sustituir el nombre de una tabla o vista, por el nombre de procedimiento en cláusulas SELECT. Los procedimientos almacenados pueden recibir parámetros de entrada y retornar valores a la aplicación

Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.

Ventajas

- Simplifican la ejecución de tareas repetitivas
- Corren más rápido que las mismas instrucciones ejecutadas en forma interactiva
- Reducen el tráfico a través de la red
- Pueden capturar errores antes que ellos puedan entrar a la base de datos
- Establece consistencia porque ejecuta las tareas de la misma forma
- Permite el desarrollo modular de aplicaciones
- Ayuda a proveer seguridad
- Puede forzar reglas y defaults complejos de los negocios

Desventajas

Se corrompe la base de datos adiós a todo el trabajo hasta el último respaldo que tengas.
Esclavitud: los procedimientos almacenados nos esclavizan al motor de base de datos.
Procedimientos que necesitan acceder a algunas estructuras de datos externas que no tienen una interface tradicional de acceso a datos.
Realizar algoritmos complejos. Por ejemplo algoritmos geográficos de mapeo.
Realizar complejos cálculos de ingeniería.

Modelo de empleo

Veamos la sintaxis.

```
CREATE PROCEDURE [ schema_name. ] procedure_name [ ;number ]

[ ( ] [ { @parameter [ type_schema_name. ] data_type }

[ VARYING ] [ = default ] [ OUT | OUTPUT ] ] [ , ...n ] [ ) ]

[ WITH procedure_option [ , ...n ] ]

[ FOR REPLICATION ]

AS

{ sql_statement [;][ ... n ] }
```

3 ejemplos

Procedimientos almacenados en MySQL
Desde MySQL 5 los procedimientos almacenados empezaron a ser soportados, como suele suceder en MySQL las sentencias se ejecutan luego de escribir el signo punto y coma (;), por esta razón antes de escribir el procedimiento almacenado la función del punto y coma se asigna a otros caracteres usando la sentencia DELIMITER seguida de un caracter tal como |, de esta manera el procedimiento puede ser escrito usando los punto y comas sin que se ejecute mientras se escribe; después de escrito el procedimiento, se escribe nuevamente la sentencia DELIMITER ; para asignar al punto y coma su función habitual.Fven
El siguiente es un ejemplo de procedimiento almacenado en MySQL:

```
DELIMITER |
 CREATE PROCEDURE autos(IN velocidad INT,IN marca VARCHAR(50))
BEGIN
IF velocidad < 120 THEN
INSERT INTO familiares VALUES(velocidad,marca);
ELSE
INSERT INTO deportivos VALUES(velocidad,marca);
END IF;
END;
```

Como se pueden dar cuenta se inicia con un CREATE PROCEDURE seguido del nombre del SP. En el nombre se puede especificar el esquema, el nombre y el numero (opcional). El esquema siempre debe ser incluido ya que asegura que tu SP, es creado en el esquema apropiado. El procedure_name es el nombre que le asignamos a nuestro SP. Podemos añadir un numero para algun grupo, pero esto lo debemos hacer con un ;

```
CREATE PROCEDURE dbo.MiStoreProcedure


AS
    SET NOCOUNT ON
    CREATE TABLE Prueba(id INT PRIMARY KEY NOT NULL);
    INSERT INTO Prueba(id)VALUES(1);
   RETURN
```

Este pequeño ejemplo crea una tabla en mi Base de Datos, e ingresa un valor en el campo de Identificador.

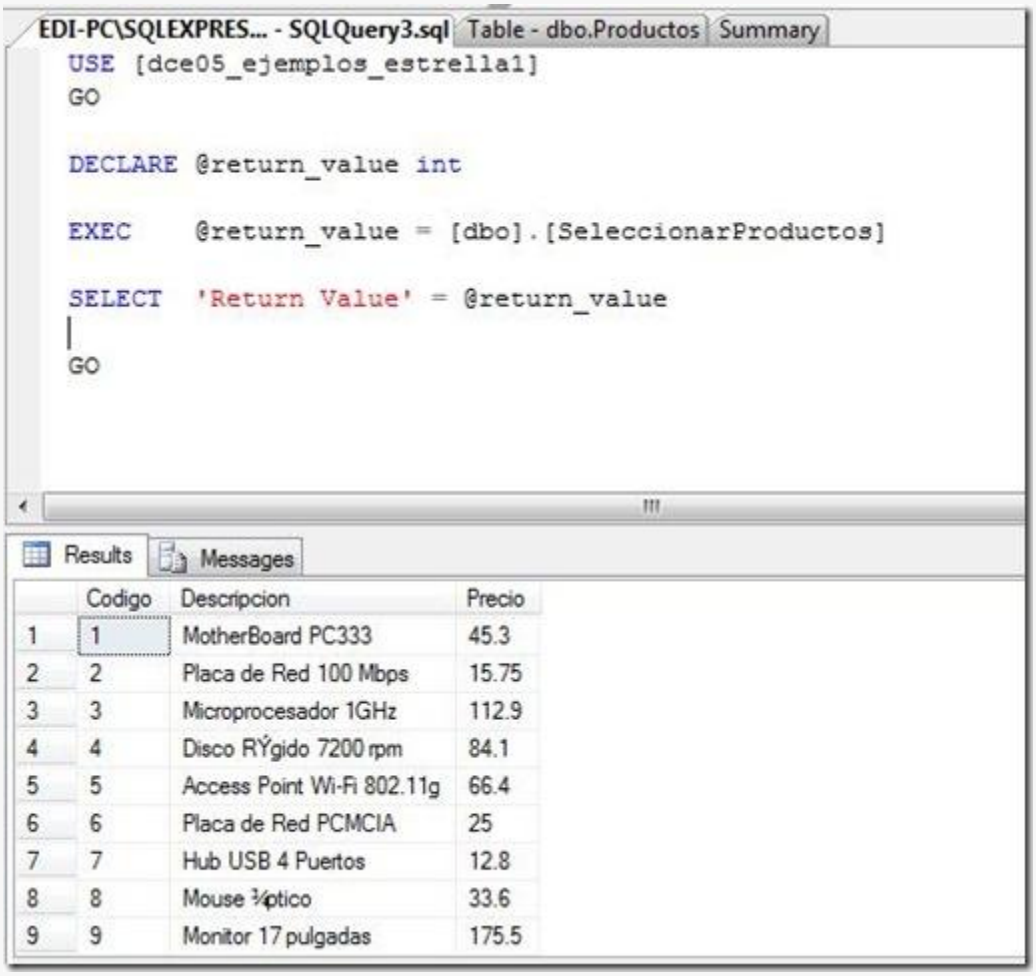
```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE dbo.SeleccionarProductos


AS
BEGIN
   SELECT
   Codigo,
   Descripcion,
   Precio
```

```
    FROM
        Productos
    RETURN
END
GO
```

Aquí hago una consulta sencilla con un SELECT a una tabla Productos



	Codigo	Descripcion	Precio
1	1	MotherBoard PC333	45.3
2	2	Placa de Red 100 Mbps	15.75
3	3	Microprocesador 1GHz	112.9
4	4	Disco RÝgido 7200 rpm	84.1
5	5	Access Point Wi-Fi 802.11g	66.4
6	6	Placa de Red PCMCIA	25
7	7	Hub USB 4 Puertos	12.8
8	8	Mouse ¾ptico	33.6
9	9	Monitor 17 pulgadas	175.5

Podemos llamar a nuestros procedimientos almacenados de la siguiente manera.

```
execute SeleccionarProductos
```

TRIGEER

¿Qué es un triger o disparador o desencadenador para SQL server?
Es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos.

SQL Server proporciona los siguientes tipos de triggers:

Trigger DML, se ejecutan cuando un usuario intenta modificar datos mediante un evento de lenguaje de manipulación de datos (DML). Los eventos DML son instrucciones INSERT, UPDATE o DELETE de una tabla o vista.

Trigger DDL, se ejecutan en respuesta a una variedad de eventos de lenguaje de definición de datos (DDL). Estos eventos corresponden principalmente a instrucciones CREATE, ALTER y DROP de Transact-SQL, y a determinados procedimientos almacenados del sistema que ejecutan operaciones de tipo DDL.

Características
Registrar, auditar y monitorear la actividad de cambio de datos
•Validar datos, cambiando o negando acciones como INSERT, UPDATE, DELETE en una tabla
•Preservar la consistencia y claridad de los datos ejecutando acciones relacionadas con otras tablas.

Ventajas
Seguridad de los datos mejorada
•Ofrecen chequeos de seguridad basada en valores.
Integridad de los datos mejorada
•Fuerzan restricciones dinámicas de integridad de datos y de integridad referencial.
•Aseguran que las operaciones relacionadas se realizan juntas de forma implícita.
•Respuesta instantánea ante un evento auditado
•Ofrece un mayor control sobre la B.D.

Desventajas
- Solo se pueden aplicar a una tabla específica, es decir, un trigger no sirve para dos o más tablas
- El trigger se crea en la base de datos que de trabajo pero desde un trigger puedes hacer referencia a otras bases de datos.
- Un Trigger devuelve resultados al programa que lo desencadena de la misma forma que un Stored Procedure aunque no es lo mas idoneo, para impedir que una instrucción de asignación devuelva un resultado se puede utilizar la sentencia SET NOCOUNT al principio del Trigger.
- Las siguientes instrucciones no se pueden utilizar en los triggers:
ALTER DATABASE CREATE DATABASE
DISK INIT DISK RESIZE
DROP DATABASE LOAD DATABASE
LOAD LOG RECONFIGURE
RESTORE DATABASE RESTORE LOG
Modelo de ejemplo

3 ejemplos. Primer ejemplo
El siguiente ejemplo, graba un historico de saldos cada vez que se modifica un saldo de la tabla cuentas.

```
CREATE TRIGGER TR_CUENTAS
ON CUENTAS
AFTER UPDATE
AS
BEGIN
-- SET NOCOUNT ON impide que se generen mensajes de texto
-- con cada instrucción
SET NOCOUNT ON;
INSERT INTO HCO_SALDOS
(IDCUENTA, SALDO, FXSALDO)
SELECT IDCUENTA, SALDO, getdate()
FROM INSERTED
END
```

La siguiente instrucción provocará que el trigger se ejecute:

```
UPDATE CUENTAS

SET SALDO = SALDO + 10

WHERE IDCUENTA = 1
```

Segundo ejemplo