[](http://www.google.com/imgres?imgurl=http://www.addenda.com.mx/blog/wp-content/uploads/2011/03/sql-server-logo-411303.jpg&imgrefurl=http://www.addenda.com.mx/blog/?cat=6&usg=__B61t7_HO-JZUMCtbXG3U48e-qso=&h=262&w=319&sz=17&hl=es&start=8&zoom=1&tbnid=ItihwXdKCTv8aM:&tbnh=97&tbnw=118&ei=6bWoTargJ-qY0QHk-8X5CA&prev=/search?q=SQL&hl=es&biw=1051&bih=559&gbv=2&tbm=isch&itbs=1) LENGUAJE DE CONSULTA ESTRUCTURADO (SQL)

**Microsoft SQL Server**

**(Lenguaje de Base de Datos Normalizado)**

[](http://www.google.com/imgres?imgurl=http://www.addenda.com.mx/blog/wp-content/uploads/2011/03/sql-server-logo-411303.jpg&imgrefurl=http://www.addenda.com.mx/blog/?cat=6&usg=__B61t7_HO-JZUMCtbXG3U48e-qso=&h=262&w=319&sz=17&hl=es&start=8&zoom=1&tbnid=ItihwXdKCTv8aM:&tbnh=97&tbnw=118&ei=6bWoTargJ-qY0QHk-8X5CA&prev=/search?q=SQL&hl=es&biw=1051&bih=559&gbv=2&tbm=isch&itbs=1)

**Desencadenador(Trigger ó Disparador)**

Un trigger (o disparador) en una [Base de datos](http://es.wikipedia.org/wiki/Base_de_datos) , es un [procedimiento](http://es.wikipedia.org/wiki/Subrutina) que se ejecuta cuando se cumple una condición establecida al realizar una operación. Dependiendo de la base de datos, los triggers pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE). Algunas bases de datos pueden ejecutar triggers al crear, borrar o editar usuarios, tablas, bases de datos u otros objetos.

Son usados para mejorar la administración de la Base de datos, sin necesidad de contar con que el usuario ejecute la sentencia de [SQL](http://es.wikipedia.org/wiki/SQL).

Además, pueden generar valores de [columnas](http://es.wikipedia.org/wiki/Columna_(base_de_datos)), previene errores de datos, sincroniza [tablas](http://es.wikipedia.org/wiki/Tabla_(base_de_datos)), modifica valores de una [vista](http://es.wikipedia.org/wiki/Vista_(base_de_datos)), etc.

Permite implementar programas basados en paradigma lógico (sistemas expertos, deducción).

## Componentes principales

La estructura básica de un trigger es:

* Llamada de activación: es la sentencia que permite "disparar" el código a ejecutar.
* Restricción: es la condición necesaria para realizar el código. Esta restricción puede ser de tipo condicional o de tipo nulidad.
* Acción a ejecutar: es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

## Tipos

Existen dos tipos de disparadores que se clasifican según la cantidad de ejecuciones a realizar:

* Row Triggers (o Disparadores de [fila](http://es.wikipedia.org/wiki/Fila_(base_de_datos))): son aquellas que se ejecutaran n-veces si se llama n-veces desde la [tabla](http://es.wikipedia.org/wiki/Tabla_(base_de_datos)) asociada al trigger
* Statement Triggers (o Disparadores de secuencia): son áquellos que sin importar la cantidad de veces que se cumpla con la condición, su ejecución es única.

Pueden ser de sesión y almacenados; pero no son de fiar[[cita requerida](http://es.wikipedia.org/wiki/Wikipedia:Verificabilidad)].

## Efectos y características

* No aceptan parámetros o argumentos (pero podrían almacenar los datos afectados en tablas temporales)
* No pueden ejecutar las operaciones [**COMMIT**](http://es.wikipedia.org/wiki/Commit) o [**ROLLBACK**](http://es.wikipedia.org/wiki/Rollback)por que estas son parte de la sentencia SQL del disparador (únicamente a través de transacciones autónomas)
* Pueden causar errores de mutaciones en las tablas, si se han escrito de manera deficiente.

-Al agregar un nuevo pedido a la tabla de PEDIDOS

--se debe incrementar las ventas del representante

--que concreto el pedido, así como también debe

--reducirse el número de existencias.

--Para ello debe crear el siguiente Desencadenador:

--select \* from productos where id\_fab='ACI' AND ID\_PROD='41001'

--select nombre,cuota,ventas from RepVentas where nombre='jorge castro'

--GO

--Use desencadenadores

--go

--CREATE DESENCADENADOR np

--ON pedidos FOR INSERT AS

--UPDATE RepVentas

--SET VENTAS =VENTAS + INSERTED.CUOTA FROM REPVENTAS INNER JOIN INSERTED

--ON REPVENTAS.NOMBRE= INSERTED.REP

--UPDATE PRODUCTOS

--SET EXISTENCIAS = EXISTENCIAS - INSERTED.CANT FROM PRODUCTOS INNER JOIN INSERTED

--ON PRODUCTOS.ID\_FAB = INSERTED.FAB

--AND PRODUCTOS.ID\_PRODUCTO = INSERTED.PRODUCTO

--GO

**Procedimientos almacenados**

Un stored procedure es una colección de sentencias del Transact-SQL las cuales organizadas lógicamente resuelven algunas de las operaciones transaccionales que requieren los usuarios, estos procedimientos se almacenan en la base de datos. Los procedimientos almacenados soportan el empleo de variables declaradas por el usuario, sentencias para toma de decisiones entre otras características.

En SQL Server existen 5 tipos de procedimientos almacenados:

• Procedimientos del sistema, son los que se encuentran almacenados en la base de datos **master** y algunas en las bases de datos de usuario, estos procedimientos almacenados brindan información acerca de los datos y características del servidor. En el nombre usan como prefijo **sp\_**.

• Procedimientos locales, son los procedimientos almacenados en una base de datos.

• Procedimientos temporales, son procedimientos locales y sus nombres empiezan con los prefijos # o ##, dependiendo si se desea que sea un procedimiento global a todas las conecciones o local a la coneccion que lo define.

• Procedimientos remotos, son procedimientos almacenados en servidores distribuidos.

• Procedimientos extendidos, son aquellos que nos permiten aprovechar las funcionalidades de otras librerías externas a SQL Server. Estos procedimientos usan el prefijo **xp\_** y se encuentran en la base de datos **master**.

Entre las principales características de un procedimiento almacenado podemos mencionar:

• Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.

• Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.

• Devolver un valor de estado que indica si la operación se ha realizado correctamente o habido un error (y el motivo del mismo).

• Permiten una ejecución más rápida, ya que los procedimientos son analizados y optimizados en el momento de su creación, y es posible utilizar una versión del procedimiento que se encuentra en la memoria después de que se ejecute por primera vez.

• Pueden reducir el tráfico de red.

• Pueden utilizarse como mecanismo de seguridad, ya que se puede conceder permisos a los usuarios para ejecutar un procedimiento almacenado, incluso si no cuentan con permiso para ejecutar directamente las instrucciones del procedimiento.

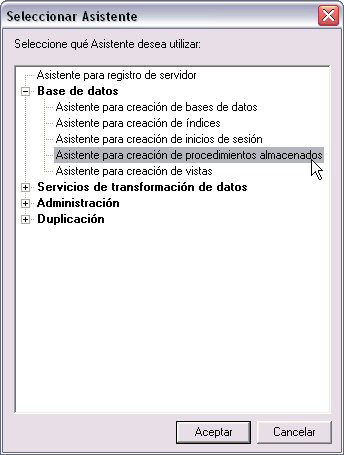
Crear, Modificar y Eliminar un **Procedimiento Almacenado**

## Crear Procedimientos Almacenados

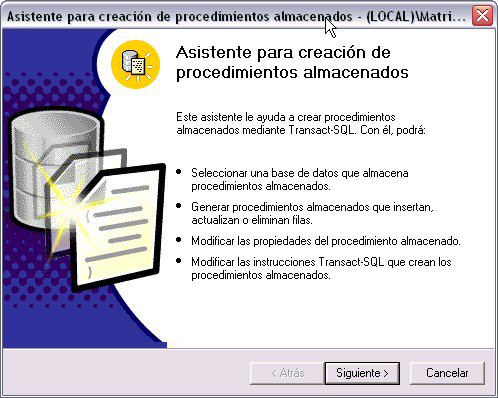
Para crear un stored procedure en SQL Server tiene la posibilidad de utilizar múltiples formas entre ellas un asistente para la creación de procedimientos para ingresar, eliminar y actualizar información en las tablas.

Siga las instrucciones para aprovechar el asistente:

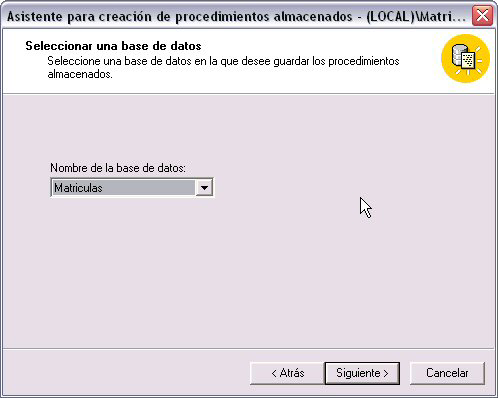
* 1. En el menú Herramientas haga clic sobre la opción Asistentes, y seleccione la opción que muestra la figura:



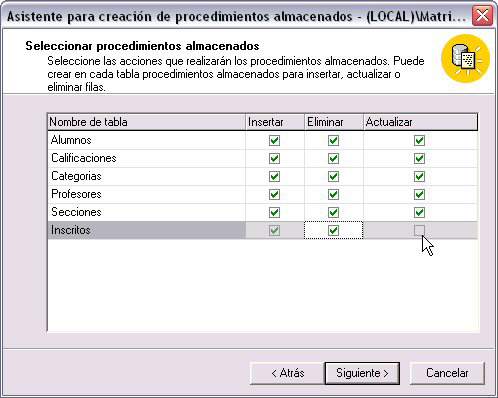
2. Luego de pulsar Aceptar, aparecerá una pantalla de bienvenida al asistente



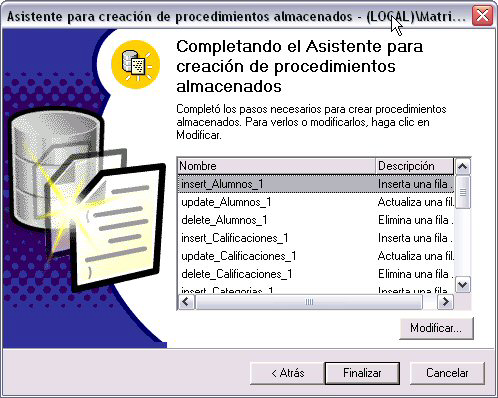
1. Pulse Siguiente y tendrá la posibilidad de elegir que base de datos utilizará.



4. Pulse Siguiente y marque las casillas que indican que procedimientos creará.



Luego de pulsar Siguiente aparecerá una pantalla que indica el fin de los pasos requeridos para crear los procedimientos.



5. Al pulsar Finalizar aparecerá un mensaje indicando cuantos procedimientos se han creado.



También puede crear los procedimientos con la sentencia CREATE PROCEDURE.

**CREATE PROC[EDURE] <Nombre Procedimiento>**

**[**

**{@*parámetro tipoDatos*} [= *predeterminado*] [OUTPUT]**

**]**

**[,...*n*]**

**[WITH**

**{ RECOMPILE**

**| ENCRYPTION**

**}**

**] AS**

***Sentencias SQL* [...*n*]**

**Argumentos**

**@parámetro**

El usuario puede tener hasta máximo de 1024 parámetros. El nombre del parámetro debe comenzar con un signo (@) . Los parámetros son locales al procedimiento.

**default**

**OUTPUT**

Es un valor predeterminado para el parámetro.

Indica que se trata de un parámetro de salida. El valor de esta opción puede devolverse a EXEC[UTE]. Utilice los parámetros OUTPUT para devolver información al procedimiento que llama. Los parámetros de texto no se pueden utilizar como parámetros OUTPUT.

**{RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}**

RECOMPILE indica que SQL Server no almacena en la caché un plan para este procedimiento, con lo que el procedimiento se vuelve a compilar cada vez que se ejecuta. Utilice la opción RECOMPILE cuando emplee valores atípicos o temporales para no anular el plan de ejecución que está almacenado en la memoria caché.

ENCRYPTION indica que SQL Server codifica la entrada de la tabla syscomments que contiene el texto de la instrucción CREATE ROCEDURE.

Entre otras observaciones podemos mencionar:

• El tamaño máximo de un procedimiento es de 128 Mb.

• Un procedimiento sólo puede crearse en la base de datos actual.

• Se puede crear otros objetos de base de datos dentro de un procedimiento almacenado. Puede hacer referencia a un objeto creado en el mismo procedimiento almacenado, siempre que se cree antes de que se haga referencia al objeto.

• Puede hacer referencia a tablas temporales dentro de un procedimiento almacenado.

Si crea una tabla temporal privada dentro de un procedimiento almacenado, la tabla temporal existirá únicamente para los fines del procedimiento; desaparecerá cuando éste finalice.

Si ejecuta un procedimiento almacenado que llama a otro procedimiento almacenado, el procedimiento al que se llama puede tener acceso a todos los objetos creados por el primer procedimiento, incluidas las tablas temporales.

Si se ejecuta un procedimiento almacenado remoto que realiza cambios en un servidor SQL Server remoto, los cambios no se podrán deshacer. Los procedimientos almacenados remotos no intervienen en las transacciones.

Las siguientes instrucciones no se pueden emplear dentro de un procedimiento.

CREATE CREATE

DEFAULT DESENCADENADOR

CREATE

PROCEDURE CREATE VIEW

CRATE RULE

Para crear un procedimiento desde el Analizador de Consultas ejecute las siguientes instrucciones:

**Use Matriculas**

**GO**

**CREATE PROCEDURE ListaPromedios as**

**SELECT NomApe = (Nom + ' ' + Pat + ' ' + Mat), Inscritos.Sec,**

**Curso = Case**

**When SubString(Inscritos.Sec, 2, 1)='1'**

**Then 'V.Basic'**

**When SubString(Inscritos.Sec, 2, 1)='2'**

**Then 'V.Fox'**

**When SubString(Inscritos.Sec, 2, 1)='3'**

**Then 'SQL Server'**

**End,**

**Promedio = (N1+N2)/2**

**From Alumnos INNER JOIN Inscritos**

**On Alumnos.codalu = Inscritos.codalu**

**INNER JOIN Calificaciones**

**On Inscritos.codalu = Calificaciones.codalu**

**AND Inscritos.sec = Calificaciones.sec**

**GO**

Para poder ejecutar el procedimiento emplearemos la siguiente sintaxis:

**EXEC ListaPromedios**

**GO**

Para ver la información de la implementación del procedimiento almacenado:

**Sp\_HelpText ListaPromedios**

**GO**

Para ver cuales son las columnas que producen la información presentada por el procedimiento:

**Sp\_Depends ListaPromedios**

**GO**

**Modificar Procedimientos Almacenados**

Si se desea modificar el procedimiento almacenado utilice la siguiente sintaxis:

***ALTER PROCEDURE <Nombre Procedimiento>***

***[***

***{@parámetro tipoDatos} [= predeterminado] [OUTPUT]***

***]***

***[,...n]***

***[WITH***

***{ RECOMPILE***

***| ENCRYPTION***

***}***

***] AS***

***Sentencias SQL [...n]***

**Para ver un ejemplo realice lo siguiente:**

**ALTER PROCEDURE ListaPromedios WITH ENCRYPTION**

**as**

**SELECT NomApe = (Nom + ' ' + Pat + ' ' + Mat), Inscritos.Sec,**

**Curso = Case**

**When SubString(Inscritos.Sec, 2, 1)='1'**

**Then 'V.Basic'**

**When SubString(Inscritos.Sec, 2, 1)='2'**

**Then 'V.Fox'**

**When SubString(Inscritos.Sec, 2, 1)='3'**

**Then 'SQL Server'**

**End,**

**Promedio = (N1+N2)/2**

**From Alumnos INNER JOIN Inscritos**

**On Alumnos.codalu = Inscritos.codalu**

**INNER JOIN Calificaciones**

**On Inscritos.codalu = Calificaciones.codalu**

**AND Inscritos.sec = Calificaciones.sec**

**GO**

***Ejemplo:***

Ahora implementaremos un procedimiento que muestre el promedio de cada alumno de acuerdo a la sección indicada en el argumento:

**CREATE PROCEDURE PromPorSeccion**

**(@seccion char(4) = NULL)**

**as**

**IF @seccion IS NULL**

**BEGIN**

**RAISERROR ('Debe indicar un codigo', 10, 1)**

**RETURN**

**END**

**IF NOT EXISTS (Select Sec From Inscritos Where Sec=@seccion)**

**BEGIN**

**RAISERROR ('La seccion no tiene alumnos', 10, 1)**

**RETURN**

**END**

**SELECT NomApe = (Nom + ' ' + Pat + ' ' + Mat), Inscritos.Sec,**

**Curso = Case**

**When SubString(Inscritos.Sec, 2, 1)='1'**

**Then 'V.Basic'**

**When SubString(Inscritos.Sec, 2, 1)='2'**

**Then 'V.Fox'**

**When SubString(Inscritos.Sec, 2, 1)='3'**

**Then 'SQL Server'**

**End,**

**Promedio = (N1+N2)/2**

**From Alumnos INNER JOIN Inscritos**

**On Alumnos.codalu = Inscritos.codalu**

**INNER JOIN Calificaciones**

**On Inscritos.codalu = Calificaciones.codalu**

**AND Inscritos.sec = Calificaciones.sec**

**WHERE Inscritos.Sec = @Seccion**

**GO**

Para ejecutar realizaremos lo siguiente:

EXEC PromPorSeccion GO

EXEC PromPorSeccion ‘9090’ GO

EXEC PromPorSeccion ‘2315’ GO

Sp\_HelpText PromPorSeccion GO

Sp\_Depends PromPorSeccion GO

## Eliminar Procedimientos Almacenados

Para eliminar un procedimiento utilice el siguiente formato:

**DROP PROCEDURE <Nombre del procedimiento>**