

Stored Procedure (Procedimientos Almacenados)

Un Stored Procedure o Procedimiento Almacenado es un programa (o función) que se almacena físicamente en una base de datos. La implementación de un stored procedure varía de un gestor de bases de datos a otro. En la mayoría de los casos, no obstante, los stored procedures permiten definir una API (capa de abstracción) para que el gestor de bases de datos interactúe con tablas u otros objetos de la base de datos, en vez de tener un cliente de la aplicación interactuando directamente.

Aunque la mayoría de los gestores de bases de datos existentes brindan soporte para estos procedimientos, existe discusión entre los desarrolladores acerca de las ventajas de sus usos.

Implementación

Los Stored procedures se usan a menudo para realizar consultas SQL, insertar, actualizar o eliminar registros sobre los objetos de la base de datos de una manera transparente, desde el punto de vista del cliente de la aplicación. Un Stored Procedure, permite agrupar en forma exclusiva parte de una tarea específica que se desee realizar o mejor dicho el SQL apropiado para dicha acción.

Son implementados desde la aplicación mediante *CALL procedure (...)*.

Pueden devolver un conjunto de resultados, es decir, los resultados de un comando SELECT y dichos conjuntos de resultados pueden ser procesados por otros procedimientos almacenados utilizando **cursores**.

Los Procedimientos Almacenados también pueden contener variables declaradas para el procesamiento de los datos y cursores, que le permiten a los bucles actuar sobre las múltiples filas de una tabla.

El estándar SQL proporciona *IF, WHILE, LOOP, REPEAT, CASE*, y mucho más. Los Procedimientos almacenados pueden, además, verificar condiciones, verificar el rendimiento o calcular resultados.

Dependiendo del motor de base de datos, los procedimientos almacenados pueden ser implementados en una gran variedad de lenguajes de programación, por ejemplo SQL, Java, C o C ++.

Dependiendo de la base de datos, y de la configuración, las sentencias de SQL se pueden pre-compilar para una ejecución más rápida.

Ejecución

Los Procedimientos almacenados se ejecutan directamente en el servidor de base de datos.

Usos

Un uso común es el encapsulamiento de un API para un proceso complejo o que podría requerir la ejecución de varias consultas SQL, tales como la manipulación de un gran conjunto de datos para producir un resultado resumido.

También pueden ser usados para el control de gestión de operaciones, y ejecutar procedimientos almacenados dentro de una transacción de tal manera que las transacciones sean efectivamente transparentes para ellos.

Cuando los procedimientos almacenados se aplican en la validación de datos, y el control de la integridad dentro de la estructura de la base de datos se ejecutan a través de los denominados "triggers" que se describen más adelante.

Ventajas y desventajas

- **Simplificación de la Gestión:** Los Procedimientos almacenados pueden permitir que la lógica del negocio se encuentre como una API en la base de datos, que puede simplificar la gestión de datos y reducir la necesidad de codificar la lógica en el resto de los programas cliente.
El programador de una aplicación puede así ejecutar la llamada al procedimiento almacenado conociendo los parámetros que éste requiere sin necesidad de conocer la lógica de dicho procedimiento. Esto puede reducir la probabilidad de que los datos sean corrompidos por el uso de programas clientes defectuosos o erróneos. De este modo, el motor de la base de datos puede asegurar la integridad de los datos y la coherencia, con la ayuda de procedimientos almacenados.
Sin embargo, algunos afirman que las bases de datos deben ser utilizadas para el almacenamiento de datos solamente, y que la lógica de negocio sólo debería ser aplicada en la capa de negocio de código, a través de las aplicaciones cliente que deban acceder a los datos. No obstante esto, el uso de procedimientos almacenados no se opone a la utilización de una capa de negocio.
- **Seguridad:** Es mucho mejor usar Stored procedure por seguridad. Los procedimientos almacenados facilitan algunas tareas de administración de seguridad y asignación de permisos. Por ejemplo, se puede conceder permiso a un usuario para ejecutar un determinado procedimiento almacenado, aunque el usuario no disponga de los permisos necesarios sobre los objetos afectados por las acciones individuales de dicho procedimiento.
- **Centralización de la definición:** al formar parte de la base de datos los procedimientos almacenados están en un lugar centralizado y pueden ser ejecutados por cualquier aplicación que tenga acceso a la misma. Si un determinado proceso es programado en una aplicación, es posible que no esté disponible en todos los lugares que se lo necesite. Los procedimientos almacenados están siempre disponibles.
- **Reducción del tráfico de red:** una sentencia formada por decenas, cientos o incluso miles de líneas de código SQL puede escribirse como un procedimiento almacenado en el servidor y ejecutarse simplemente mediante el nombre de dicho procedimiento, en lugar de enviar todas las líneas de código por la red desde el cliente hasta el servidor (esta reducción del tráfico de red será especialmente significativa en redes no muy veloces, como por ejemplo, algunas redes WAN).
- **Encapsulamiento:** como se dijo anteriormente, los procedimientos almacenados encapsulan gran parte de la lógica del negocio a las aplicaciones que los utilizan. Por ejemplo, una aplicación puede llamar al procedimiento almacenado sin conocer cómo funciona internamente este proceso.
- **Ejecución centralizada en el Servidor:** Esta ejecución puede verse como una ventaja o desventaja dependiendo de los recursos con los que se cuenta.
La ventaja es que cuando está en acción, en respuesta a una petición de usuario, el procedimiento almacenado corre directamente bajo el control del motor de bases de datos, generalmente en un servidor separado aumentando con ello, generalmente, la rapidez del procesamiento del requerimiento. El gestor de la base de datos tiene acceso directo a los datos necesarios y solo necesita enviar el resultado final al usuario. Esto disminuye o evita por completo los gastos de red.
Sin embargo, esto puede verse como una desventaja ya que hay que tener en cuenta que la innecesaria o excesiva declaración de procedimientos de ejecución en el gestor de bases de datos puede ser perjudicial, en general, para el rendimiento del servidor.
- **Reducción de la escalabilidad:** los procedimientos almacenados nos esclavizan al motor de base de datos. Para migrar de un gestor de base de datos con muchos procedimientos almacenados a otro, se deberá reescribir casi la totalidad de los mismos. Esto se debe, principalmente, a que los lenguajes de procedimientos almacenados de distintos fabricantes no son compatibles entre sí.

GESTION DE DATOS. AÑO LECTIVO 2008

Teniendo en cuenta las ventajas y desventajas es aconsejable no abusar de los procedimientos almacenados y utilizarlos sólo cuando su aplicación sea efectiva.

Diferencias entre Motores. Características de los procedimientos almacenados

- Sybase y SQL Server de Microsoft: Pueden entregar varias filas, pero no soportan cursores. Deben usar Transact-SQL, un lenguaje procedimental propietario, para crearlos, que se compilan en un solo paso y se guardan en el catálogo. Se llaman con el comando EXECUTE de SQL y pasando a éste el nombre del procedimiento y del servidor en el que se halla.
- ORACLE: Entregan una sola fila, pero sí soportan cursores. Deben usar PL/SQL, un lenguaje procedimental propietario. Son llamados anteponiendo al nombre del procedimiento o de la función un link a la base de datos que apunte al servidor remoto. A partir de Oracle 8i puede escribirse también el código del procedimiento en Java; además; proporciona una máquina virtual de Java muy eficiente que se ejecuta dentro del servidor de base de datos.
- DB2/UDB de IBM: Esta familia los implementa como funciones DLL (dynamic link libraries, bibliotecas de enlace dinámico) o de las bibliotecas compartidas escritas en lenguajes de programación comunes. Además, es posible escribirlos como clases Java. Se alojan en el mismo servidor que la base de datos, pero no se guardan dentro de ella. No es necesario que las aplicaciones Cliente que llaman a los procedimientos almacenados de DB2 sepan qué lenguaje se empleó para escribir el código del procedimiento.
Soporta tipos de datos definidos por el usuario, como en SQL3, denominados complementos (extenders). Se puede usar estos tipos de datos abstractos para extender la base de datos con funciones y tipos nuevos. Actualmente IBM ofrece complementos para texto, imagen, audio, video y huellas digitales. Los DBA y terceros pueden agregar sus propios complementos.
- Informix: Proporciona un lenguaje (Store Procedure Language) que no permite que las transacciones compartan procedimientos almacenados. Soporta procedimientos basados en Java: también soporta cuchillas de datos (DataBlades) que permiten extender la base de datos con tipos de objetos propios.
- MySql: Los cursores del lado del servidor están implementados a partir de *MySQL 5.0.2* a través de la función de la API de C `mysql_stmt_attr_set()`. Un cursor del lado del servidor permite a un resultado ser generado del lado del servidor, pero no transmitido al cliente excepto para aquellos registros que el cliente solicite. Por ejemplo, si un cliente ejecuta una consulta pero de ésta sólo le interesa el primer registro, los registros sobrantes no son transferidos.
Los cursores son de sólo lectura; no puede usar un cursor para actualizar registros. Los cursores no son nombrados. El proceso de la sentencia actúa como el ID del cursor.

GESTION DE DATOS. AÑO LECTIVO 2008

Sintaxis básica:

```
CREATE PROCEDURE sp_name ([parameter[,...]])  
    [characteristic ...] routine_body
```

Ejemplo en MySql

// Definición:

```
CREATE PROCEDURE miProcedimiento (IN vEdad INTEGER)  
BEGIN  
SELECT * from User where edad > vEdad;  
END
```

// Llamado al procedimiento anterior, en php;

```
$link = "ACA ESTA DEFINIDA LA CONEXIÓN A BD"  
$edad = 26;  
$query = "call miProcedimiento($edad)";  
$resultados = mysqli_query($link,$query);
```

// trabajamos con los resultados a partir de la variable \$resultados....

Ejemplo en SQL Server 2005

```
CREATE PROCEDURE VENCIMIENTO_PRODUCTOS  
@FECHA DATETIME  
AS  
SELECT stock.*,lotevencimiento.* FORM stock INNER JOIN lotevencimiento ON stock.idlote  
= lotevencimiento.idlote  
WHERE lotevencimiento.vencimiento > @FECHA  
GO
```

Progress

```
CREATE PROCEDURE [prefijo]nombre_procedimiento  
( [ {IN | OUT | INOUT} nombre_parametro tipo_dato ] )  
[ RESULT ( nombre_columna tipo_dato [... , ...] ) ]  
[ IMPORT  
    Importar_clase_java ]  
BEGIN  
    Fragmento_java  
END
```

Triggers

Un Trigger es una porción de código que se "dispara" al ocurrir un evento, es decir que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

Son usados para mejorar la administración de la Base de datos, sin necesidad de contar con el usuario para que ejecute una sentencia de SQL. Además, pueden generar valores de columnas, prevenir errores de datos, sincronizar tablas, modificar valores de una vista, etc. Permiten implementar programas basados en paradigma lógico (sistemas expertos, deducción).

Suelen ser implementados para realizar tareas relacionadas con cambios en las tablas; como auditorías, búsqueda de límites de los valores, o establecimiento de valores por omisión en columnas.

Relación-Diferencia con los Stored Procedure:

Los eventos que tienen origen en la base de datos llaman de manera implícita al trigger, en tanto que son aplicaciones cliente las que inician de manera explícita a los procedimientos almacenados.

Las implementaciones de servidor de los triggers son muy pocos estándares; más bien son específicas de cada fabricante.

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 ejecutar 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.

Existen dos tipos de triggers según la cantidad de ejecuciones que realizan:

Row Triggers (o Triggers de fila): son aquellos que se ejecutarán n-veces si se llama n-veces desde la tabla asociada al trigger

Statement Triggers (o Triggers de secuencia): son aquellos que, sin importar la cantidad de veces que se cumpla con la condición, su ejecución es única.

Pueden ser de sesión y almacenados.

Diferencias entre motores:

- Sybase: Soporta sólo un trigger por operación de inserción, actualización o eliminación.
- SQL Server de Microsoft: Soporta un máximo de 3 triggers por tabla. Cada uno de ellos puede llamar hasta 16 procedimientos almacenados.
- Ingress: Soporta varios triggers pero su ejecución no es determinística.
- Oracle: soporta hasta 12 triggers por tabla: lo hace permitiendo especificar lo siguiente por cada comando de operación: se dispara un trigger previo antes de que se ejecute el comando de SQL y otro posterior después de la ejecución. Además, Oracle permite indicar el número de veces que se dispara un trigger.
- Informix: soporta triggers previos y posteriores, y más de uno por operación; usa el número de la columna para determinar la secuencia de ejecución de los triggers.
- DB2/UDB: soporta varios triggers por cada comando de operación, en una tabla. DB2 ejecuta los triggers en el orden en que fueron creados. También soporta un tipo especial de gatillo llamado de "alerta", el cual tiene la capacidad de informar a una aplicación externa del cambio de condición de la base de datos.

GESTION DE DATOS. AÑO LECTIVO 2008

- MySql: No soportan sentencias que hacen commits o rollbacks explícitos o implícitos, ni sentencias que devuelvan un resultado. Esto incluye sentencias SELECT que no tienen una cláusula INTO y la sentencia SHOW. Una función puede procesar un resultado tanto con SELECT ... INTO como con el uso de un cursor y de la sentencia FETCH.

Los Errores de integridad referencial

La integridad referencial forzada con triggers no es estándar, es propensa a errores y difícil de implementar y mantener. Los triggers exigen esfuerzos de programación para implementar la integridad referencial; la integridad referencial mediante declaraciones no requiere tales esfuerzos. Como un servidor no tiene forma de saber que está usándose un trigger para la integridad referencial no puede hacer nada para optimizarlo.

Por ejemplo, una transacción que agrega 100 registros nuevos a la tabla de órdenes de compra y que tiene forzada la integridad referencial a la tabla de proveedores a través de un trigger, ocasionará que el trigger se ejecute 100 veces a fin de revisar el mismo valor de la clave cada vez; esa ejecución es impostergable.

En contraste, la implementación de integridad referencial mediante declaraciones en el servidor, revisaría ese valor sólo una vez para toda la operación. La integridad referencial mediante declaraciones ofrece mejor documentación y claridad utilizando instrucciones de SQL de DDL estándar basadas en el catálogo.

Cómo se usan los triggers

Para crear un trigger se usa la sentencia CREATE TRIGGER, junto a cláusulas que le indican cuando activarse y que hacer cuando se active.

Si por alguna razón, quisiéramos que el trigger no se ejecute más, debemos eliminarlo. Para eliminar un trigger, se utiliza la sentencia DROP TRIGGER indicándole el nombre del trigger a borrar;

```
DROP TRIGGER set_date;
```

Sintaxis básica:

```
CREATE TRIGGER nombre_trigger [TIME (before-after)] ON nombre_tabla FOR { INSERT,  
UPDATE, DELETE } [WITH ENCRYPTION] AS instrucción_sql
```

Ejemplo en MYSQL

1. DELIMITER //
2. **CREATE** TRIGGER set_date
3. BEFORE **INSERT** ON test
4. FOR EACH ROW **BEGIN**
5. **SET** NEW.**DATE** = **NOW**();
6. **END**;
7. //

GESTION DE DATOS. AÑO LECTIVO 2008

Ejemplo de un trigger para insertar un pedido de algún producto cuando la cantidad de éste en nuestro stock sea inferior a un valor dado.

Ejemplo en MYSQL

```
BEFORE UPDATE ON tabla_almacen
FOR ALL records
  IF :NEW.producto < 100 THEN
    INSERT INTO tabla_pedidos(producto) VALUES ('1000');
  END IF;
END;
```

Ejemplo en SQL Server 2005

```
CREATE TRIGGER ControlPedido ON Pedidos FOR INSERT AS
Begin
  INSERT INTO TablaControl (Usuario, fechaPedido) VALUES (Current_User,
  Getdate())
End;
```

Progress

```
CREATE TRIGGER bug_update_trigger
  AFTER UPDATE OF status, PRIORITY ON bug_info
  REFERENCING columna_vieja, columna_nueva
  FOR EACH columna
IMPORT
Import java.sql.*;
```

GESTION DE DATOS. AÑO LECTIVO 2008

Bibliografía:

- http://es.wikipedia.org/wiki/Stored_procedure
- http://en.wikipedia.org/wiki/Stored_procedure
- http://es.wikipedia.org/wiki/Trigger_%28base_de_datos%29
- <http://sherekan.com.ar/blog/2008/04/01/triggers-en-mysql/>
- http://www.netveloper.com/contenido2.aspx?IDC=49_0&IDP=3&P=47
- [http://msdn.microsoft.com/en-us/library/aa258254\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258254(SQL.80).aspx)
- <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/r0000931.htm>
- http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_7004.htm#sthref7885
- <http://dev.mysql.com/doc/refman/5.0/es/using-triggers.html>
- http://www.myqnet.net/articulos/sql/procedimiento_almacenados.775
- "Cliente/Servidor y Objetos " – "Guia de Supervivencia" 3era edicion. De Robert Orgali-Dan Harkey-Jeri Edwards.
- "Programación en Microsoft SQL Server 2005" –**Prentice Hall** - de Guerrero y Rojas.
- <http://www.bit-util.com/mysql/restrictions.html>
- http://www.progress.com/progress/ptw...cs/ptw_012.ppt