Gatilhos (Triggers) e Procedimentos Armazenados (Stored Procedures)

Triggers

Um trigger é um tipo especial de procedimento armazenado, que é executado sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele. Por isso temos:

- Associados a uma tabela: os triggers são definidos em uma tabela específica, que é denominada tabela de triggers;

- Chamados Automaticamente: quando há uma tentativa de inserir, atualizar ou excluir os dados em uma tabela, e um trigger tiver sido definido na tabela para essa ação específica, ele será executado automaticamente, não podendo nunca ser ignorado.

- Não podem ser chamados diretamente: ao contrário dos procedimentos armazenados do sistema, os disparadores não podem ser chamados diretamente e não passam nem aceitam parâmetros.

- É parte de uma transação: o trigger e a instrução que o aciona são tratados como uma única transação, que poderá ser revertida em qualquer ponto do procedimento, caso você queria usar “rollback”, conceitos que veremos mais a frente.

Orientações Básicas Quando Estiver Usando Trigger

As definições de TRIGGERS podem conter uma instrução “rollback transaction”, mesmo que não exista uma instrução explícita de “begin transaction”.

Se uma instrução “rollback transaction” for encontrada, então toda a transação (o trigger e a instrução que o disparou) será revertida ou desfeita. Se uma instrução no script do trigger seguir uma instrução “rollback transaction”, a instrução será executada, então, isso nos obriga a ter uma condição IF contendo uma cláusula return para impedir o processamento de outras instruções.

Não é uma boa prática utilizar “rollback transaction” dentro de seus triggers, pois isso gerará um retrabalho, afetando muito no desempenho de seu banco de dados, pois toda a consistência deverá ser feita quando uma transação falhar, lembrando que tanto a instrução quanto o trigger formam uma única transação.

O mais indicado é validar as informações fora das transações com trigger para então efetuar, evitando que a transação seja desfeita.

Para que um trigger seja disparado, o usuário o qual entrou com as instruções, deverá ter permissão de acessar tanto a entidade e consequentemente ao trigger.

Usos e aplicabilidade dos triggers

- Impor uma integridade de dados mais complexa do que uma restrição check;

- Definir mensagens de erro personalizadas;

- Manter dados desnormalizados;

- Comparar a consistência dos dados – posterior e anterior – de uma instrução update;

Os triggers são usados com enorme eficiência para impor e manter integridade referencial de baixo nível, e não para retornar resultados de consultas. A principal vantagem é que eles podem conter uma lógica de processamento complexa.

Você pode usar triggers para atualizações e exclusões em cascata através de tabelas relacionadas em um banco de dados, impor integridades mais complexas do que uma restrição check, definir mensagens de erro personalizadas, manter dados desnormalizados e fazer comparações dos momentos anteriores e posteriores a uma transação.

Quando queremos efetuar transações em cascata, por exemplo, um trigger de exclusão na tabela produto do banco de dados Northwind pode excluir os registros correspondentes em outras tabelas que possuem registros com os mesmos valores de productid excluídos para que não haja quebra na integridade, como a dito popular “pai pode não ter filhos, mas filhos sem um pai é raro”.

Você pode utilizar os triggers para impor integridade referencial da seguinte maneira:

- Executando uma ação ou atualizações e exclusões em cascata: A integridade referencial pode ser definida através do uso das restrições foreignkey e reference, com a instrução create table. Os triggers fazem bem o trabalho de checagem de violações e garantem que haja coerência de acordo com a sua regra de negócios. Se você exclui um cliente, de certo, você terá que excluir também todo o seu histórico de movimentações. Não seria boa coisa se somente uma parte desta transação acontecesse.

- Criando disparadores de vários registros: Quando mais de um registro é atualizado, inserido ou excluído, você deve implementar um trigger para manipular vários registros.

Criando Triggers

As TRIGGERS são criadas utilizando a instrução create trigger que especifica a tabela onde ela atuará, para que tipo de ação ele irá disparar suas ações seguido pela instrução de conferência para disparo da ação.

E meio a esses comandos, temos algumas restrições para o bom funcionamento. O SQL Server não permite que as instruções a seguir, sejam utilizadas na definição de uma TRIGGER:

- Alter Database;
- Create Database;
- Diskinit;
- Diskresize;
- Drop Database;
- Load Database;
- Load Log;
- Reconfigure;
- Reatore Database;
- Restorelog.

Caso você queira determinar referencias que um TRIGGER faz a objetos, execute o procedimento armazenado do sistema sp_depends, criando o seguinte comando da Figura 1:

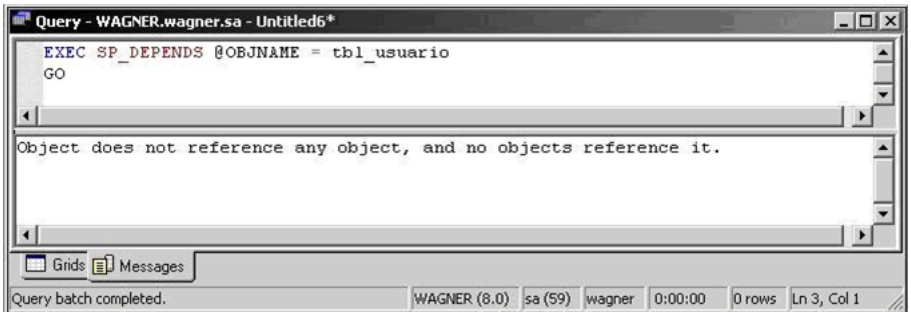

Figura 1. Mostra que o trigger criado, não referência e nem é referenciado por nenhum outro objeto

Para determinar os TRIGGERS existentes em uma específica e suas ações, execute o procedimento armazenado do sistema sp_helptrigger, criando o seguinte comando da Figura 2:

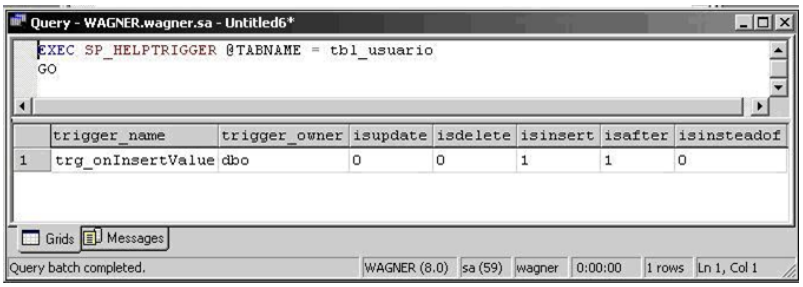

Figura 2. Mostra que temos um trigger somente para insert.

Como Funcionam os Triggers

Quando incluímos, excluímos ao alteramos algum registro em um banco de dados, são criadas tabelas temporárias que passam a conter os registros excluídos, inseridos e também o antes e depois de uma atualização.

Quando você exclui um determinado registro de uma tabela, na verdade você estará apagando a referência desse registro, que ficará, após o delete, numa tabela temporária de nome deleted. Um TRIGGER implementado com uma instrução select poderá lhe trazer todos ou um número de registro que foram excluídos.

Assim como acontece com delete, também ocorrerá com inserções em tabelas, podendo obter os dados ou o número de linhas afetadas buscando na tabela inserted. Já no update ou atualização de registros em uma tabela, temos uma variação e uma concatenação para verificar o antes e o depois.

De fato, quando executamos uma instrução update, a “engine” de qualquer banco de dados tem um trabalho semelhante, primeiro exclui os dados tupla e posteriormente faz a inserção do novo registro que ocupará aquela posição na tabela, ou seja, um delete seguido por um insert. Quando então, há uma atualização, podemos buscar o antes e o depois, pois o antes estará na tabela deleted e o depois estará na tabela inserted.

Nada nos impede de retornar dados das tabelas temporárias (inserted, deleted) de volta às tabelas de nosso banco de dados, mas atente-se, os dados manipulados são temporários assim como as tabelas e só estarão disponíveis nesta conexão. Após fechar, as tabelas e os dados não serão mais acessíveis.

Trigger Insert

De acordo com a sua vontade, um trigger por lhe enviar mensagens de erro ou sucesso, de acordo com as transações. Estas mensagens podem ser definidas em meio a um trigger utilizando condicionais IF e indicando em PRINT sua mensagem personalizada. Por exemplo, vamos criar uma tabela chamada tbl_usuario, com a qual simularemos um cadastro de usuários que você também poderá criar para fazer os seus testes. A cada inserção de novo usuário, podemos exibir uma mensagem personalizada de sucesso na inserção. Na Figura 3 vemos o código para criar a tabela.

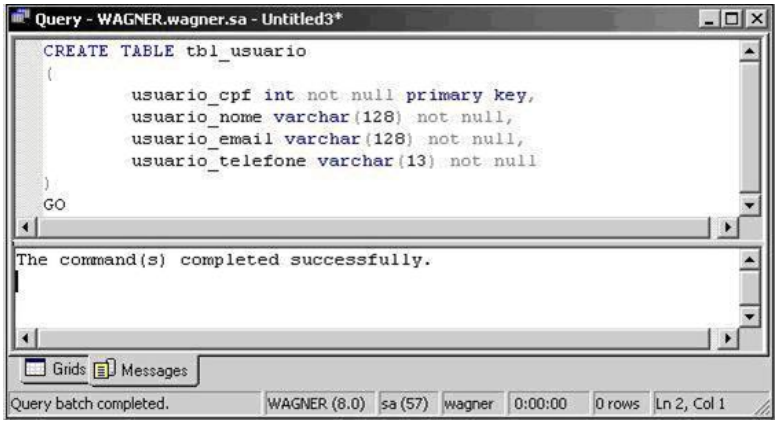

Figura 3. Criando então a tabela

Após criamos a tabela iremos criar nossa trigger personalizada que nos mostrará uma mensagem personalizada a partir de uma inserção. Para isso observe a Figura 4.

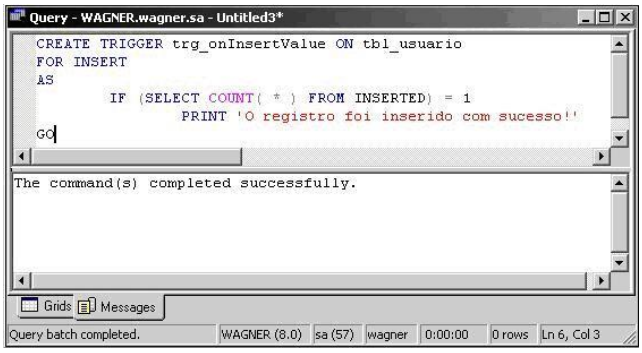

Figura 4. Criando então o trigger

Assim que começarmos a pular nossa tabela tbl_usuario, a cada registro inserido, o SGBD nos devolverá uma mensagem, caso a inserção seja realizada com sucesso, com base nos registros que são adicionados também a nossa tabela lógica/temporária inserted. Na Figura 5 temos a Mensagem disparada pelo trigger.

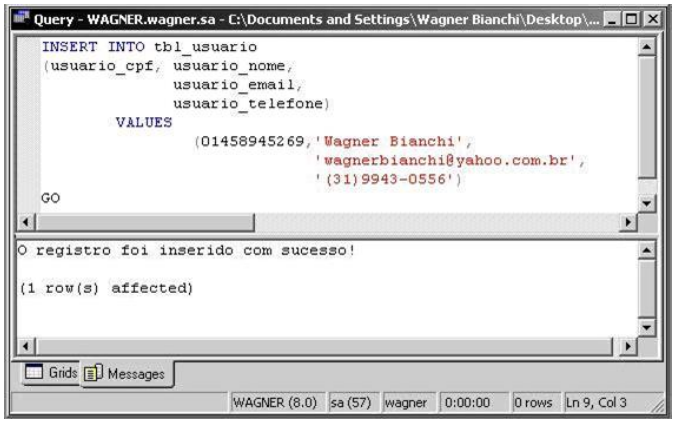

Figura 5. Recebendo a mensagem disparada pelo trigger

Existem várias outras abordagens para o uso de triggers com inserts, por exemplo, quando se faz um controle de entrada e saídas de produtos, podemos ter um trigger executando a baixa dos produtos no estoque (entrada) diante daqueles que foram solicitados num pedido (saída). O trigger pode automatizar essa rotina.

Trigger Delete

Podemos usar um trigger para monitorar certas exclusões de registros de acordo com a sua regra de negócios e também para proteger a integridade dos dados em um determinado banco de dados.

Alguns fatos devem ser considerados ao usar triggers delete:

- Quando um registro é acrescentado a tabela temporária deleted, ele deixa de existir na tabela do banco de dados. Portanto, a tabela deleted, não apresentará registros em comum com as tabelas do banco de dados;

- É alocado espaço na memória para criar a tabela deleted, que está sempre em cache;

- Um trigger para uma ação delete não é executado para a instrução truncate table porque a mesma não é registrada no log de transações.

Podemos criar um trigger que não permita a exclusão de usuários de nossa tabela tbl_usuario, da seguinte forma apresentada na Figura 6.

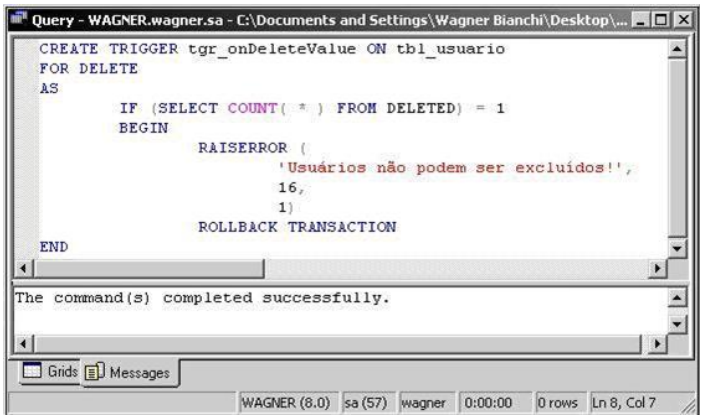

Figura 6. Criando o trigger delete

O trigger foi criado com sucesso. Na sequência tentaremos executar uma instrução para excluir o usuário que temos já em nossa tabela. A Figura 7 mostra a mensagem disparada pelo trigger.

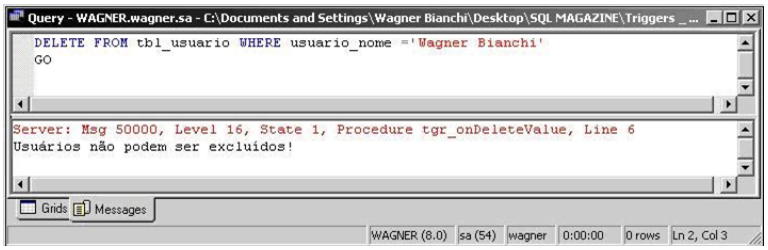

Figura 7. Veja que é apresentada também a mensagem de erro que definimos no trigger e depois a mensagem que personalizamos

Veja que realmente, triggers pode nos trazer segurança e também integridade dos dados mesmo que desnormalizados. No caso apresentado, a mensagem de erro em vermelho foi chamada pelo nosso trigger e após ele dispara também, a mensagem que personalizamos.

Assim, existem várias outras maneiras de expandirmos o exemplo.

Trigger Update

Partindo então do princípio que uma instrução update apresenta as duas etapas, as quais já citamos, com ela podemos verificar o antes e o depois, já que os dados estarão disponíveis nas tabelas temporárias deleted – momento anterior – inserted – momento atual, logo após o update.

Na Figura 8 criamos o trigger update.

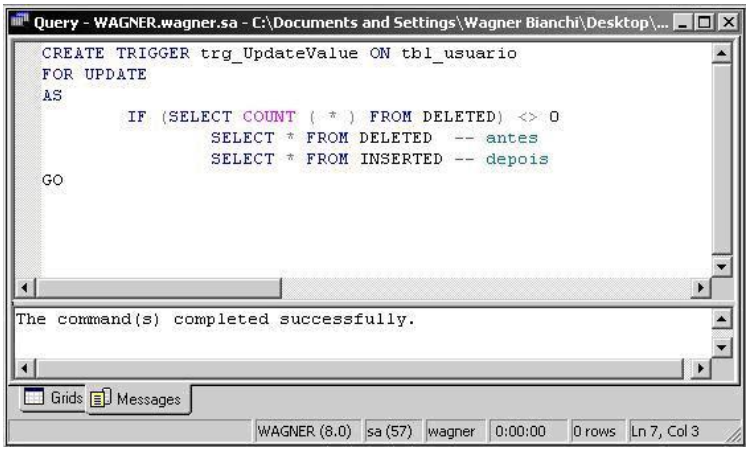

Figura 8. trigger para update

A qualquer momento daqui por diante, poderemos então contemplar, sempre que executada uma instrução para atualização de registros, o antes e o depois, o que acabamos de definir no trigger para update.

Executar a instrução para atualizar o único registro que temos então em nossa tabela. Mudemos o nome WAGNER BIANCHI para WAGNER BIANCHI Jr. Veremos então como era o nome antes e como é o nome agora!

Na Figura 9 vemos o conteúdo disparado pelo trigger.

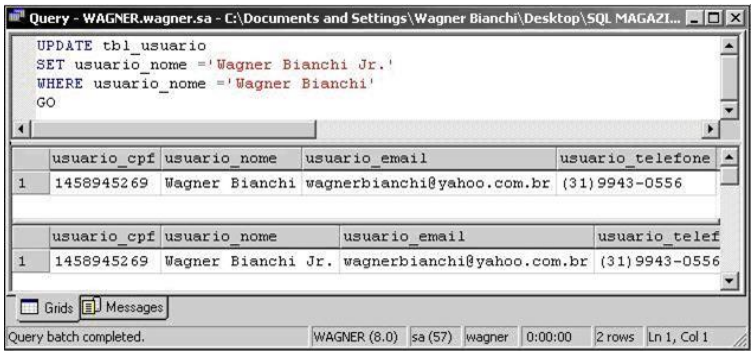

Figura 9. Visualizando o antes e o depois com a definição de trigger for update

Alterando o Conteúdo de Um Trigger

O comando aletr dá suporte para a alteração de muitos objetos criados dentro de um banco de dados. Podemos chamar alterar o conteúdo de um trigger facilmente, trocando a palavra create por alter e em seguida, executando o comando.

OBS: Para definição de triggers, podemos também usar o comando with encryption, mas, caso você não se lembre do que foi escrito no trigger, não será possível altera-lo.

Vamos alterar então o trigger para upadte, para que ele, ao invés de mostrar passado e presente, como mostra a Figura 10 Vamos fazer com ele nos mostre uma mensagem, como mostra a Figura 11.

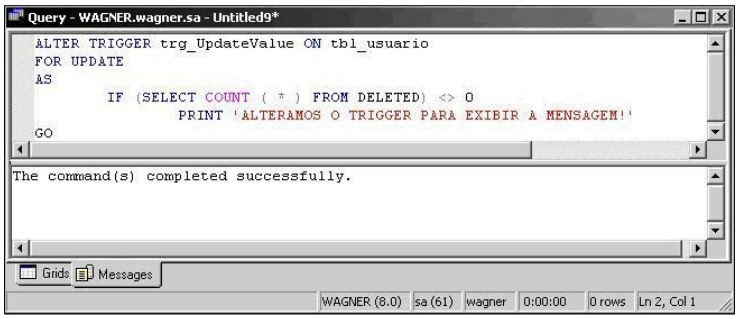

Figura 10. Alter trigger para update

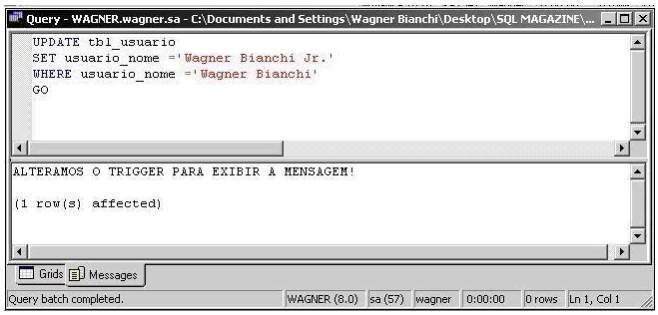

Figura 11. Exibindo a mensagem de nossa trigger alterada

Apagando Um Trigger

Caso você queira apagar um trigger do banco de dados, utilize drop trigger mais o nome do trigger que deseja apagar, como mostra a Figura 12.

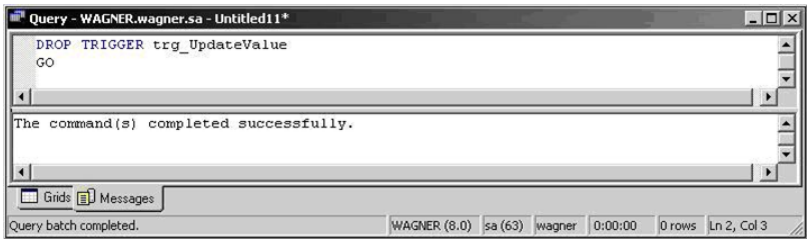

Figura 12. Apagando o trigger

Procedimentos Armazenados (Stored Procedures)

Procedimento armazenado ou Stored Procedure é uma coleção de comandos em SQL, que podem ser executadas em um Banco de dados de uma só vez, como em uma função. Os procedimentos armazenados encapsulam tarefas repetitivas, aceitam parâmetros de entrada, são capazes de utilizar os comandos como IF e ELSE, WHILE, LOOP, REPEAT e CASE, além de poderem chamar outros procedimentos armazenados e retornam um valor de status (para indicar aceitação ou falha na execução).

Existem diversos usos para procedimentos armazenados,pois, dentro do procedimento podemos utilizar diversos tipos de comandos como INSERT, UPGRADE, DELETE, MERGE, DROP, CREATE e ALTER assim fornecendo um grande leque de utilidades para procedimentos armazenados.

Um procedimento armazenado também podem ser utilizado para validação de dados e controle de acesso.

Os procedimentos são como funções que serão guardadas no servidor, que podem ou não ser executadas através de um comando “EXEC [nome da procedure] ” (em seu caso sem a necessidade de parâmetros de entrada).

Por ser executada dentro do servidor, o trafego de dados existente na rede é drasticamente reduzido, pois, as únicas coisas que serão passadas pela rede são os valores dos parâmetros de entrada e o nome do procedimento assim otimizando o tempo de execução, diminuindo o uso da CPU e diminuindo a necessidade de memória. Além criar mecanismos de segurança entre a manipulação dos dados do Banco de Dados. Exemplo: (MS-SQL Server)

```
Create procedure busca

@nomedebusca varchar (50)

as

select nome1, nome2

from nome_da_tabela

where nome = @nomedebusca
```

Características dos Procedimentos no MS-SQL Server

- Procedimentos do Sistema - Armazenadas no banco de dados Master, são identificadas com o prefixo sp_, executam tarefas administrativas, podem ser executadas em qualquer banco de dados.
- Procedimentos Locais - São criadas em bancos de dados do usuário.
- Procedimentos Temporários - Locais devem começar com #. Globais devem começar com ##.
- Procedimentos Remotos - Apenas por compatibilidade. No seu lugar se usa Queries distribuídas.
- Procedimentos Estendidos - São implementadas como DLL e executadas fora do ambiente do SQL Server. Identificadas com o prefixo xp_.

Procedimento armazenado utilizando Parâmetros

Podemos também, na stored procedured, utilizar parâmetros para que possamos ter mais utilidade.

Exemplo utilizando o Microsoft SQL Server:

```
-- Criar o procedimento

CREATE PROCEDURE Authors

AS SELECT primeiro_nome, ultimo_nome

FROM authors ORDER BY primeiro_nome ASC

-- Executar o procedimento

EXEC pAuthors

-- Deletar o procedimento

DROP PROCEDURE Authors
```

É possível criar uma stored procedured passando um ou mais parâmetros. Seguindo o exemplo acima, podemos fazer da seguinte maneira:

```
-- Criar o procedimento
```

```
CREATE PROCEDURE Authors @cidade varchar(50), @estado varchar(25)

AS SELECT primeiro_nome, ultimo_nome

FROM authors

WHERE authors.cidade = @cidade

AND authors.estado = @estado

ORDER BY primeiro_nome ASC

-- Executar o procedimento

EXEC Authors @cidade = 'Recife', @estado = 'Pernambuco';

-- Deletar o procedimento

DROP PROCEDURE Authors
```

Outros usos

Em alguns sistemas, os procedimentos armazenados podem ser usados para controlar o gerenciamento de transações; em outros, procedimentos armazenados são executados dentro de uma transação de forma que as transações sejam efetivamente transparentes para eles. Os procedimentos armazenados também podem ser chamados de um Gatilho ou de um tratador de condição. Por exemplo, um procedimento armazenado pode ser disparado por uma inserção em uma tabela específica, ou atualização de um campo específico em uma tabela, e o código dentro do procedimento armazenado seria executado. Escrever procedimentos armazenados como manipuladores de condição também permite que os administradores de banco de dados rastreiem erros no sistema com mais detalhes usando procedimentos armazenados para capturar os erros e registrar algumas informações de auditoria no banco de dados ou um recurso externo como um arquivo.

Vantagens e Desvantagens

Principais Vantagens de usar uma stored procedured:

- Melhoria na performance, já que terá uma menor quantidade de trânsito entre redes;
- Pode ser compartilhado entre as aplicações;
- Portabilidade;

Principal Desvantagem de usar uma stored procedured:

Se utilizarmos uma stored procedured podemos ficar bastante dependente da base de dados, com isso, se precisássemos mudar de base por algum motivo, iriamos ter que reescrever todas as storeds procedureds, o que ocasionaria em um grande custo de tempo. Existe ferramentas que consegue fazer uma migração como essa, entretanto nem sempre é funcional.

Comparando Procedimentos Armazenados com Funções

- Funções são subprogramas escritos para realizar uma determinada tarefa enquanto procedimentos armazenados são sub-rotinas utilizadas para evitar grande repetição de consultas.
- Normalmente funções retornam apenas um único resultado ou NULL enquanto procedimentos podem retornar uma grande quantidade de resultados o parâmetro OUT ou simplesmente não retornar nada.
- Funções precisam retornar valores utilizando a palavra chave RETURN enquanto nos procedimentos armazenados isso não é necessário.

- Um procedimento armazenado pode utilizar a palavra chave RETURN sem passar nenhum parâmetro de retorno.
- O procedimento armazenado guarda a consulta em tempo de compilação.
- O procedimento armazenado é um objeto do banco de dados e também pode ser utilizado como material para criação de outro objeto no banco de dados.