# CRIANDO VISÕES (VIEW)

As views nada mais são do que instruções SELECT já pré-definidas e armazenadas no banco. É um objeto do SQL Server utilizado para dar nome a uma query. Este conceito pode parecer simples, e é, mas pode-se resolver muitos problemas com as views.

Uma view não é um programa, portanto dentro dela você só pode escrever o comando Select. Você pode colocar dentro de uma view o comando select feito com as seguintes critérios:

* Consultas com união (Join)
* Consultas com união (Union)
* Consultas com Group By com função de totalização
* Consultas com subconsultas (Subquery)
* Consultas com Order By apenas com Top.

Uma view não pode ter um select com order by sem a clausula Top. Uma view não recebe parâmetros de entrada.

# TRIGGERS (GATILHOS)

Triggers(Gatilhos) são procedimentos que podem ser gravados em Transact SQL, Java, PL/SQL ou C, e são executados (ou disparados) implicitamente quando uma tabela é modificada, um objeto é criado ou ocorrem algumas ações de usuário ou de sistema de banco de dados.

As triggers são similares as stored procedures diferindo, apenas, na maneira como são chamadas. As Triggers são chamadas indiretamente pelos comandos INSERT, UPDATE ou DELETE executada implicitamente quando ocorre algum evento de trigger enquanto a stored procedure deve ser executado explicitamente.

A principal aplicação de uma trigger é a criação de restrições de acesso ao banco de dados, como rotinas de segurança.

Em vez de deixarmos o controle da aplicação para a própria tabela, passamos a executar por meio de triggers, ou gatilhos, esses controles, tornando muito mais seguro o manuseio de nossa base de dados.

Entre as várias utilidades de um trigger, podemos destacar:

* Criar mecanismos de validação, os quais envolvam pesquisas em mais de uma tabela;
* Inserir o conteúdo de uma coluna derivada de outras colunas;
* Atualizar outras tabelas em função de inclusão ou alteração de dados da tabela que estamos utilizando;
* Criação de logs, ou registros de inclusão e alterações de usuários.

Há outras aplicações que podemos fazer com triggers, isso depende da necessidade do desenvolvedor ou DBA.

# EXECUÇÕES DINÂMICAS O SQL SERVER

Muito bem, agora vamos ver como funciona a execução dinâmica de instruções do SQL Server.

O SQL Server arquiva o plano de execução no procedure cache e a cada vez que uma Stored Procedure e for executada, ele executar a instrução e compila informação de uma forma dinâmica, assim ele sempre terá o trabalho para rodar a Stored Procedure quando for solicitada, porem os seus processos e procedimentos ficam guardados no próprio servidor o que permite a execução automática do processo.

Mas quais são os pontos negativos da execução dinâmica?

Um dos pontos importantes a serem avaliados, é a necessidade da para execução uma ação muitas vezes, o que onera muito o desempenho do servidor. Podemos contornar o problema acima utilizando a System Stored Procedure (sp\_executesql) no lugar do EXECUTE(). Para mais informações sobre a sp\_executesql procurem a documentação oficial do SQL Server no Site Microsoft ou no famoso “Pai Google”. Pois há muito conteúdo sempre atualizado e que contém uma boa descrição de seus parâmetros e alguns exemplos de uso.

O principal inconveniente da execução dinâmica é o retorno de dados. Se desejarmos fazer alguma manipulação nos dados retornados por uma execução dinâmica, devemos fazer uso de uma tabela temporária que será preenchida com o retorno da execução. Esta tabela temporária vai armazenar o último resultado da(s) instrução(es) dinâmica(s) e deve conter a mesma quantidade e tipos de dados das colunas que o resultado.

# CRIANDO STORED PROCEDURE (PROCEDIMENTOS)

Stored Procedures: são conjuntos de instruções, armazenados em bancos de dados, que realizam operações com os demais dados do banco. São úteis em um programa por várias razões, uma delas é para se evitar a necessidade de ler os dados do banco, fazer as operações neles e depois gravar novamente. Outra é pela possibilidade que trazem de se balancear o processamento entre o servidor de banco de dados e o servidor da aplicação que acessa o banco.

Um procedimento armazenado (procedure) é uma coleção salva de instruções Transact-SQL ou uma referência a um método CLR (Common Language Runtime) Microsoft .NET Framework que pode usar e retornar parâmetros fornecidos pelo usuário. Os procedimentos podem ser criados para uso permanente ou temporário em uma sessão, procedimento temporário local, ou uso temporário em todas as sessões, procedimento temporário global.

Os procedimentos armazenados também podem ser criados para serem executados automaticamente quando uma instância do SQL Server é iniciada.

Em seguida vamos concatenar o valor do parâmetro com uma string e colocar a string completa dentro de uma variável. Por fim, iremos executar esta instrução com o comando EXECUTE() do SQL Server.

# CRIANDO FUNCTION ou USER DEFINED FUNCTION

Nas linguagens de programação as funções quebram grandes tarefas de processamento de dados em pequenas partes. Conjuntos complexos de funções podem ser divididos em pequenas unidades lógicas mais facilmente gerenciáveis. Qualquer código que deve executar uma tarefa lógica específica, quando incorporado a uma função, pode ser chamado de dentro de um programa sem a necessidade de repetir o código responsável por aquela tarefa várias vezes.

Uma **FUNÇÃO DEFINIDA PELO USUÁRIO** (UDF) é um módulo de instruções Transact-SQL que pode ou não possuir argumentos e que retorna um resultado. Por exemplo, uma função pode informar o valor de uma parcela mensal a partir de uma taxa de juros informada, calcular a distância entre dois pontos, selecionar uma lista de empregados que pertencem a um departamento, etc.

Uma UDF é muito parecida com uma stored procedure, ou seja, possui instruções que controlam o fluxo do programa e a manipulação dos dados.

Uma UDF difere de uma stored procedure nos seguintes aspectos:

* Não permite qualquer alteração no estado global da sessão, por exemplo, como a mudança do banco de dados.
* Não permite a utilização de parâmetros com a clásula OUTPUT.
* Não pode ser usada com a opção “FOR XML” para obtenção do resultado. Portanto, a opção “FOR XML” pode ser utilizada apenas em stored procedures.
* Erros do Transact-SQL que causam o cancelamento da instrução, e então continuam na próxima instrução de um módulo (como em triggers e stored procedures) são tratados de modo diferente dentro de uma função. Em uma função, os erros encontrados em instruções Transact-SQL interrompem seu processamento.

Entretanto, se você necessita de procedimentos que retornam resultados, uma UDF é frequentemente a melhor escolha para uma aplicação SQL, pelo simples fato de poder ser chamada a partir de uma instrução SQL.

# function UDF:

Antes de poder definir uma UDF, deve-se determinar suas características, tais como, seu nome, sua vinculação ou não aos objetos referenciados, o número e tipos de dados dos parâmetros que ela terá, e finalmente, os tipos de valores retornados. Você então executa a instrução CREATE FUNCTION para criar a UDF. Caso após definir a função, você venha a descobrir que qualquer uma das características não são apropriadas, você poderá usar a instrução ALTER FUNCTION para alterar as informações utilizadas em sua definição.

Você não poderá fazer uso da instrução ALTER FUNCTION para modificar o tipo de uma função, ou seja, uma função do tipo scalar valued para uma função do tipo table valued ou vice-versa. Similarmente, você não poderá utilizar a instrução ALTER FUNCTION para modificar uma função do tipo inline para uma função multi-statement ou vice-versa. Entretanto, você poderá remover funções UDF com a instrução DROP FUNCTION.

É necessário ter permissão sobre a instrução CREATE FUNCTION para criar, alerar ou excluir UDFs. Outros usuários, além do criador da UDF, devem ter as permissões apropriadas antes de poder utilizá-la em instruções Transact-SQL. Para criar ou alterar tabelas com CHECK constraint, cláusula DEFAULT ou uma computed column, que façam referências a UDFs, o usuário deve ter a permissão de REFERENCES sobre a função.

As instruções válidas em uma função incluem:

* A instrução DECLARE poderá ser utilizada para definir as variáveis e cursores que serão locais para a função.
* Operações com cursores locais que são declarados, abertos, fechados e liberados dentro da função. A instrução FETCH que retorna dados para o cliente não é permitida.
* São permitidas as instruções de controle de fluxo (tais como IF..ELSE, WHILE, RETURN, GOTO, BREAK e CONTINUE).

As seguintes regras se aplicam quando uma UDF faz chamadas a uma extended stored procedure:

* A extended stored procedure não pode retornar dados para o cliente. Se a stored procedure retorna dados, a execução da função chamadora falhará.
* A extended stored procedure é executada no contexto de segurança da conta de usuário do Microsoft Windows® sob a qual o SQL Server está rodando. O criador da stored procedure deve considerar isso quando atribuir direitos de EXECUTE para os usuários.

# Tipos de Function:

O SQL Server 2000 implementa três tipos de UDFs assim conhecidas:

* Scalar valued functions
* In-line table valued functions
* Multi-statement table valued functions

## Funções Scalar valued:

Uma função definida pelo usuário do tipo scalar valued retorna um único valor em resposta a cada uma das chamadas à função. Uma função é do tipo scalar valued se a cláusula de retorno especifica um tipo de dado scalar do SQL Server. Funções do tipo Scalar valued podem ser definidas com a utilização de múltiplas instruções Transact-SQL.

A cláusula RETURNS desse método, especifica que um decimal scalar\_data\_type será retornado pela função. A cláusula RETURNS pode ser qualquer um dos tipos de dados conhecidos como scalar\_data\_types do SQL Server exceto timestamp, text, ntext, or image.

Para as funções do tipo scalar valued, utiliza-se a instrução RETURN com um argumento. O valor desse argumento é retornado como o resultado da função. O tipo de dado do argumento passado com a instrução RETURN deve poder ser implicitamente convertido para o tipo de dado do valor de retorno da função.

## Funções Table Valued:

Para funções do tipo inline table valued, a cláudula RETURNS é seguida da palavra TABLE sem uma lista de colunas. As funções do tipo inline table valued retornam valores apresentados como se extraídos de tabelas e são definidas com uma única instrução SELECT em sua construção. As colunas, incluindo os tipos de dados retornados pela função, são derivados da lista da instrução SELECT que define a função.

Funções Multi-Statement Table Valued:

Se a cláusula RETURNS especificar uma tabela com suas colunas e tipos de dados, a função será do tipo multi-statement table valued.

A cláusula RETURNS desse método define uma variável local de retorno do tipo table, e também define a estrutura da tabela com a definição das colunas.

De uma forma simplifica a diferença entre as duas é que uma (Scalar) retorna um valor variável e a outra (Table) um tabela ou consulta inteira.

# Trabalhos (Jobs)

O Trabalho ou Job, como é conhecido, é uma forma de automatizar quaisquer tipo de execução dentro do SQL server. Pois permite o agendamento de execuções de quaisquer tipos de processos em quaisquer horários, auxiliando tanto na manutenção da estrutura com a possibilidade de execuções fora do horário de funcionamento da empresa, quanto com a realização de tarefas que devem ser executadas diversas vezes iguais ou em uma determinada data em especial sem precisar da intervenção de um executor ou de quaisquer intervenções manuais.

Antes de iniciarmos

Devemos identificar se o usuários que estamos utilizando é membro de uma das funções de banco de dados fixas do SQL Server Agent ou da função de servidor fixa sysadmin . Um trabalho só pode ser editado por seu proprietário ou por membros da função sysadmin, bem como executar qualquer trabalho, independentemente de seu proprietário.

Também deveremos ter certeza que existe instalado no nosso servidor um banco de dados chamado MSDB, que é o banco utilizado pelo SQL Server para controlar os JOBs.