Uma Stored Procedure (SP) tem por objetivo utilizar o mesmo conceito que já conhecemos na programação como procedimento para criar e armazenar um conjunto de comandos dentro de um banco de dados. Esse procedimento pode ser executados a qualquer momento pelo Sistema Gerenciador de Banco de Dados, pelo usuário ou por uma aplicação qualquer.

Uma Stored Procedure pode fazer uso de várias estruturas de criação de programas inerentes a uma linguagem de programação, portanto podemos fazer uso de comandos de controle, comandos condicionais, comandos de comparação, comandos de repetição e até de variáveis para criar um procedimento no banco de dados.

Quando fazemos uso de Stored Procedures, trazemos os comandos SQL para o banco de dados evitando manutenções desnecessárias no aplicativo sempre que for necessário fazer alterações na forma de manipular as informações armazenadas no banco de Dados.

VANTAGENS DO USO DE SPs

Modularidade: Com o uso de stored procedures passamos a ter as ações realizadas no banco de dados separadas das outras partes do software, bastando alterar somente os seus comandos ou a sua estrutura para que as alterações sejam refletidas para as aplicações que usam o banco;

Diminuição de trafego de rede: usando a modularização e a passagem de parâmetros para a execução de uma SP no servidor, as ações ficam concentradas nele não havendo "conversa" intermediária entre a aplicação e o SGBD. Sendo assim, apenas no final da execução da SP os resultados de uma transação são retornados para a aplicação ou para quem solicitou a sua execução evitando o tráfego elevado de dados pela rede;

Rapidez na execução: Como as SPs são armazenadas no servidor elas ficam aguardando o momento de sua execução de forma pré-compilada. Após a sua primeira execução, elas ficam armazenas em memória cache o que torna a execução mais rápida assim como acontece com as páginas de internet.

Segurança de dados: omitindo do usuário final ou do programador a estrutura do banco, evitamos que inconsistências sejam geradas no banco e mantemos o controle dos dados com o SGBD.

CRIANDO STORED PROCEDURES

Antes de começar a criar nossas SPs, é importante que tenhamos em mente algumas coisas:

Dentro das nossas SPs, podemos fazer referencia as tabelas dos nossos bancos assim como a tabelas temporárias, a VIEWS e a TRIGGERS do nosso banco.

Tabelas temporárias criadas dentro de uma SP só existirão ao longo da execução desta SP.

Não podemos criar uma SP dentro de uma SP.

Somente os donos do banco ou o administrador do sistema tem o privilégio de criar SPs em um banco.

Não podemos criar VIEWS ou TRIGGERS dentro de uma SP.

SINTAXE BÁSICA

```
CREATE PROCEDURE <nome_do_procedimento>
     @param 1 tipo_de_dado,
     @param 2 tipo_de_dado,
     @param n tipo_de_dado

AS
   Comando01
   Comando02
   Comando03
   ComandoN
GO
```

Podemos ainda criar uma SP que retorna valores apenas colocando a palavra OUTPUT após a declaração de um de seus parâmetros (Normalmente o último).

```
CREATE PROCEDURE <nome_do_procedimento>
     @param 1 tipo_de_dado,
     @param 2 tipo_de_dado,
     @param n tipo_de_dado OUTPUT

AS
   Comando01
   Comando02
   Comando03
   ComandoN
GO
```

EXEMPLO

Vamos criar dentro do seu banco de dados uma tabela de nome exsp_usuario com o seguinte comando:

```
CREATE TABLE exsp_usuario(
               email varchar(100) not null,
               nome  varchar(30) not null,
               data_cadastro datetime null DEFAULT getdate()

               constraint pk_exspusuario primary key (email)
)
```

Vamos então criar uma SP para fazer inserção de dados nesta tabela. Para efetuar essa operação, teremos que passar para a nossa SP todos os dados que queremos inserir. O processo se dará de forma semelhante ao que conhecemos como chamada de função com passagem de parâmetros em lingagem de programação portanto devemos programar a nossa SP para receber tais parâmetros e inseri-los na tabela usando o comando INSERT que já nos é conhecido. Vejamos:

```
CREATE PROCEDURE sp_InserirUsuario
       @email varchar(100),
        @nome  varchar(30)
```

```
AS
   INSERT INTO exsp_usuario(email, nome, data_cadastro)
           VALUES (@email, @nome, getdate())
GO
```

Após executar o comando acima a SP estará criada!

Vamos agora executar a nossa SP da seguinte forma:

```
exec sp_InserirUsuario
   @email='andreia@cotuca.unicamp.br',
        @nome='Andreia Souza'
```

Execute o comando acima e os dados serão inseridos!

Comando condicional: Podemos colocar condições para a execução dos nossos comandos. Por exemplo, se quisermos inserir os dados apenas se o valor do e-mail não for vazio podemos alterar a nossa SP da seguinte forma:

```
ALTER PROCEDURE sp_InserirUsuario
   @email varchar(100),
        @nome  varchar(30)

AS
  IF (@email<>'')
       BEGIN
          INSERT INTO exsp_usuario(email, nome, data_cadastro)
           VALUES (@email, @nome, getdate())
          PRINT 'Dados inseridos!'
       END
   ELSE
       PRINT 'Digite um e-mail'
GO
```

Agora tente executar a SP passando um e-mail em branco da seguinte forma:

```
exec sp_InserirUsuario
   @email='',
        @nome='Andreia Souza'
```

Vejamos um exemplo de Stored Procedure com retorno de valor:

```
CREATE PROCEDURE multiplica @val1 int,
                            @val2 int = 10,
                            @val3 int output
 AS
     SET @val3 = @val1 * @val2

-- Declarando uma variável para ser o resultado
DECLARE  @res int
```

```
-- Executando a SP
exec multiplica 10,10,@res output

--Exibindo o resultado
SELECT @res
```

Vejamos alguns comandos que podemos usar dentro de uma SP!

Declaração de Variáveis

Para declarar uma variável, usamos a palavra DECLARE seguida do nome da variável começando por @, seguida por seu tipo.

```
DECLARE @nome char(20), @endereco char(200)
DECLARE @idade int
```

Atribuindo valores para variáveis

Para atribuir valores para uma variável usamos a instrução SET seguida pelo nome da variável a ser atribuída seguida pelo operador de atribuição = e depois o valor a ser atribuído.

```
SET @nome='Andréia'
```

Podemos ainda atribuir a uma variável o valor de retorno de um comando SQL da seguinte forma:

```
DECLARE @qtos int
SET @qtos = (SELECT COUNT(*) FROM CLIENTES)
```

Controle de Fluxo

Para o controle do fluxo de execução de uma Strored Procedure, utilizamos basicamente os mesmos recursos que temos dentro de uma linguagem de programação.

Comando condicional

Como comando condicional, a linguagem de programação considerada em uma Stored Procedure comporta a estrutura IF..ELSE e também a estrutura CASE..WHEN..THEN..END.

Vejamos o IF

```
IF teste_lógico
   Comando_Se_Teste_Verdadeiro
ELSE
   Comando_Se_Teste_Falso
```

Caso seja necessário executar mais do que um comando, devemos usar o comando de bloco como nas linguagens de programação que sabemos. Para uma SP, o comando de bloco se inicia pela palavra BEGIN e termina pela palavra END.

```
IF teste_lógico
   BEGIN
            PRIMEIRO_Comando_Se_Teste_Verdadeiro
            SEGUNDO_Comando_Se_Teste_Verdadeiro
            TERCEIRO_Comando_Se_Teste_Verdadeiro
   END
ELSE
   Comando_Se_Teste_Falso
```

Já comando CASE é utilizado para conferir uma lista de condições e, então, retornar uma entre várias expressões de resultado possíveis. CASE, que é usado em conjunto com o comando SELECT, é útil quando a intenção é a de evitar que sejam criados comandos IF aninhados. Ele é considerado como uma alternativa ao IF.

Vejamos o CASE..WHEN..THEN..END.

```
CASE Variavel_a_ser_avaliada
   WHEN valor_1 THEN comando_para_valor_1
   WHEN valor_2 THEN comando_para_valor_2
   WHEN valor_3 THEN comando_para_valor_3
   ELSE comando_caso_nenhum_dos_acima
END
```

Comando de Repetição

O comando de repetição que normalmente usamos para o controle dentro de uma Stored Procedure é a estrutura WHILE .. BREAK .. CONTINUE.

```
WHILE Teste_Lógico
BEGIN
   Comandos
END
```

Caso queiramos interromper a execução de um WHILE podemos executar o comando BREAK e caso queiramos desviar e execução dos comandos para o teste do WHILE executamos o comando CONTINUE.

Comando RETURN

O comando RETURN termina a execução de qualquer procedimento. Caso coloquemos algum valor após a palavra return, esse valor será retornado a quem solicitou a chamada do procedimento.