# Definição do SQL Server

O SQL Server é um SGDB desenvolvido pela Microsoft, é considerado seguro em suas várias medidas de proteção, como boa administração de usuários e permissões. Apesar de utilizar comandos base em SQL, o SQL Server utiliza o T-SQL. A linguagem Transact-SQL é uma extensão ao padrão SQL-92, sendo a linguagem utilizada por desenvolvedores na construção de aplicações que manipulam dados mantidos no SQL Server.

Mais informações sobre o T-SQL em: <https://www.devmedia.com.br/trabalhando-com-a-linguagem-t-sql/38126>

Para manipular os dados do SQL Server é utilizado um ambiente chamado SSMS (SQL Server Microsoft Studio). Através dele todas as operações administrativas podem ser feitas, ou através do SQLCMD, onde comandos SQL poderão ser aceitos na linha de comando.

Um exemplo para o uso no CMD:

C:\Users\eduar>sqlcmd -U SA -S DESKTOP-J741TPG\SQLSERVER

Senha:

1> USE db\_teste

2> SELECT \* FROM tbl\_funcionarios

3> GO

Chama-se primeiro o comando ‘sqlcmd’, depois informa-se o usuário por ‘-U SA’, onde no caso está sendo utilizado o AS, informa-se o nome do servidor por ‘-S DESKTOP-J741TPG\SQLSERVER’.

A documentação oficial do SQL Server pode ser encontrada em: <https://docs.microsoft.com/pt-br/sql/sql-server/index?view=sql-server-ver15>

## A Arquitetura de Banco de Dados do SQL Server

**Um banco de dados no SQL Server é composto de uma coleção de tabelas que armazena um conjunto específico de dados estruturados**. Uma tabela contém uma coleção de linhas, também chamadas de registros ou tuplas, e colunas, também chamadas de atributos. Cada coluna da tabela é projetada para armazenar um determinado tipo de informação, por exemplo, datas, nomes, valores em dinheiro e números.

Um computador pode ter uma ou mais de uma instância do SQL Server instalada. **Cada instância do SQL Server pode conter um ou mais bancos de dados. Dentro de um banco de dados, há um ou vários grupos de propriedade de objeto denominados esquemas. Em cada esquema, há objetos de banco de dados como tabelas, exibições e procedimentos armazenados.** Alguns objetos, como certificados e chaves assimétricas, estão contidos no banco de dados, mas não estão contidos em um esquema.

Quando as pessoas obtiverem acesso a uma instância do SQL Server, elas serão identificadas como um logon. **Quando as pessoas obtiverem acesso a um banco de dados, elas serão identificadas como um usuário de banco de dados. Um usuário de banco de dados pode ser baseado em um logon.** Se forem habilitados bancos de dados independentes, um usuário de banco de dados não baseado em logon poderá ser criado.

**Um usuário que tem acesso a um banco de dados pode receber permissão para acessar os objetos no banco de dados.** Embora possam ser concedidas permissões a usuários individuais, é recomendável criar funções de banco de dados, adicionando os usuários de banco de dados às funções, e conceder permissão de acesso às funções. A concessão de permissões a funções, e não a usuários, faz com que seja mais fácil manter as permissões consistentes e compreensíveis à medida que o número de usuários aumenta e muda continuamente.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/databases>

## Bancos de Dados do Sistema

SQL Server inclui os seguintes bancos de dados do sistema:

| **Banco de dados do sistema** | **DESCRIÇÃO** |
| --- | --- |
| [Banco de dados mestre](https://docs.microsoft.com/pt-br/sql/relational-databases/databases/master-database?view=sql-server-ver15) (master) | Registra **todas as informações de nível de sistema** por uma instância do SQL Server. |
| [Banco de dados msdb](https://docs.microsoft.com/pt-br/sql/relational-databases/databases/msdb-database?view=sql-server-ver15) (msdb) | É usado pelo SQL Server Agent para **programar alertas e trabalhos**. |
| [Modelo de banco de dados](https://docs.microsoft.com/pt-br/sql/relational-databases/databases/model-database?view=sql-server-ver15) (model) | É usado como **modelo de todos os bancos de dados** criados na instância do SQL Server. As modificações feitas no banco de dados modelo, como tamanho, ordenação, modelo de recuperação, e outras opções de bancos de dados, são aplicadas a qualquer banco de dados criados em seguida. |
| [Banco de dados de recursos](https://docs.microsoft.com/pt-br/sql/relational-databases/databases/resource-database?view=sql-server-ver15) | É um banco de dados do tipo **somente leitura que contém objetos de sistema incluídos no SQL Server**. Os objetos de sistema são fisicamente persistentes no banco de dados Recurso , mas aparecem logicamente no esquema sys de todo banco de dados. |
| [Banco de dados tempdb](https://docs.microsoft.com/pt-br/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15) (tempdb) | É um workspace para reter objetos temporários ou conjuntos de resultados intermediários. |

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/system-databases>

### Banco de dados Mestre

**O banco de dados master registra todas as informações no nível de sistema para um sistema do SQL Server** . Isto inclui metadados de ampla instância como **contas de logon, pontos de extremidade, servidores vinculados e parâmetros de configuração de sistema**. No SQL Server, os objetos de sistema não são mais armazenados no banco de dados mestre; em vez disso, eles são armazenados no Banco de dados de recurso. Além disso, mestre é o banco de dados que **registra a existência de todos os outros bancos de dados e o local desses arquivos de bancos de dados, e registra as informações de inicialização para o SQL Server**. Portanto, o SQL Server não poderá iniciar se o banco de dados mestre não estiver disponível.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/master-database>

### Banco de Dados Modelo

O banco de dados modelo **é usado como modelo para todos os bancos de dados criados em uma instância do SQL Server**. Como tempdb é criado toda vez que SQL Server é iniciado, o banco de dados modelo deve sempre existir em um sistema SQL Server. Todo o conteúdo do banco de dados modelo, incluindo as opções, é copiado para o novo banco de dados. Algumas configurações do modelo também são usadas para criar um novo tempdb durante a inicialização, de modo que um banco de dados modelo sempre deve existir em um sistema SQL Server.

Bancos de dados de usuários recém-criados usam o mesmo modelo de recuperação do banco de dados modelo. O padrão é configurável pelo usuário.

**Quando uma instrução CREATE DATABASE é emitida, a primeira parte do banco de dados é criada por meio de cópia do conteúdo do banco de dados modelo.** O restante do novo banco de dados é então preenchido com páginas vazias.

**Se o banco de dados modelo for modificado, todos os bancos de dados criados posteriormente herdarão as mudanças.** Por exemplo, você poderia definir permissões ou opções de banco de dados, ou adicionar objetos, como tabelas, funções ou procedimentos armazenados. Propriedades de arquivo do banco de dados modelo são uma exceção e são ignoradas, exceto o tamanho inicial do arquivo de dados. O tamanho inicial padrão do arquivo de dados e do arquivo de log do banco de dados modelo é de 8 MB.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/model-database>

### Banco de Dados MSDB

**O banco de dados msdb é usado pelo SQL Server Agent para agendar alertas e trabalhos e por outros recursos, como o SQL Server Management Studio, o Agente de Serviço e o Database Mail.**

Por exemplo, **o SQL Server mantém automaticamente um histórico de backup e restauração online completo nas tabelas no msdb.** Estas informações incluem o nome da parte que executou o backup, a hora do backup, e os dispositivos ou arquivos onde o backup é armazenado. O SQL Server Management Studio usa essas informações para propor um plano de restauração de um banco de dados e aplicar qualquer backup de log de transações. **Os eventos de backup de todos os bancos de dados são registrados, mesmo que tenham sido criados com aplicativos personalizados ou ferramentas de terceiros.** Por exemplo, se você usar um aplicativo do Microsoft Visual Basic que chama objetos SMO (SQL Server Management Objects) para executar operações de backup, o evento será registrado nas tabelas do sistema do msdb, no log do aplicativo do Microsoft Windows e no log de erros do SQL Server. Para ajudar a proteger as informações armazenadas no msdb, recomendamos que você considere a colocação do log de transações msdb no repositório tolerante a falhas.

Por padrão, msdb usa o modelo de recuperação simples. Se você usar as tabelas de histórico de backup e restauração , será recomendável utilizar o modelo de recuperação completa para msdb. Para obter mais informações, veja Modelos de recuperação (SQL Server). Observe que, quando o SQL Server é instalado ou atualizado e sempre que Setup.exe é usado para recriar bancos de dados do sistema, o modelo de recuperação do msdb será definido automaticamente como simples.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/msdb-database>

### Banco de Dados Recursos

**O banco de dados Recurso é um banco de dados somente leitura que contém todos os objetos de sistema inclusos no SQL Server.** SQL Server objetos do sistema como sys.objects, são fisicamente persistentes no banco de dados Resource, mas aparecem logicamente no esquema sys de cada banco de dados. O banco de dados Recurso não contém dados de usuário ou metadados de usuário.

O banco de dados Recurso torna a atualização para uma nova versão do SQL Server um procedimento mais fácil e mais rápido. Em versões anteriores do SQL Server, a atualização exigia remover e criar objetos do sistema. **Como o arquivo de banco de dados Recurso contém todos os objetos de sistema, agora, uma atualização é obtida simplesmente pela cópia do único arquivo de banco de dados Recurso no servidor local.**

**SQL Server não pode fazer backup do banco de dados Resource.** Você pode executar seu próprio backup baseado em arquivo ou baseado em disco ao tratar o arquivo mssqlsystemresource.mdf como se ele fosse um arquivo binário (.EXE), em vez de um arquivo de banco de dados, mas você não pode usar o SQL Server para restaurar seus backups. Só é possível restaurar uma cópia de backup do mssqlsystemresource.mdf manualmente, e você deve ter cuidado para não sobrescrever o atual banco de dados Recurso com uma versão desatualizada ou potencialmente insegura.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/resource-database>

### Banco de Dados TempDB

O banco de dados do sistema TempDB é um recurso global disponível para todos os usuários conectados à instância do SQL Server ou ao Banco de Dados SQL. O Tempdb é usado para manter:

Objetos de usuário temporários criados explicitamente como: índices e tabelas temporárias globais ou locais, procedimentos armazenados temporários, variáveis de tabela, Tabelas retornadas em funções com valor de tabela ou cursores.

Objetos internos criados pelo mecanismo de banco de dados. Eles incluem:

Tabelas de trabalho para armazenar resultados intermediários para spools, cursores, classificações e armazenamento temporário LOB (objeto grande).

Arquivos de trabalho para operações de junção de hash ou de agregação de hash.

Resultados intermediários de classificação para operações como criar ou recriar índices (se SORT\_IN\_TEMPDB for especificado) ou determinadas consultas GROUP BY, ORDER BY ou UNION.

Repositórios de versão, que são uma coleção de páginas de dados que contém linhas de dados necessárias para dar suporte aos recursos que usam o controle de versão de linha. Existem dois armazenamentos de versão: um repositório de versão comum e um armazenamento de versão de criação de índice online. Os armazenamentos de versão contêm:

Versões de linha geradas por transações de modificação de dados em um banco de dados que usa a leitura de confirmados usando transações de isolação de controle de versão de linha ou isolação de instantâneo.

As versões de linhas geradas por meio de transações de modificação de dados para recursos como: operações de índice on-line, vários conjuntos de resultados ativos (MARS) e gatilhos AFTER.

As operações no TempDB são registradas minimamente em log para que as transações possam ser revertidas. TempDB é recriado a cada vez que SQL Server é iniciado, de modo que o sistema sempre começa com uma cópia limpa do banco de dados. As tabelas temporárias e procedimentos armazenados são descartados automaticamente ou desconectados e nenhuma conexão fica ativa quando o sistema é desligado. Portanto, nunca há nada em TempDB a ser gravado de uma sessão de SQL Server para outra. As operações de backup e restauração não são permitidas em TempDB.

Qualquer usuário pode criar objetos temporários no TempDB. Os usuários podem acessar somente seus próprios objetos, a menos que recebam permissões adicionais. É possível revogar a permissão de conexão ao TempDB para impedir um usuário de usar o TempDB, mas isso não é recomendado porque algumas operações de rotina exigem o uso do TempDB.

Qualquer usuário pode criar objetos temporários no TempDB. Os usuários podem acessar somente seus próprios objetos, a menos que recebam permissões adicionais. É possível revogar a permissão de conexão ao TempDB para impedir um usuário de usar o TempDB, mas isso não é recomendado porque algumas operações de rotina exigem o uso do TempDB.

Leia mais sobre a otimização do espaço do TempDB na fonte descrita.

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/tempdb-database>

Em caso de necessidade de recriar os bancos de dados do sistema, por corrupção nos bancos, ou para modificar a ordenação em nível de servidor padrão, siga as instruções descritas nesse documento:

<https://docs.microsoft.com/pt-br/sql/relational-databases/databases/rebuild-system-databases>

## Instalação, primeira conexão e recomendações

Tudo que é necessário para a instalação pode ser encontrado no site oficial. São somente dois itens, sendo o SQL Server e o SSMS. Recomenda-se baixar primeiro o SQL Server e, após a configuração, ele mesmo oferecerá a opção para baixar o SSMS.

Um ponto de atenção: no momento de configurar o SQL Server, optar pela configuração PERSONALIZADA. O motivo disso é a escolha do nome da instância do servidor e configuração da senha em um usuário MISTO, ou seja, que utilizará o Windows e o SQL Server. Essa senha escolhida, inicialmente será do usuário SA.

O usuário SA (Sys Admin) controla e possui total permissão em relação ao sistema, está a nível de administrador. Por esse motivo, não é recomendado utiliza-lo para as operações nas instâncias, mas sim criar um outro usuário para isso, prevenindo a segurança.

Após a instalação do SQL Server vem a instalação do SSMS, e logo quando abrir solicitará uma conexão para manipulação de dados, devendo informar o nome da instância configurada, o usuário (SA) e a senha gravada.

# Conceitos do T-SQL

O T-SQL possui um conjunto de diferenças em relação ao SQL comum, aqui são algumas delas e sua importância no desenvolvimento do banco. O livro referência desse capítulo é o The Bible of SQL Server 2012, capítulo 16, página 389.

## Batch

Uma query é uma operação no banco de dados, principalmente focado na busca (comandos DML). Um batch é uma coleção de operações SQL ou T-SQL. Cada batch de operações é lido como uma unidade de operações, e pra isso possui uma palavra-chave separada: GO.

O uso da palavra-chave GO limita um batch. Alguns comandos como CREATE PROCEDURE obrigatoriamente precisam ser limitados em um batch. Uma vantagem na sua utilização em códigos longos é a identificação de erros.

É uma boa prática que seja especificado o database usado no batch com o comando USE.

## Formatação

A formatação do T-SQL diverge em alguns detalhes com o SQL. Por exemplo, os ponto e vírgula no final das operações são opcionais, salvo em comandos como CASE, IF, em que não deve ser utilizado.

Os comentários são aceitos com dois hífens ( -- ) e um comentário longo é feito com barras e asteriscos ( /\* comentário longo de várias linhas \*/ ).

## Variáveis

Variáveis podem ser criadas no SQL Server com o comando DECLARE, seguido do nome da variável e seu tipo, como o exemplo:

DECLARE @NomeDaVariavel INT

O valor e duração de uma variável declarada estão limitados ao batch pertencente, fora dele não há significado algum.

As variáveis podem ser incrementadas ou decrementadas, como em códigos de lógica usuais.

Exemplo:

DECLARE @x INT = 1

SET @x += 5

SET @x -=3

Os comandos SET e SELECT podem setar valores em variáveis. Porém, o SET faz isso em uma variável por vez, enquanto o SELECT pode fazer em várias, e também o SELECT pode fazer atribuição com retorno de dados, ao contrário do SET. Esse artigo explica melhor o uso dos dois: <https://pt.stackoverflow.com/questions/241884/quando-usar-set-e-select>

As variáveis são dinâmicas e podem ser usadas com tranquilidade em querys SQL. Exemplo:

USE AdventureWorks2012;

DECLARE @ProductID int = 999;

SELECT Name FROM Production.Product WHERE ProductID = @ProductID;

## Fluxo Procedural

O T-SQL é como uma linguagem de programação, então permite vários padrões já conhecidos no mundo da programação, à sua maneira.

### Condicionais

O condicional IF, por exemplo, só espera um comando de comparação, se true realiza a próxima ação, se falso, a ação abaixo dessa. Exemplo:

IF 1 = 0

PRINT 'Line One';

PRINT 'Line Two';

Result: Line Two

Se usado dessa forma, o IF seguirá somente 1 linha. Para utilizar múltiplos comandos em caso verdadeiro ou falso, utilize BEGIN e END para cercar os múltiplos comandos.

Junto ao IF pode vir a função EXISTS( --comando-- ) que executa uma ação caso haja o retorno de ao menos uma linha. Exemplo:

USE AdventureWorks2012;

IF EXISTS

(

SELECT \* FROM Production.ProductInventory WHERE Quantity = 0

)

BEGIN;

PRINT 'Replenish Inventory';

END;

Os IFs podem ser aninhados, assim como em programação. O ELSE também pode ser usado, mas segue o raciocínio de que atenderá à próxima linha (ou comando BEGIN END) para executar.

### Laços de Repetição

A palavra-chave WHILE pode ser usada para laços de repetição, sempre seguida da condição a ser analisada e de um bloco de comando com BEGIN e END. Lembre-se sempre de modificar a variável da condição ao final para que não entre em Loop infinito. Exemplo de uso:

DECLARE @Temp INT;

SET @Temp = 0;

WHILE @Temp < 3

BEGIN;

PRINT 'tested condition' + STR(@Temp);

SET @Temp = @Temp + 1;

END;

As palavras-chave CONTINUE e BREAK também podem ser utilizadas em momentos mais complexos.

### Sintaxe GOTO

A sintaxe GOTO diz um bloco que deve ser lido de imediato, ignorando o restante que deveria ser considerado antes dele. É uma maneira AGRESSIVA de controle de fluxo de dados e deve ser evitada. Exemplo de uso:

GOTO ErrorHandler;

Print 'more code';

ErrorHandler:

Print 'Logging the error';

Result:

Logging the error

## Objetos de Gerenciamento Dinâmico

Sp\_help -> retorna informações do servidor, banco de dados, objetos, conexões e mais, listando todos os objetos envolvidos. Caso tenha um parâmetro, retorna informações sobre o mesmo, como uma tabela.

Exemplo> sp\_help ‘tbl\_funcionarios’

O SQL Server possui algumas funções próprias, que agem como constantes internas, que possuem dois @ antes do nome.

Tabelas temporárias podem ser criadas, da mesma forma que são criadas outras tabelas, e duram até o final do seu batch. Para a criação de uma tabela temporária, colocar um # na frente do seu nome. Elas ficam armazenadas no banco de dados tmpbd. Tabelas temporárias globais, acessadas por todos, são criadas com dois # na frente do nome.

Tabelas variáveis também podem ser criadas, elas agem como variáveis, mas são tabelas, com toda a estrutura, salvo exceção de algumas constraints. Para declara-las, coloque @ na frente do nome e defina como uma tabela normal.

OFFSET -> Determina quando as linhas começaram a ser retornadas

FETCH -> Determina o número de linhas a ser retornado

Exemplo:

SELECT DepartmentID, Name

FROM HumanResources.Department

ORDER BY DepartmentID

OFFSET 2 ROWS

FETCH NEXT 5 ROWS ONLY;

Para o tratamento de erros existem algumas funções internas ou o BEGIN TRY END TRY e BEGIN CATCH END CATCH para o tratamento.

# Utilizando Java Spring Boot com SQL Server

Esse site apresenta todo o procedimento de configuração do ambiente para que isso seja possível: <https://springframework.guru/configuring-spring-boot-for-microsoft-sql-server/>

Adendos importantes:

* O projeto inicialmente deve ser construído pelo spring initialzr, contendo as dependências: Dev Tools, JPA, Web Server e o SQL Server Driver.
* Observar a porta 1433 no IPALL do SQL Server Manager

# Administração de Usuários

O primeiro usuário utilizado, assim que o SQL Server é instalado é o SA, não recomendado para uso geral, pois possui todas as permissões de acesso do sistema. Para criação de outros usuários, alguns passos devem ser feitos.

## Login

O login é uma identificação para acesso a uma instância do SQL Server, sendo um usuário mapeado por logins, um login precisa estar criado antes de um usuário que irá utiliza-lo.

A criação do Login pode ser efetuada por:

CREATE LOGIN loginTeste

WITH PASSWORD = 'senha321' MUST\_CHANGE,

CHECK\_EXPIRATION = ON;

O comando MUST\_CHANGE indica que, quando esse login for realizado, essa senha PRECISA ser alterada obrigatoriamente, e o check expiration garante a mudança da mesma.

A localização dos logins no banco pode ser feita em:

SELECT \*

FROM master.sys.sql\_logins;

Essa e outras operações com Login em: <https://www.techonthenet.com/sql_server/users/create_login.php>

## Usuário

A criação do usuário é vinculada a um login já existente, um exemplo:

CREATE USER usuarioTeste2 FOR LOGIN loginTeste;

A localização dos usuários no banco pode ser feita por:

SELECT \*

FROM master.sys.database\_principals;

É importante salientar que quando um usuário é criado, ele vem sem privilégios para operações, que podem ser oferecidos por GRANT e retirados por REVOKE, que podem ser proporcionados para databases ou tables:

GRANT SELECT, INSERT, UPDATE, DELETE ON [db\_teste] TO usuarioTeste2

OU

GRANT ALL ON [db\_teste] TO usuarioTeste2

E para retirar privilégios

REVOKE SELECT, INSERT, UPDATE, DELETE ON [db\_teste] FROM usuarioTeste2

Mais informações sobre o GRANT e o REVOKE em: <https://www.techonthenet.com/sql_server/grant_revoke.php>

Mais operações com usuários em: <https://www.techonthenet.com/sql_server/users/index.php>

# Programando no SQL Server

Todas as ferramentas que envolvem lógica programada no SQL Server.

## Functions

Funções são blocos de programação que quando executados retornam um valor.

A sintaxe padrão na criação de funções (considerando opcionais os parâmetros em colchetes) é:

CREATE FUNCTION [schema\_name.]function\_name

( [ @parameter [ AS ] [type\_schema\_name.] datatype

[ = default ] [ READONLY ]

, @parameter [ AS ] [type\_schema\_name.] datatype

[ = default ] [ READONLY ] ]

)

RETURNS return\_datatype

[ WITH { ENCRYPTION

| SCHEMABINDING

| RETURNS NULL ON NULL INPUT

| CALLED ON NULL INPUT

| EXECUTE AS Clause ]

[ AS ]

BEGIN

[declaration\_section]

executable\_section

RETURN return\_value

END;

Para executar basta chamar a função com algum comando, como um SELECT.

Mais informações sobre funções e execução em: <https://www.techonthenet.com/sql_server/functions.php>

## Procedures

Uma procedure é um bloco de programação armazenado que não retorna algum valor, como a função, mas pode informar se houve sucesso ou falha na sua execução.

A sintaxe padrão é:

CREATE { PROCEDURE | PROC } [schema\_name.]procedure\_name

[ @parameter [type\_schema\_name.] datatype

[ VARYING ] [ = default ] [ OUT | OUTPUT | READONLY ]

, @parameter [type\_schema\_name.] datatype

[ VARYING ] [ = default ] [ OUT | OUTPUT | READONLY ] ]

[ WITH { ENCRYPTION | RECOMPILE | EXECUTE AS Clause } ]

[ FOR REPLICATION ]

AS

BEGIN

[declaration\_section]

executable\_section

END;

Para execução, fazer a declaração de variável que será utilizada e utilizar o comando EXEC.

Exemplo de execução:

USE [test]

GO

DECLARE @site\_name varchar(50);

EXEC FindSite @site\_name OUT;

PRINT @site\_name;

GO

Mais informações sobre Procedures em: <https://www.techonthenet.com/sql_server/procedures.php>

## Laços de Repetição

Os laços de repetição já foram abordados nos Conceitos do T-SQL, na parte de Fluxo Procedural, então aqui serão abordados apenas complementos à iteração.

DECLARE @site\_value INT;

SET @site\_value = 0;

WHILE @site\_value <= 10

BEGIN

IF @site\_value = 2

BREAK;

ELSE

BEGIN

SET @site\_value = @site\_value + 1;

PRINT 'Inside WHILE LOOP on TechOnTheNet.com';

CONTINUE;

END;

END;

PRINT 'Done WHILE LOOP on TechOnTheNet.com';

GO

O comando BREAK encerra o código e o CONTINUE permite que ele continue após o END da operação.

Mais informações sobre laço de repetição em: <https://www.techonthenet.com/sql_server/loops/while.php>

## Sequências

Sequências em SQL Server são números autogerados que atendem critérios definidos, podem ser usados como primary key ou para numerar campos, porém não é indicada para uso em primary key, sendo preferido o IDENTITY, por questões de segurança e respeito a regras. Mais informações sobre a diferença entre SEQUENCE e IDENTITY podem ser encontradas em: <http://www.linhadecodigo.com.br/artigo/3403/diferencas-entre-sequences-x-identity-no-microsoft-sql-server-2012.aspx>

A sintaxe básica para a criação de SEQUENCE é:

CREATE SEQUENCE [schema.]sequence\_name

[ AS datatype ]

[ START WITH value ]

[ INCREMENT BY value ]

[ MINVALUE value | NO MINVALUE ]

[ MAXVALUE value | NO MAXVALUE ]

[ CYCLE | NO CYCLE ]

[ CACHE value | NO CACHE ];

Mais informações sobre o uso de SEQUENCE em <https://www.techonthenet.com/sql_server/sequences.php>

# Banco de Dados Dimensional

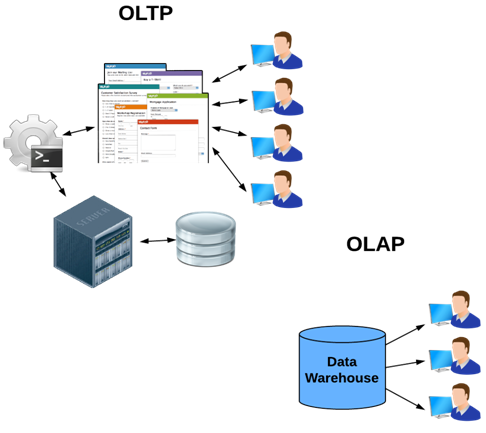
A **modelagem multidimensional**, ou dimensional como às vezes é chamada, **é a técnica de modelagem de banco de dados para o auxílio às consultas do Data Warehouse nas mais diferentes perspectivas**. A visão multidimensional permite o uso mais intuitivo para o processamento analítico pelas ferramentas OLAP (On-line Analytical Processing).

O **OLAP**, do inglês "On-line Analytical Processing", **trata da capacidade de analisar grandes volumes de informações nas mais diversas perspectivas dentro de um Data Warehouse (DW).** O OLAP também faz referência às ferramentas analíticas utilizadas no BI para a visualização das informações gerenciais e dá suporte para as funções de análises do negócio organizacional.

As siglas OLTP e OLAP são bastante utilizadas no universo do [Business Intelligence (BI)](https://canaltech.com.br/business-intelligence/A-composicao-da-Inteligencia-de-Negocios/). Porém, ambas possuem conceitos divergentes e são aplicadas em contextos diferentes.

O OLTP, do inglês "On-line Transaction Processing", é o termo usado para se referir aos sistemas transacionais, ou seja, os sistemas operacionais das organizações. São utilizados no processamento dos dados de rotina que são gerados diariamente através dos sistemas informacionais da empresa e dão suporte às funções de execução do negócio organizacional.

Em resumo podemos dizer que **a grande diferença está no fato de que um está direcionado ao funcionamento dentro do ambiente operacional (OLTP) e o outro com foco essencialmente gerencial (OLAP).**



**Toda modelagem dimensional possuem dois elementos imprescindíveis: as tabelas Fatos e as tabelas Dimensões**. Ambas são obrigatórias e possuem característica complementares dentro de um Data Warehouse.

**As Dimensões são os descritores dos dados oriundos da Fato.** Possui o caráter qualitativo da informação e relacionamento de “um para muitos” com a tabela Fato**. É a Dimensão que permite a visualização das informações por diversos aspectos e perspectivas.**

**As Fatos contém as métricas.** Possui o caráter quantitativo das informações descritivas armazenadas nas Dimensões. **É onde estão armazenadas as ocorrências do negócio e possui relacionamento de “muitos para um” com as tabelas periféricas (Dimensão).**

A modelagem dimensional possui dois modelos: **o modelo estrela (star schema) e o modelo floco de neve (snow flake).** Cada um com aplicabilidade diferente a depender da especificidade do problema. As Dimensões do modelo estrela são desnormalizados, ao contrário do snow flake, que parcialmente possui normalização.

A estrutura relacional diferencia-se da estrutura multidimensional principalmente devido a normalização, pouca redundância e a frequência de atualizações suportadas. A estrutura multidimensional possui, normalmente, desnormalização de tabelas, alta redundância e suporta periodicidade de atualizações de dados muito menor do que uma estrutura relacional convencional.

Mais sobre bancos dimensionais: <https://canaltech.com.br/business-intelligence/entendendo-a-modelagem-multidimensional-19988/>

Mais sobre OLAP/OLTP: <https://canaltech.com.br/business-intelligence/o-que-significa-oltp-e-olap-na-pratica/>

## Integração de Dados

Uma integração consiste em, basicamente, reunir vários componentes independentes e colocá-los para trabalhar em conjunto, extraindo recursos de cada um para resolver problemas ou otimizar processos. Ou seja, quando se trabalha com softwares integrados, eles são capazes de se comunicar entre si e solicitar ao outro as informações necessárias para as suas rotinas.

Todos os dados requeridos são documentados e trabalhados em um layout especial para a integração dos sistemas. Este layout deve dispor de um campo para origem e destino de cada dado, ajudando no entendimento de ambas partes.

A integração pode ser feita entre sistemas, equipamentos e, também, [aplicativos](https://blog.vexpenses.com/aplicativo-mobile-gestao-empresarial/?utm_campaign=meio_funil&utm_medium=post&utm_source=blog&utm_content=integracao-de-sistemas-gestao). As partes integradas podem ser tanto programas especialistas já desenvolvidos, quanto pode-se criar um módulo novo integrado a outros já existentes de uma determinada solução.

Depois que você integrar os sistemas da sua empresa, a informação precisará ser gerada uma única vez e será automaticamente transmitida para todos os outros sistemas que necessitarem dela, prevenindo a perda de dados valiosos. Como consequência desse processo estão os cortes de gastos, a melhoria da produtividade, melhores tomadas de decisões e maior [vantagem competitiva](https://blog.vexpenses.com/estrategia-competitiva/?utm_campaign=meio_funil&utm_medium=post&utm_source=blog&utm_content=integracao-de-sistemas-gestao) para o seu negócio.

Mais sobre conceitos de integração em: <https://blog.vexpenses.com/integracao-de-sistemas-gestao/>

### SQL Server Integration Services

Quando se inicia em SSIS, uma das maiores confusões é identificar o que é Fluxo de Controle e o que é Fluxo de Dados. Bem, resumidamente, o Fluxo de Dados nada mais é que uma Tarefa do Fluxo de Controle. Isso mesmo, Data Flow é uma Task, pode conferir na Toolbox do Control Flow. Só que não é uma Task como as outras, é especial, tanto é que tem uma aba só para ela.

Pense da seguinte forma: Data Flow é a Task mais complexa SSIS.

E quando devo usar o Fluxo de Dados? A resposta é simples: Quando necessitar de uma transferência de dados.

O Fluxo de Dados serve para Importar, exportar e transformar dados, o famoso ETL (Extract, Transform and Load). Essa transferência pode ser de um arquivo de texto para um Banco de Dados, de um Banco de Dados para um arquivo Excel, entre Bancos diferentes (SQL Server e Oracle) e etc.

O Fluxo de Dados possui seus próprios componentes (assim como o Fluxo de Controle), porém tem outros nomes: são as Origens (Data Flow Sources), as Transformações (Data Flow Transformations) e os Destinos (Data Flow Destinations).

Podemos ter como Origem e Destino conexões OLDB, ADO NET, Excel, Flat File (Arquivos de texto), XML e etc.

Como Transformação podemos ter o OLDB Command, Sort, Pivot, Derived Column, Data Conversion, Lookup e etc.

Todas as Origens, Transformações e Destinos do Fluxo de Dados possuem suas próprias Propriedades.

Mais sobre a ferramenta em: <https://www.devmedia.com.br/microsoft-etl-arquitetura-ssis-sql-server-integration-services/30862>

Vídeo explicativo sobre modelagem dimensional e integração: <https://www.youtube.com/watch?v=JLZ3tiRS0oA>

Breve resumo do vídeo:

O principal objetivo da integração é coletar os dados que já existem historicamente, estando ou não em produção, transforma-los em uma forma desejada e aplica-los dentro do Data Warehouse, esse seguindo o padrão de modelagem dimensional, com star schema ou snowflake. Tudo isso resume o processo de ETL (Extract, Trasform e Load).

# Cursores e a leitura linha a linha

Considerando a integração de dados e a carga de dados para alimentar os Data Warehouse, os cursores podem se revelar muito úteis na leitura linha a linha de uma tabela para que sejam feitas operações.

Cursores são estruturas da linguagem T-SQL que permitem o processamento das linhas retornadas por uma consulta (SELECT), através de estruturas complexas de programação, como repetições ou comandos condicionais.

O exemplo básico de cursor consiste em uma repetição (loop) onde um mesmo conjunto de comandos é executado para todas as linhas do retorno de uma consulta.

Um exemplo do uso:

DECLARE @schemaName VARCHAR(30), @procName VARCHAR(30), @fullName VARCHAR(60)

-- Cursor para percorrer os nomes dos objetos

**DECLARE cursor\_objects CURSOR** FOR

SELECT

ROUTINE\_SCHEMA

, ROUTINE\_NAME

FROM

INFORMATION\_SCHEMA.ROUTINES

WHERE

ROUTINE\_TYPE = 'PROCEDURE'

-- Abrindo Cursor para leitura

**OPEN cursor\_objects**

-- Lendo a próxima linha

**FETCH NEXT FROM cursor\_objects INTO @schemaName, @procName**

-- Percorrendo linhas do cursor (enquanto houverem)

**WHILE @@FETCH\_STATUS = 0**

BEGIN

SELECT @fullName = @schemaName + '.' + @procName

EXEC sp\_helptext @fullName

-- Lendo a próxima linha

FETCH NEXT FROM cursor\_objects INTO @schemaName, @procName

END

-- Fechando Cursor para leitura

**CLOSE cursor\_objects**

-- Desalocando o cursor

**DEALLOCATE cursor\_objects**

**OBSERVAÇÕES:**

* Um cursor deve estar sempre associado a uma consulta, especificada ao declarar o cursor.
* O comando FETCH popula as variáveis recebidas como parâmetro com os valores da próxima linha da consulta a ser lida. O número de variáveis passadas como parâmetro deve ser igual ao número de colunas retornadas na consulta associada ao cursor.
* A variável global @@FETCH\_STATUS retorna o resultado da última operação FETCH executada por um cursor na conexão.

O status 0 significa que o comando FETCH retornou uma linha, qualquer outro resultado significa que não houve linha retornada.

* Cursores são estruturas relativamente lentas se comparadas ao desempenho de consultas do banco. O uso descuidado dessa ferramenta pode causar sérios problemas de performance.

Mais sobre a estrutura do Cursor em: <http://www.sqlserverdicas.com/2010/12/cursores-exemplo-basico-de-utilizacao.html>

## Alternativas ao Cursor

O cursor pode trazer alguns problemas de performance, e na maioria dos casos pode ser substituído por tabelas temporárias e loops comuns. Um exemplo abaixo:

-- Create a temporary table, note the IDENTITY

-- column that will be used to loop through

-- the rows of this table

CREATE TABLE #ActiveVisitors (

RowID int IDENTITY(1, 1),

VisitorID int,

FirstName varchar(30),

LastName varchar(30)

)

DECLARE @NumberRecords int, @RowCounter int

DECLARE @VisitorID int, @FirstName varchar(30), @LastName varchar(30)

-- Insert the resultset we want to loop through

-- into the temporary table

INSERT INTO #ActiveVisitors (VisitorID, FirstName, LastName)

SELECT VisitorID, FirstName, LastName

FROM Visitors

WHERE Active = 1

-- Get the number of records in the temporary table

SET @NumberRecords = @@RowCount

--You can use: SET @NumberRecords = SELECT COUNT(\*) FROM #ActiveVisitors

SET @RowCounter = 1

-- loop through all records in the temporary table

-- using the WHILE loop construct

WHILE @RowCounter <= @NumberRecords

BEGIN

SELECT @VisitorID = VisitorID, @FirstName = FirstName, @LastName = LastName

FROM #ActiveVisitors

WHERE RowID = @RowCounter

EXEC MyCallingStoredProc @VisitorID, @FirstName, @LastName

SET @RowCounter = @RowCounter + 1

END

-- drop the temporary table

DROP TABLE #ActiveVisitors

Mais exemplos em: <https://stackoverflow.com/questions/5425642/what-is-an-alternative-to-cursors-for-sql-looping>