Programação em Banco de Dados

Autoria: Douglas Fugita de Oliveira Cezar

Tema 01

Criação de Estruturas e Gerenciamento de Acesso

Anhanguera

Tema 01

Criação de Estruturas e Gerenciamento de Acesso

Autoria: Douglas Fugita de Oliveira Cezar

Como citar esse documento:

CEZAR, Douglas Fugita de Oliveira. *Programação em Banco de Dados*: Criação da Estruturas e Gerenciamento de Acesso. Caderno de Atividades. Valinhos: Anhanguera Educacional, 2015.

Índice

CONVITE À LEITURA

Neste caderno de atividades você terá o primeiro contato com a linguagem utilizada na programação de banco de dados, a *Structured Query Language* (SQL). Aqui você aprenderá como criar as estruturas básicas de um banco de dados, desde a criação do próprio banco, passando pela criação das tabelas e seus campos, chegando até nas *constraints*, que são restrições que visam garantir a integridade dos dados que serão armazenados nas tabelas.

Este material será importante para você que deseja compreender uma das principais camadas do desenvolvimento de sistemas: a camada de dados, que é a responsável por armazenar e garantir a integridade de toda informação que trafega pelo sistema.

Utilize os exemplos deste caderno de estudos como guia, e crie também sua própria estrutura de banco de dados para que possa avançar nos estudos.

POR DENTRO DO TEMA

Criação de estruturas e Gerenciamento de Acesso

Uma linguagem de programação de banco de dados pode ser dividida em três grandes grupos: a linguagem de definição de dados (DDL – *Data Definition Language*), a linguagem de controle de dados (*DCL – Data Control Language*) e a linguagem de manipulação de dados (*DML – Data Manipulation Language*).

Através da linguagem de definição de dados é possível que todas as estruturas básicas de um banco de dados sejam criadas, desde uma simples tabela até toda a configuração do arquivo que irá armazenar todo o banco de dados.

Você poderá criar uma base dados através do comando CREATE DATABASE. No quadro 1.1 você poderá acompanhar a criação da base de dados que abrigará todos os componentes que serão utilizados como exemplo por este caderno e, logo em seguida, a explicação dos comandos.

Quadro 1.1 – Create Database

```
CREATE DATABASE [BD_TADS]
   ON (PRIMARY NAME = 'BD_TADS_FILE',
       FILENAME = 'Z:\Aula_BD_TADS\',
       SIZE = 5120KB,
       MAXSIZE = UNLIMITED ,
       FILEGROWTH = 1MB )
   LOG ON ( NAME = 'BD_TADS_LOG',
       FILENAME = ' Z:\Aula_BD_TADS\',
       SIZE = 5120KB,
       MAXSIZE = UNLIMITED ,
       FILEGROWTH = 1MB )
   COLLATE Latin1_General_BIN
```

Criar ou definir uma base de dados é a atividade de dar um nome a ela, bem como definir o local em que ela será armazenada e o tamanho dos arquivos que a compõe. Toda base de dados deve ter ao menos um arquivo de dados e um **arquivo de log**. O argumento BD_TADS após o comando CREATE DATABASE define o nome do novo banco de dados.

Em seguida, pode ser observada uma sessão iniciada por ON e o atributo PRIMARY. Esta sessão contém informações sobre o arquivo que armazenará todas as tabelas da base de dados, também chamado de arquivo de dados primário. FILENAME é o atributo no qual deverá ser indicado o local para armazenamento do arquivo de dados primário. Na sequência, temos as propriedades SIZE, MAXSIZE e FILEGROWTH, que estão diretamente relacionados ao tamanho do arquivo. No exemplo do quadro 1.1, a base está sendo criada com 5120Kb (5MB), possui tamanho máximo ilimitado e, sempre que for necessário aumentar o arquivo, ele crescerá numa taxa de 20%.

A próxima sessão inicia-se com LOG ON, e as propriedades desta sessão são semelhantes à anterior. No entanto, todas as informações são relacionadas à criação do arquivo de log.

Por último, você pode ver o atributo COLLATE, que está relacionado aos caracteres permitidos para esta base de dados, bem como as regras que serão utilizadas para manipulação dos dados. O esquema “Latin1_General_BIN” indica que serão utilizados os conjuntos latinos de caracteres e que a ordem de classificação será binária, além de diferenciar caracteres maiúsculos e minúsculos e acentuados ou não.

Uma vez que o arquivo esteja criado, ele pode ser alterado utilizando o comando ALTER DATABASE. O quadro 1.2 mostra um exemplo de alteração da base criada.

Quadro 1.2 - ALTER DATABASE

```
ALTER DATABASE [BD_TADS]
  MODIFY FILE ( NAME = ‘BD_TADS_log’,
    SIZE = 204800KB )
```

O quadro 1.2 mostra a alteração do tamanho do arquivo de LOG para 200MB. Além do exemplo citado, o comando ALTER DATABASE pode alterar não somente as propriedades indicadas no comando CREATE DATABASE, mas também as opções de **estatísticas**, **cursores**, recuperação e permissão de leitura e escrita no arquivo como um todo.

A base de dados pode ser excluída, juntamente com seus arquivos de dados e de LOG. Para esta função é utilizado o comando DROP DATABASE, e o exemplo é mostrado no quadro 1.3.

Quadro 1.3 - DROP DATABASE

```
DROP DATABASE [BD_TADS]
```

Após a criação da base de dados, o próximo componente da estrutura básica de um banco de dados que será criado serão as tabelas. As tabelas são compostas por campos e registros (colunas e linhas, respectivamente) e, neste momento, somente serão definidos os campos e seus tipos.

Para a criação da tabela deve ser usado o comando CREATE TABLE e um exemplo do seu uso pode ser visto no quadro 1.4.

Quadro 1. 4 – CREATE TABLE

```
CREATE TABLE [Produto] (
    [ID_Produto]          int IDENTITY NOT NULL,
    [Nome_Produto]        nvarchar(25) NOT NULL,
    [Descricao_Produto]   ntext NOT NULL,
    [Habilitado_Produto]  binary NULL
    )
```

O quadro 1.4 indica a criação de uma tabela que será a responsável por armazenar as informações referentes aos produtos que serão cadastrados na base de dados.

Neste exemplo, o argumento Produto do comando CREATE TABLE indica o nome da tabela que será criada. A seguir, entre parênteses são indicados os campos desta tabela, sendo eles: ID_Produto, Nome_Produto, Descricão_Produto e Habilitado_Produto. Cada um deles possui um tipo diferente, devido à natureza dos dados que eles serão capazes de armazenar.

O campo ID_Produto será um campo numérico inteiro e gerado automaticamente, e isto é indicado pelos argumentos int e IDENTITY, respectivamente. Este campo também possui uma indicação de NOT NULL, ou seja, não é permitido que os registros que serão armazenados nesta tabela tenham este campo **NULO**.

Os campos Nome_Produto e Descricao_Produto, embora ambos sejam compostos de caracteres alfanuméricos, possuem tipos de dados diferentes. O tipo nvarchar(*n*) limita o tamanho do campo ao atributo *n* indicado na declaração do tipo, que, no exemplo do quadro 1.4 permite a criação de um campo que terá a possibilidade de armazenar 25 caracteres. Já o tipo ntext é um tipo especial que permite um número bem maior de caracteres (2^30 caracteres).

O campo Habilitado_Produto é do tipo binary, pois seu uso será somente para indicar que o produto em questão poderá ser utilizado ou não. A indicação de NULL indica que este campo poderá conter um valor NULO.

Assim como a base de dados, as tabelas também podem sofrer alterações, e isto será feito através do comando ALTER TABLE.

Quadro 1. 5 – ALTER TABLE ADD

```
ALTER TABLE [Produto]
    ADD [ValorVenda_Produto] float NOT NULL,
    [QuantidadeEstoque_Produto] float NOT NULL,
    [UnidadeMedida_Produto] varchar(3) NOT NULL
```

Os mesmos argumentos utilizados para a criação da tabela devem ser utilizados no comando ALTER TABLE junto ao comando ADD, para que as colunas possam ser incluídas na estrutura criada anteriormente.

Os campos ValorVenda_Produto e QuantidadeEstoque_Produto foram criados para armazenar campos do tipo float, ou seja, números com ponto flutuante. Todos os campos criados nesta alteração de tabela devem respeitar a regra de não armazenar um valor NULO.

Uma coluna também pode ser removida de uma tabela e, para isso, deve ser utilizado o mesmo comando ALTER TABLE, no entanto, isso deve ser feito com o comando DROP, como mostrado no quadro 1.6.

Quadro 1.6 – ALTER TABLE DROP

```
ALTER TABLE [dbo].[Produto]
    DROP COLUMN [Habilitado_Produto]
```

O exemplo que será utilizado durante este curso utilizará a estrutura pela query, indicada no quadro 1.7.

Quadro 1.7 – Estrutura da base de dados BD_TADS

```
CREATE TABLE [Produto] (
    [ID_Produto]          int IDENTITY NOT NULL,
    [Nome_Produto]        nvarchar(25) NOT NULL,
    [Descricao_Produto]   ntext NOT NULL,
    [Habilitado_Produto]  binary NULL
```

```
    [ValorVenda_Produto] float NOT NULL,
    [QuantidadeEstoque_Produto] float NOT NULL,
        [UnidadeMedida_Produto]  varchar(3)  NOT
NULL
    )

CREATE TABLE [Clientes] (
    [ID_Cliente]    int IDENTITY NOT NULL,
    [Nome_Cliente]  nvarchar(25) NOT NULL,
    [Telefone_Cliente]      nvarchar(25) NOT NULL,
    [CPF_Cliente] nvarchar(20) NOT NULL,
    [ID_Municipio]  int NOT NULL,
    [ID_Vendedor]   int NOT NULL
    )

CREATE TABLE [Municipo] (
    [ID_Municipo]       int IDENTITY NOT NULL,
    [Nome_Municipio]    nvarchar(60) NOT NULL,
    [UF_Municipio]      nvarchar(2) NULL
    )
```

PORDENTRODOTEMA

```
CREATE TABLE [Vendedor] (
   [ID_Vendedor]       int IDENTITY NOT NULL,
   [Nome_Vendedor]     nvarchar(40) NULL,
   [Comissao_Vendedor] float NULL
   )

CREATE TABLE [NF] (
   [Numero_NF]         nvarchar(9) NOT NULL,
   [Serie_NF]          nvarchar(3) NOT NULL,
   [ID_Cliente]        int NOT NULL,
   [ID_Vendedor]       int NOT NULL,
   [DataEmissao_NF]    datetime NOT NULL,
   [DataSaida_NF]      datetime NOT NULL,
   [ValorTotal_NF]     float NOT NULL
   )

CREATE TABLE [NFItens] (
   [Numero_NF]            nvarchar(9) NOT NULL,
   [Serie_NF]             nvarchar(3) NOT NULL,
   [Item_NFItens]         int NOT NULL,
   [ID_Produto]           int NOT NULL,
   [Qtdade_NFItens]       float NOT NULL,
   [PrecoVenda_NFItens]   float NOT NULL,
   [Desconto_NFItens]     float NULL
   )
```

PORDENTRODOTEMA

O próximo passo é garantir a integridade dos dados que serão armazenados na base de dados e, para isso, a linguagem SQL conta com as restrições, ou *constraints*. Elas são utilizadas a fim de garantir a integridade dos dados, mesmo em casos de alterações ou inserções de dados diretamente na base de dados por usuário que tenham permissão para tal.

As restrições podem ser divididas em três tipos: integridade de domínio, integridade de entidade e integridade referencial.

A integridade de domínio está relacionada aos campos de uma tabela e podem especificar qual é o conjunto de dados válidos para aquele campo. Um exemplo de restrição de integridade de domínio que você já viu neste caderno é a proibição de valor nulo para um determinado campo. As restrições deste tipo são: DEFAULT, CHECK e REFERENCES.

Integridade de entidade está relacionada aos registros, e é utilizada para garantir que cada um dos registros possua um identificador que torne possível sua identificação de forma exclusiva. Pode ser considerada uma restrição de integridade de entidade o uso de chaves-primárias. As restrições deste tipo são: PRIMARY KEY e UNIQUE.

A integridade referencial está correlatada aos relacionamentos entre tabelas e deve garantir que o relacionamento entre tabelas ligadas por chaves-primárias / estrangeiras seja mantido, proibindo a alteração de dados em uma tabela sem que as mudanças sejam refletidas na outra tabela. As restrições deste tipo são: FOREIGN KEY e CHECK.

As *constraints* podem ser definidas durante a criação da tabela, através do comando CREATE TABLE, ou então por uma alteração de tabela existente, através do comando ALTER TABLE.

A restrição de integridade de domínio DEFAULT deve ser utilizada quando um campo deve ser preenchido com um valor padrão e quando este campo for deixado em branco. Você pode observar um exemplo no quadro 1.8, no qual, sempre que um produto for cadastro e a quantidade em estoque não for informada, o valor será preenchido com 0 (zero).

Quadro 1.8 – *constraint* DEFAULT

```
ALTER TABLE [Produto]
   ADD CONSTRAINT [DF_QuantidadeEstoque_Produto] DEFAULT 0 FOR [QuantidadeEstoque_Produto]
```

A restrição CHECK faz uma verificação dos dados baseados em uma cláusula condicional. O quadro 1.9 mostra a inclusão de cláusulas CHECK para verificar se, na tabela NFItens, os campos Qtdade_NFItens e PrecoVenda_NFItens possuem valores maiores que 0 (zero) e se o valor total do item é maior que o valor do campo Desconto_NFItens.

Quadro 1.9 – *constraint* CHECK

```
ALTER TABLE dbo.NFItens
ADD CONSTRAINT CK_Qtdade_NFItens CHECK (Qtdade_NFItens > 0),
CONSTRAINT CK_PrecoVenda_NFItens CHECK (PrecoVenda_NFItens > 0),
CONSTRAINT CK_Desconto_NFItens CHECK (PrecoVenda_NFItens * Qtdade_NFItens > Desconto_NFItens)
```

A restrição PRIMARY KEY é utilizada para identificar quais campos irão compor a chave-primária, ou seja, o identificador único de cada registro. No quadro 1.10 você pode acompanhar a criação da chave-primária PK_SerieNumeroNF, no qual os campos Numero_NF e Serie_NF são relacionados como componentes da chave-primária.

Quadro 1. 10 – *constraint* PRIMARY KEY

```
ALTER TABLE NF
ADD CONSTRAINT PK_SerieNumeroNF PRIMARY KEY (Numero_NF, Serie_NF)
```

Ao definir a chave-primária, deve-se ter em mente que somente uma chave pode ser definida por tabela.

A *constraint* UNIQUE define um conjunto de campos no qual seu valor deve ser único em toda tabela. Ele é semelhante à chave-primária, e é bastante útil quando outra informação deve ser mantida como única, como o exemplo do quadro 1.11.

Quadro 1. 11 – *constraint* UNIQUE

```
ALTER TABLE Clientes
ADD CONSTRAINT PK_ID_Cliente PRIMARY KEY (ID_Cliente),
CONSTRAINT U_CPF_Cliente UNIQUE (CPF_Cliente)
```

As restrições FOREIGN KEY e REFERENCES geralmente são utilizadas em conjunto. A primeira é utilizada para indicar o relacionamento entre duas tabelas indicando o campo que será utilizado como chave-estrangeira. A segunda restrição é utilizada para indicar qual será a chave-primária da outra tabela que irá se relacionar com a chave-estrangeira indicada anteriormente. Um exemplo pode ser visto no quadro 1.12.

POR DENTRO DO TEMA

Quadro 1. 12 – *constraints* FOREIGN KEY e REFERENCES

```
ALTER TABLE Vendedor
ADD CONSTRAINT PK_ID_Vendedor PRIMARY KEY (ID_Vendedor)

ALTER TABLE Clientes
ADD CONSTRAINT FK_ID_Vendedor FOREIGN KEY (ID_Vendedor)
   REFERENCES Vendedor(ID_Vendedor)
```

Agora, a estrutura está pronta, estando definido o banco de dados, as tabelas e também os recursos que permitem que seja garantida a integridade dos dados que estão por vir.

ACOMPANHE NA WEB

Relação de parâmetros de collation do MS SQL Server.

- Este site contém todas as possibilidades de collation que podem ser utilizadas na implementação de um banco de dados.

Link: <http://msdn.microsoft.com/pt-br/library/ms188046(v=sql.105).aspx>. Acesso em: 26 mar. 2014.

Tipos de dados (Transact SQL).

- Aqui contém a relação de tipos de dados e o detalhamento de cada tipo que pode ser utilizado durante a criação de uma tabela

Link: <http://msdn.microsoft.com/pt-br/library/ms187752.aspx>. Acesso em: 26 mar. 2014.

AGORAÉASUAVEZ

Instruções:

Agora, chegou a sua vez de exercitar seu aprendizado. A seguir, você encontrará algumas questões de múltipla escolha e dissertativas. Leia cuidadosamente os enunciados e atente-se para o que está sendo pedido.

Questão 1

Explique o que é a Linguagem de Definição de Dados (DDL) e qual sua função para um banco de dados.

Questão 2

Baseado na query abaixo, identifique quais tipos de integridade são atendidas:

```
ALTER TABLE NF
ADD CONSTRAINT PK_SerieNumeroNF PRIMARY KEY (Numero_NF, Serie_NF)
CONSTRAINT U_SerieNumeroNF UNIQUE (Numero_NF, Serie_NF)
CONSTRAINT CK_ValorTotalNF CHECK (ValorTotalNF > 0)
```

a) ( ) Integridade referencial.

b) ( ) Integridade de domínio.

c) ( ) Integridade de entidade.

d) ( ) Nenhuma das respostas anteriores.

AGORA É A SUA VEZ

Questão 3

Marque Verdadeiro ou Falso para as afirmações abaixo:

a) ( ) Uma base de dados pode ser criada somente com um arquivo de LOG.

b) ( ) Após a criação da base de dados, sua estrutura não pode ser alterada.

c) ( ) O comando CREATE TABLE pode ser utilizado tanto para criar arquivos de dados como para criar tabelas.

d) ( ) O atributo COLLATE pode ser utilizado para indicar o tamanho máximo do banco de dados.

Questão 4

A estrutura de banco de dados utilizada durante este caderno deve ser complementada/alterada, de forma que você inclua uma tabela para armazenar grupo de produtos que, por sua vez, deve estar relacionada com a tabela produtos.

A tabela de grupo de produtos deve conter os seguintes campos:

- Código do Grupo de Produtos.
- Descrição do Grupo.
- Percentual máximo de desconto para o grupo.

Questão 5

Qual seria o comando para alterar a estrutura da base de dados utilizada, mudando a taxa de crescimento do arquivo primário para 200Mb?

FINALIZANDO

Neste caderno você teve a oportunidade de aprender sobre os principais comandos da linguagem de definição de dados.

Estes comandos são responsáveis pela criação do alicerce que sustentará o banco de dados por toda sua vida útil, por isso, a fase da definição da estrutura é bastante importante.

REFERÊNCIAS

MICROSOFT SQL Server. *Nome de agrupamento do Windows (Transact – STL)*. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms188046%28v=sql.105%29.aspx>. Acesso em: 26 mar. 2014.

MICROSOFT SQL Server. *Tipos de dados (Transact – STL)*. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms187752.aspx>. Acesso em: 26 mar. 2014.

SILBERSCHATZ, Abrahan; KORTH, Henry; SUDARSHAN, S. *Sistema de Banco de Dados*. Elsevier, 2012.

GLOSSÁRIO

Arquivo de log: arquivo que, por um determinado período de tempo, armazena todas as transações realizadas em um banco de dados.

Estatística: é uma coleção de dados detalhados sobre o banco de dados, e são utilizadas pelo otimizador do banco para que seja possível propor melhores planos de execução.

Cursores: recurso que possibilita a manipulação de uma linha ou um pequeno bloco de dados.

Nulo: campo que não possui nenhum valor.

GABARITO

Questão 1

Resposta: A DDL é um conjunto de comandos responsável pela criação e manutenção de toda base de dados, desde o arquivo que armazenará os dados até as estruturas lógicas como tabelas, campos, views e cursores.

Questão 2

Resposta: B, C.

Questão 3

Resposta: F, F, F, F.

GABARITO

Questão 4

Resposta:

```
CREATE TABLE [GrupoProduto] (
   [ID_GrupoProduto]          int IDENTITY NOT NULL,
   [Descricao_GrupoProduto]        nvarchar(25) NOT NULL,
   [PerctDesconto_GrupoProduto] float NOT NULL,
   )
ALTER TABLE [Produto]
   ADD [ID_GrupoProduto] int NOT NULL,

ALTER TABLE GrupoProduto
ADD CONSTRAINT PK_ID_GrupoProduto PRIMARY KEY (ID_GrupoProduto)

ALTER TABLE Produto
ADD CONSTRAINT FK_ID_GrupoProduto FOREIGN KEY (ID_GrupoProduto)
   REFERENCES GrupoProduto(ID_GrupoProduto)
```

Questão 5

Resposta:

```
ALTER DATABASE [BD_TADS]
  MODIFY FILE ( NAME = 'BD_TADS',
    FILEGROWTH = 200MB )
```


Anhanguera