Aula 2

Data Definition Language
DDL

Faculdade IMPACTA

Faculdade IMPACTA

Introdução à Data Definition Language DDL

Data Definition Language (DDL)

Para persistirem os dados em um banco de dados, precisamos armazená-los em objetos do tipo tabela. Comandos de criação, alteração ou eliminação de objetos fazem parte da categoria de comandos **DDL (Data Definition Language)**.

Para iniciar a criação de tabelas, precisaremos definir adequadamente suas colunas de forma que permitam a movimentação de informações segundo seus tipos de dados e obrigatoriedade ou não de informação.

Data Definition Language (DDL)

Sintaxe básica para o comando de criação de tabelas:

Uma atenção especial deve ser dada a em que banco de dados estamos criando a estrutura de dados, portanto usamos o comando **USE <database>** para posicionar na base de dados requerida:

```
USE <database>
GO

CREATE TABLE <nome da tabela>
    (
    <nome coluna 1> <tipo da coluna> (<tamanho da coluna>) [NOT NULL]
    , <nome coluna 2> <tipo da coluna> (<tamanho da coluna>) [NOT NULL]
    , ...
    );
```

Data Definition Language (DDL)

O comando anterior é uma simplificação da cláusula **CREATE TABLE**, sendo que a mesma cláusula aceita inúmeros parâmetros e objetos associados. Como exemplo do comando mostrado, as cláusulas abaixo criam a tabela **Aluno** na base de dados **FIT**:

```
USE FIT
GO

CREATE TABLE Aluno
    (
    Matricula int
    , Nome varchar(20)
    , MeioNome varchar(20)
    , SobreNome varchar(20)
    );
```

Data Definition Language (DDL)

As cláusulas **NULL** e **NOT NULL** definem se o preenchimento de determinada coluna é ou não obrigatório. A cláusula **NULL** é a padrão, ou seja, quando não mencionamos nada, como na criação da tabela do slide anterior, os campos PERMITIRÃO valores **NULL**.

Se quisermos OBRIGATORIAMENTE que um campo seja preenchido, utilizamos o **NOT NULL** imediatamente após a definição do tipo de dados da coluna:

```
CREATE TABLE Veiculo
    (
    Placa char(8) NOT NULL
    , Marca varchar(20)  NOT NULL
    , NomeProprietario varchar(60)
    );
```

Data Definition Language (DDL)

Podemos precisar que uma determinada coluna GERE números automaticamente, como, por exemplo, número de matrícula.

Os bancos de dados possuem funções específicas para geração de números e podem ser associados a uma coluna. O SQL Server permite que qualquer tabela tenha uma campo com autonumeração, mas apenas uma coluna da tabela pode receber essa funcionalidade:

IDENTITY (Seed, Increment)

Em que **SEED** representará o primeiro número a ser gerado pela série, e **INCREMENT** significa de quanto em quanto os próximos números serão gerados.

Data Definition Language (DDL)

Perceba que, dependendo do tipo de dados, o **IDENTITY** não se encaixará. Veja alguns exemplos da função:

IDENTITY (Seed, Increment)

IDENTITY ou IDENTITY (1, 1) □ É o padrão da função (qualquer tipo numérico)

IDENTITY (0, 1) □ Qualquer tipo numérico. Inicia no 0 e incrementa de 1 em 1

IDENTITY (0, 10) □ Tinyint ou maior. Inicia no 0, 10, 20, ...

Data Definition Language (DDL)

Note que, por definição, uma coluna autonumerada será automaticamente colocada como **NOT NULL**, mesmo não escrevendo explicitamente esta cláusula. O exemplo abaixo mostra a aplicação do **IDENTITY**:

```
CREATE TABLE Veiculo (
    idVeiculo INT NOT NULL  IDENTITY
    , Placa char(8) NOT NULL
    , Marca varchar(20)  NOT NULL
    );
CREATE TABLE Aluno (
    Matricula int IDENTITY (500, 1)
    , Nome varchar(20)
    , MeioNome varchar(20)
    , SobreNome varchar(20)
    );
```

Data Definition Language (DDL)

PRIMARY KEY: Podemos ter uma e apenas uma chave primária em cada tabela. Por definição, chave primária possui um número único para todos os registros, ou seja, dado um valor correspondente à chave primária de uma tabela, o retorno máximo de registros é uma linha.

A chave primária pode ser simples (apenas uma coluna) ou composta (mais de uma coluna).

A definição da chave primária (primary key) segue o seguinte modelo:

```
CONSTRAINT <nome da primary key> PRIMARY KEY (coluna1, coluna2, ...)
```

Data Definition Language (DDL)

Exemplo:

```
CREATE TABLE Aluno
	(
	Matricula int  NOT NULL IDENTITY (500, 1)
	, Nome varchar(20)
	, MeioNome varchar(20)
	, SobreNome varchar(20)
	, CONSTRAINT pkAluno PRIMARY KEY (Matricula)
	);
```

Data Definition Language (DDL)

FOREIGN KEY: Chave estrangeira faz o relacionamento entre uma ou mais colunas de uma tabela com a chave primária ou única de outra tabela. O formato do comando deve referir as colunas de ambas as tabelas e a tabela onde temos a chave primária.

Uma tabela pode ter várias chaves estrangeiras para outras tabelas, representando o relacionamento que possui com cada uma das outras tabelas.

CONSTRAINT <nome da foreign key> FOREIGN KEY (coluna1, coluna2, ...)

REFERENCES <tabela da primary key> (coluna1, coluna2, ...)

Data Definition Language (DDL)

Faculdade IMPACTA

Exemplo:

```
CREATE TABLE Aluno
    (
    Matricula int not null IDENTITY (500, 1)
    , Nome varchar(20)
    , CONSTRAINT pkAluno PRIMARY KEY (Matricula)
    );
CREATE TABLE Prova (
    idProva int NOT NULL  IDENTITY (1, 1)
    , Matricula int NOT NULL
    , Nota decimal(4,2) NOT NULL
    , CONSTRAINT pkProva PRIMARY KEY (idProva)
    , CONSTRAINT fkProva FOREIGN KEY (Matricula)
     REFERENCES Aluno(Matricula)
    );
```

Data Definition Language (DDL)

Faculdade IMPACTA

```
CREATE TABLE Aluno
    (
    Matricula int not null IDENTITY (500, 1)
    , Nome varchar(20)
    , CONSTRAINT pkAluno
      PRIMARY KEY (Matricula)
    );
```

Matricula	Nome
500	José
501	Pedro
502	Mario

Data Definition Language (DDL)

```
CREATE TABLE Prova
      (
      idProva int NOT NULL  IDENTITY (1, 1)
      , Matricula int  NOT NULL
      , Nota decimal(4,2) NOT NULL
      , CONSTRAINT pkProva PRIMARY KEY (idProva)
      , CONSTRAINT fkProva FOREIGN KEY (Matricula)
        REFERENCES Aluno(Matricula)
      );
```

idProva	Matricula	Nota
1	500	9
2	500	8
3	502	7
4	502	3
5	502	1

Atividades extraclasse

- Leitura do arquivo PDF disponibilizado na plataforma