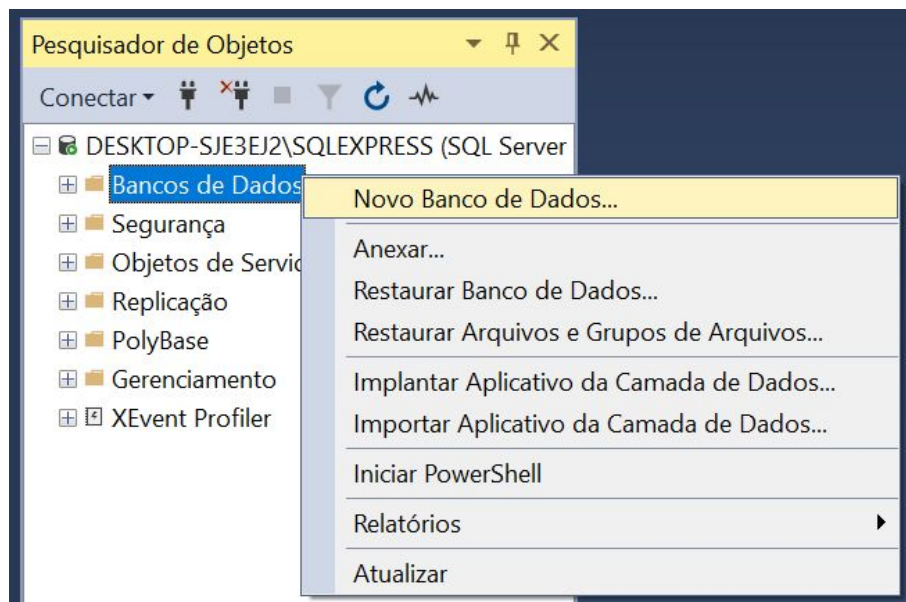


Introdução a Banco de dados Comandos SQL Server

Como criar um banco de dados

Para criarmos um novo Banco de dados:



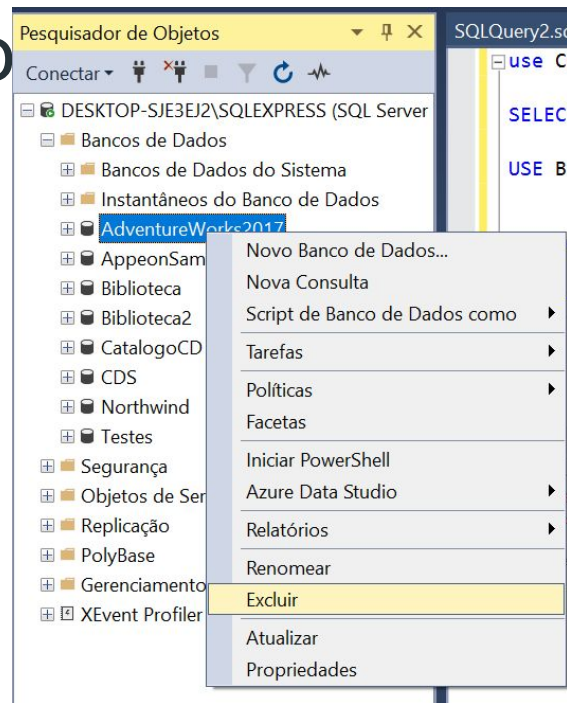
Comando USE

O comando USE instrui o SGBD a utilizar o banco de dados especificado para rodar os comandos.

USE nome_banco_de_dados;

Excluir Banco de Dados

Podemos excluir um banco de dados existente com o comando



SQL Constraints (Restrições) no SQL

- ◎ As Restrições são regras aplicadas nas colunas de uma tabela.
- ◎ São usadas para limitar os tipos de dados que são inseridos.
- ◎ Podem ser especificadas no momento de criação da tabela (CREATE) ou após a tabela ter sido criada (ALTER)

SQL Constraints (Restrições) no SQL Server

As principais constraints são as seguintes:

- ◎ NOT NULL
- ◎ UNIQUE
- ◎ PRIMARY KEY
- ◎ FOREIGN KEY
- ◎ DEFAULT

NOT NULL

- ◎ A constraint NOT NULL impõe a uma coluna a NÃO aceitar valores NULL.
- ◎ Ou seja, a constraint NOT NULL obriga um campo a sempre possuir um valor.
- ◎ Deste modo, não é possível inserir um registro (ou atualizar) sem entrar com um valor neste campo.

UNIQUE

- © A restrição UNIQUE identifica de forma única cada registro em uma tabela de um banco de dados.
- © As constraints UNIQUE e PRIMARY KEY garantem a unicidade em uma coluna ou conjunto de colunas.

UNIQUE

- © Uma constraint PRIMARY KEY automaticamente possui uma restrição UNIQUE definida, portanto não é necessário especificar essa constraint neste caso.

Exemplo:

Phone VARCHAR(12) NOT NULL
UNIQUE,

UNIQUE

- © É possível termos várias constraints UNIQUE em uma mesma tabela, mas apenas uma Chave Primária por tabela (lembrando que uma PK pode ser composta, ou seja, constituída por mais de uma coluna – mas ainda assim, será uma única chave primária).

PRIMARY KEY

- ◎ A restrição PRIMARY KEY (Chave Primária) identifica de forma única cada registro em uma tabela de banco de dados.
- ◎ As Chaves Primárias devem sempre conter valores únicos.

Exemplo:

ID_Livro SMALLINT INDENTITY PRIMARY KEY,

PRIMARY KEY

- © Uma coluna de chave primária não pode conter valores NULL
- © Cada tabela deve ter uma chave primária e apenas uma chave primária.

FOREIGN KEY

- © Uma FOREIGN KEY (Chave Estrangeira) em uma tabela é um campo que aponta para uma chave primária em outra tabela. Desta forma, é usada para criar os relacionamentos entre as tabelas no banco de dados.

FOREIGN KEY

Veja um exemplo de restrição Foreign Key aplicada:

```
CONSTRAINT fk_ID_Autor FOREIGN KEY  
(ID_Autor)
```

```
REFERENCES tbl_autores(ID_Autor)
```

Neste exemplo a chave primária está na tabela `tbl_autores` e uma chave estrangeira de nome `ID_Autor` foi criada na tabela atual, usando o nome `fk_ID_Autor`

DEFAULT

- ◎ A restrição DEFAULT é usada para inserir um valor padrão especificado em uma coluna.
- ◎ O valor padrão será adicionado a todos os novos registros caso nenhum outro valor seja especificado na hora de inserir dados.

Exemplo:

nacionalidade varchar(20) default 'Brasil',

Opções de Chave Estrangeira no SQL

Existem algumas opções aplicáveis às chaves estrangeiras que auxiliam a manter a integridade dos dados nas tabelas do banco de dados. Vamos relembrar a sintaxe SQL para criação de uma chave estrangeira em uma definição de tabela.

Opções de Chave Estrangeira no SQL Server

[CONSTRAINT nome_chave_estrangeira] FOREIGN KEY (nomes de colunas separados por vírgulas)


REFERENCES nome_tabela_pai (nomes de colunas separados por vírgulas na tabela pai)

[ON DELETE ação referencial]

[ON UPDATE ação referencial];

Opções de Chave Estrangeira no SQL

Os itens entre colchetes [] são opcionais. **ON DELETE** significa que a ação referencial será executada quando um registro for excluído da tabela pai, e **ON UPDATE** indica que a ação referencial será executada quando um registro for modificado na tabela pai.



Opções de Chave Estrangeira no SQL Server

As principais opções para as ações referenciais são as seguintes:

- ◎ **CASCADE**: A opção CASCADE permite excluir ou atualizar os registros relacionados presentes na tabela filha automaticamente, quando um registro da tabela pai for atualizado (**ON UPDATE**) ou excluído (**ON DELETE**). É a opção mais comum aplicada.

Opções de Chave Estrangeira no SQL

- ◎ **RESTRICT:** Impede que ocorra a exclusão ou a atualização de um registro da tabela pai, caso ainda haja registros na tabela filha. Uma exceção de violação de chave estrangeira é retornada. A verificação de integridade referencial é realizada antes de tentar executar a instrução UPDATE ou DELETE

Opções de Chave Estrangeira no SQL Server

- ◎ **SET NULL:** Esta opção é usada para definir com o valor NULL o campo na tabela filha quando um registro da tabela pai for atualizado ou excluído.

Opções de Chave Estrangeira no SQL

- ◎ **NO ACTION:** Essa opção equivale à opção RESTRICT, porém a verificação de integridade referencial é executada após a tentativa de alterar a tabela. É a opção padrão, aplicada caso nenhuma das opções seja definida na criação da chave estrangeira.

Opções de Chave Estrangeira no SQL Server

- ◎ **SET DEFAULT:** “Configura Padrão” – Define um valor padrão na coluna na tabela filha, aplicado quando um registro da tabela pai for atualizado ou excluído.

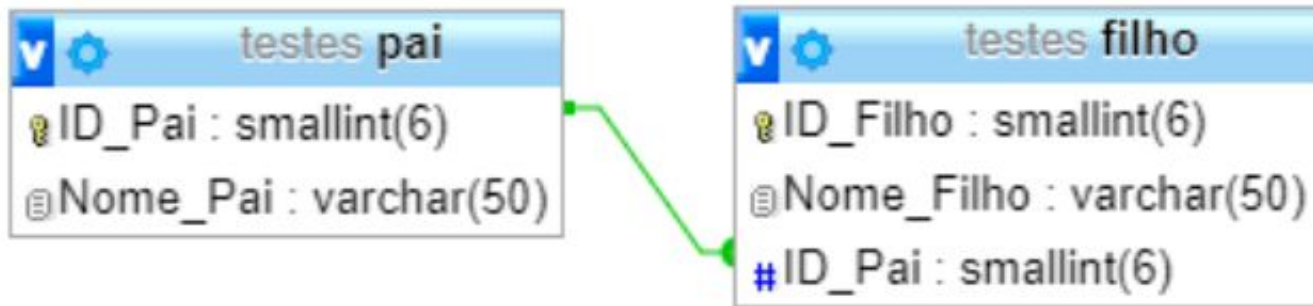
Opções de Chave Estrangeira no SQL Server

Vejamos um exemplo usando a cláusula **ON DELETE CASCADE**, que é uma das mais comuns usadas em chaves estrangeiras. Todos os exemplos mostrados aqui também podem ser utilizados com a cláusula **ON UPDATE** e, na prática, podemos usar ambas as cláusulas na mesma tabela.

Opções de Chave Estrangeira no SQL Server

Para isso, vamos criar um banco de dados de nome “testes”, contendo duas tabelas relacionadas, chamadas de “Pai” e “Filho”, conforme a seguinte estrutura:

Opções de Chave Estrangeira no SQL Server



Códigos SQL para criar o banco de teste e as tabelas:

Vamos criar um banco de dados para os nossos testes e criar as tabelas abaixo:

```
CREATE TABLE Pai (  
  ID_Pai SMALLINT PRIMARY KEY,  
  Nome_Pai VARCHAR(50)  
)  
  
CREATE TABLE Filho (  
  ID_Filho SMALLINT IDENTITY PRIMARY KEY,  
  Nome_Filho VARCHAR(50),  
  ID_Pai SMALLINT,  
  CONSTRAINT fk_id_pai FOREIGN KEY (ID_Pai)  
  REFERENCES Pai(ID_Pai)  
  ON DELETE CASCADE  
)
```

Carregando dados de teste nas tabelas:

```
INSERT INTO Pai
```

```
VALUES (1,'João'), (2,'Mário'), (3,'Renato'), (4,'Emerson'), (5,'André');
```

```
INSERT INTO Filho (Nome_Filho, ID_Pai)
```

```
VALUES ('João',1), ('Mário',1), ('Renato',3), ('Emerson',4), ('André',3);
```

Consultando os dados carregados:

```
SELECT Pai.ID_Pai, Pai.Nome_Pai, Filho.ID_Filho, Filho.Nome_Filho  
FROM Filho  
INNER JOIN Pai  
ON Filho.ID_Pai = Pai.ID_Pai;
```

| ID_Pai | Nome_Pai | ID_Filho | Nome_Filho |
|--------|----------|----------|------------|
| 1 | João | 1 | João |
| 1 | João | 2 | Mário |
| 3 | Renato | 3 | Renato |
| 4 | Emerson | 4 | Emerson |
| 3 | Renato | 5 | André |

ON DELETE CASCADE

Vamos testar agora a exclusão de um filho:

```
DELETE FROM Filho
```

```
WHERE Nome_Filho = 'Renato';
```

Ao excluirmos o filho Renato, seu pai, que também se chama Renato, continuará a existir na tabela de pais:

```
SELECT Nome_Pai, Nome_Filho
```

```
FROM Filho
```

```
INNER JOIN Pai
```

```
ON Filho.ID_Pai = Pai.ID_Pai;
```

| Nome_Pai | Nome_Filho |
|----------|------------|
| João | João |
| João | Mário |
| Emerson | Emerson |
| Renato | André |

ON DELETE CASCADE

Agora vamos testar a cláusula **ON DELETE CASCADE**. Vamos excluir o Pai Renato da tabela de pais. Neste caso, a exclusão deverá se propagar para a tabela de filhos, eliminando o registro do filho relacionado, que no caso é o André:

```
DELETE FROM Pai  
WHERE Nome_Pai = 'Renato';
```

Verificando a exclusão do pai:

```
SELECT Nome_Pai, Nome_Filho  
FROM Filho  
INNER JOIN Pai  
ON Filho.ID_Pai = Pai.ID_Pai;
```

| Nome_Pai | Nome_Filho |
|----------|------------|
| João | João |
| João | Mário |
| Emerson | Emerson |

ON DELETE CASCADE

Verificando a exclusão cascadeada do filho:

```
SELECT * FROM Filho;
```

| ID_Filho | Nome_Filho | ID_Pai |
|----------|------------|--------|
| 1 | João | 1 |
| 2 | Mário | 1 |
| 4 | Emerson | 4 |

Agora sobraram apenas os filhos João, Mário e Emerson; Já o André, que era filho do Renato, foi excluído automaticamente após eliminarmos o registro de seu pai da tabela de pais, devido à cláusula ON DELETE CASCADE.

Exemplo com SET NULL

Suponha que, ao excluir um pai do banco de dados, em vez de excluir imediatamente seus filhos (cascateamento) nós queiramos manter esses registros, e o campo de ID_Pai da tabela de filhos passe então a conter um valor NULL (“filhos órfãos”).

Exemplo com SET NULL

Neste caso, a tabela de filhos deve ser criada da maneira mostrada a seguir, substituindo a cláusula ON DELETE CASCADE por ON DELETE SET NULL:

Vamos excluir as tabelas:

```
DROP TABLE Filho;
```

```
DROP TABLE Pai;
```

Exemplo com SET NULL

```
CREATE TABLE Pai (  
  ID_Pai SMALLINT PRIMARY KEY,  
  Nome_Pai VARCHAR(50)  
)  
  
CREATE TABLE Filho (  
  ID_Filho SMALLINT IDENTITY PRIMARY KEY,  
  Nome_Filho VARCHAR(50),  
  ID_Pai SMALLINT,  
  CONSTRAINT fk_id_pai FOREIGN KEY (ID_Pai)  
    REFERENCES Pai(ID_Pai)  
    ON DELETE SET NULL  
)
```

Exemplo com SET NULL

INSERT INTO Pai

VALUES (1,'João'), (2,'Mário'), (3,'Renato'), (4,'Emerson'), (5,'André');

INSERT INTO Filho (Nome_Filho, ID_Pai)

VALUES ('João',1), ('Mário',1), ('Renato',3), ('Emerson',4), ('André',3);

Após recriar a tabela de filhos e preencher novamente, com a nova opção de chave estrangeira, vamos realizar um teste excluindo um dos pais da tabela de pais – por exemplo, novamente o Renato:

DELETE FROM Pai

WHERE Nome_Pai = 'Renato';

SELECT * FROM Filho;

| ID_Filho | Nome_Filho | ID_Pai |
|----------|------------|--------|
| 1 | João | 1 |
| 2 | Mário | 1 |
| 3 | Renato | NULL |
| 4 | Emerson | 4 |
| 5 | André | NULL |

Exemplo com NO ACTION

Vamos excluir as tabelas:

DROP TABLE Filho;

DROP TABLE Pai;

Exemplo com NO ACTION

```
CREATE TABLE Pai (  
  ID_Pai SMALLINT PRIMARY KEY,  
  Nome_Pai VARCHAR(50)  
)  
  
CREATE TABLE Filho (  
  ID_Filho SMALLINT IDENTITY PRIMARY KEY,  
  Nome_Filho VARCHAR(50),  
  ID_Pai SMALLINT,  
  CONSTRAINT fk_id_pai FOREIGN KEY (ID_Pai)  
    REFERENCES Pai(ID_Pai)  
    ON DELETE NO ACTION  
)
```

Exemplo com NO ACTION

INSERT INTO Pai

VALUES (1,'João'), (2,'Mário'), (3,'Renato'), (4,'Emerson'), (5,'André');

INSERT INTO Filho (Nome_Filho, ID_Pai)

VALUES ('João',1), ('Mário',1), ('Renato',3), ('Emerson',4), ('André',3);

DELETE FROM Pai

where Nome_Pai = 'Renato';

O sistema não irá aceitar apagar o pai devido a restrição criada.

CREATE TABLE

Para criar tabelas em um banco de dados, usamos a declaração **CREATE TABLE**.

CREATE TABLE

Veja a sintaxe:

CREATE TABLE [IF NOT EXISTS]

nome_tabela (

coluna tipo_dados constraints

coluna tipo_dados constraints

coluna tipo_dados constraints

...)

CREATE TABLE

USE Biblioteca;

```
CREATE TABLE tbl_livro (  
  ID_Livro SMALLINT IDENTITY PRIMARY KEY,  
  Nome_Livro VARCHAR(70) NOT NULL,  
  ISBN13 CHAR(13),  
  ISBN10 CHAR(10),  
  ID_Categoria SMALLINT,  
  ID_Autor SMALLINT NOT NULL,  
  ID_Editora SMALLINT,  
  Data_Pub DATE NOT NULL,  
  Preco_Livro DECIMAL(6,2) NOT NULL  
);
```

CREATE TABLE

criar uma tabela para armazenar os dados dos autores:

```
CREATE TABLE tbl_autores (  
  ID_Autor SMALLINT,  
  Nome_Autor VARCHAR(40) NOT NULL,  
  Sobrenome_Autor VARCHAR(60)  
  CONSTRAINT pk_ID_Autor PRIMARY KEY (ID_Autor)  
);
```

CREATE TABLE

Uma para armazenar as categorias de livros:

```
CREATE TABLE tbl_categorias (  
ID_Categoria SMALLINT PRIMARY KEY,  
Categoria VARCHAR(30) NOT NULL  
)
```

CREATE TABLE

Por fim, criamos uma tabela para armazenar os dados das editoras:

```
CREATE TABLE tbl_editoras (  
ID_Editora SMALLINT PRIMARY KEY IDENTITY,  
Nome_Editora VARCHAR(50) NOT NULL  
)
```

As tabelas criadas ainda não estão relacionadas entre si. Mostraremos como realizar essa operação nas próximas lições.

AUTO_INCREMENT SQL SERVER

- ◎ O auto incremento permite que um número único seja gerado automaticamente quando um novo registro é inserido em uma tabela.
- ◎ Em SQL SERVER usamos a palavra chave **IDENTITY** (“identidade”) para denotar o auto incremento em uma coluna, cujo valor inicial padrão é 1, e se incrementa também em 1.
- ◎ Para que o valor de IDENTITY inicie, por exemplo em 100, e se incremente de 2 em 2, use IDENTITY(100,2)

AUTO_INCREMENT SQL SERVER

- ⦿ Ao inserir valores na tabela, não é necessário especificar o valor para a coluna de auto-incremento.
- ⦿ Não é possível alterar uma coluna existente para configurar IDENTITY.
- ⦿ Se necessário, crie uma nova tabela com IDENTITY e exclua a atual.
- ⦿ Só é permitido usar uma coluna de identidade por tabela.

Auto Incremento – Exemplo

```
CREATE TABLE tbl_teste_identidade  
(ID_Testes SMALLINT PRIMARY KEY IDENTITY(10,2),  
valor SMALLINT NOT NULL);
```


Auto Incremento – Exemplo

```
INSERT INTO tbl_teste_identidade (valor) VALUES (10);  
INSERT INTO tbl_teste_identidade (valor) VALUES (20);  
INSERT INTO tbl_teste_identidade (valor) VALUES (30);  
INSERT INTO tbl_teste_identidade (valor) VALUES (40);  
SELECT * FROM tbl_teste_identidade;
```

Alterar o próximo valor no campo de Identidade

Para incremento atual

```
SELECT IDENT_CURRENT('tbl_teste_identidade');
```

Para alterar o valor de identidade do próximo registro a ser armazenado em uma tabela, use o comando a seguir:

```
DBCC CHECKIDENT (nome_tabela, RESEED, valor_semente)
```

Exemplo. Usar o valor 151 a partir do próximo registro:

```
DBCC CHECKIDENT (tbl_teste_identidade, RESEED, 149);
```

```
INSERT INTO tbl_teste_identidade (valor) VALUES (70);
```

```
INSERT INTO tbl_teste_identidade (valor) VALUES (80);
```

```
INSERT INTO tbl_teste_identidade (valor) VALUES (90);
```

```
SELECT * FROM tbl_teste_identidade;
```