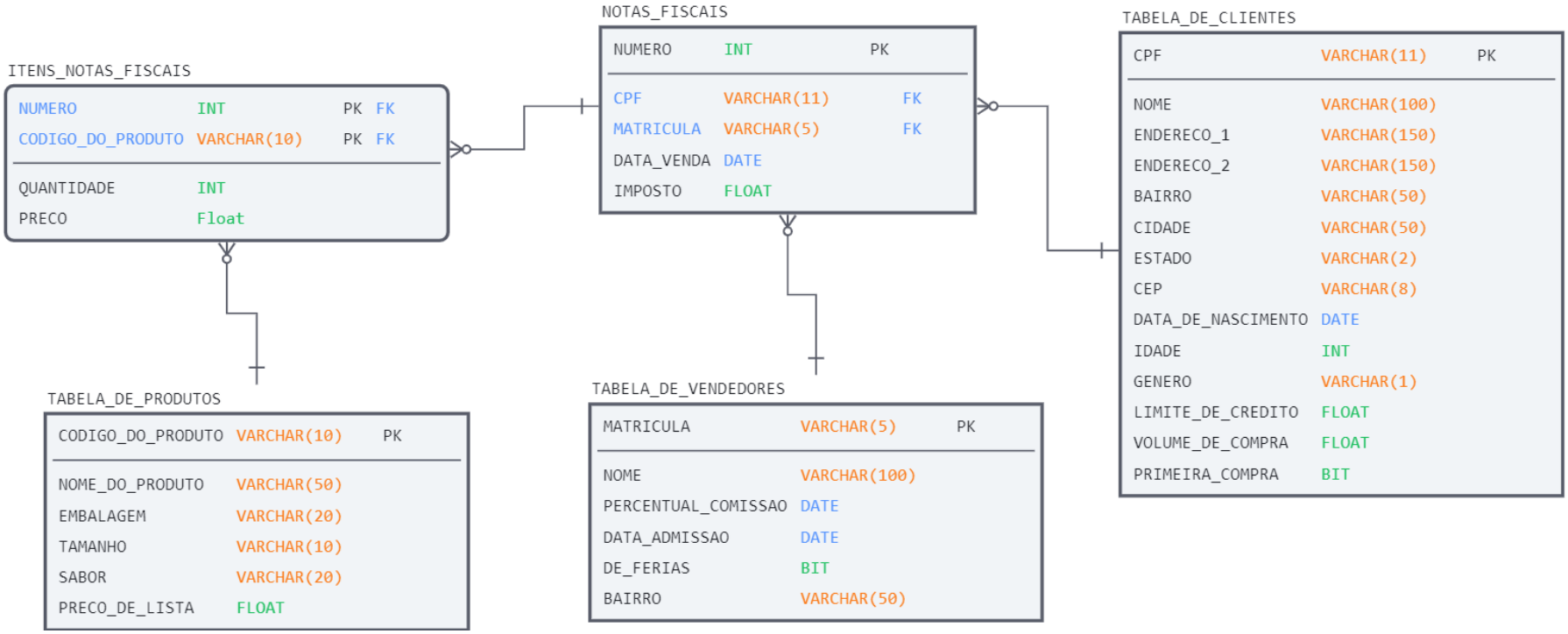
Aula 8 - Estrutura de comandos

Docupedia Export

Author:Goncalves Donathan (SO/OPM-TS21-BR)
Date:24-Jul-2024 16:20

Table of Contents

1 INSERÇÃO DE NOVOS DADOS



2 TRIGGER

Procedimento realizado automaticamente sempre que ocorre um evento especial no banco de dados.

Operações em que isso ocorre:

- **INSERÇÃO**
- **EXCLUSÃO**
- **ATUALIZAR**

Existem dois tipos de triggers:
Triggers DDL e Triggers DML

Você lembra a diferença entre DDL e DML?
Diferença
Triggers DDL (Data Definition Language)
Eventos que **alteram** a **estrutura** (como criar, modificar ou deletar uma tabela) ou em **determinados** eventos **relacionados** ao **servidor**, como alterações de **segurança** ou atualização de eventos **estatísticos**.

Triggers DML (Data Manipulation Language)
Mais utilizadas. Evento de disparo é uma **declaração** de **modificação** de **dados**.

Podem ser executados em 3 momentos diferentes:

- **FOR** - O gatilho é disparado **junto** da ação.
- **AFTER** - O disparo é realizado **após** a ação ser concluída.
- **INSTEAD OF** - Seja executado **no lugar** da ação que o gerou.

Vantagens:

- Impedir transações inválidas (Integridade, Segurança).
- Registro de eventos que ocorreram.
- Integridade referencial.

Como Utilizar:

```
CREATE  TRIGGER  [Nome_Trigger] -- É o nome definido pelo usuário para o novo trigger
ON  [Nome_tabela] -- É a tabela à qual o trigger se aplica.
AFTER DELETE
```

```
AS
BEGIN
    -- É possível declarar variáveis,
    -- inserir, excluir ou alterar dados em outras tabelas.
END
```

2.1 Exercício 1

Crie uma tabela log para armazenar todas as ocorrências de inserção que ocorrer na tabela Pessoa.
Crie uma trigger que fará isso automaticamente.
Dica: Utilize INSERTED

```
-- Criando a tabela que será populada pela Trigger
CREATE TABLE Log (
Data DATETIME,
Operacao VARCHAR(50),  -- "Inserção"
Observacao VARCHAR(255) -- "Inserido Pessoa (Nome)"
PRIMARY KEY (Data, Operacao))  -- Os dois juntos são a Primary Key
```

Trigger

```
-- Criando Trigger para inserir os dados automaticamente
CREATE TRIGGER tgLog ON Tabela_De_Clientes
FOR INSERT AS
BEGIN
    DECLARE -- Cria variáveis
    @Data DATETIME,
    @Operacao VARCHAR(50),
    @Observacao VARCHAR(255)
       -- Atribui valores às variáveis
    SELECT @Data = GETDATE(), @Operacao = 'Inserção', @Observacao = CONCAT('Inserido Cliente ', Nome) FROM INSERTED
```

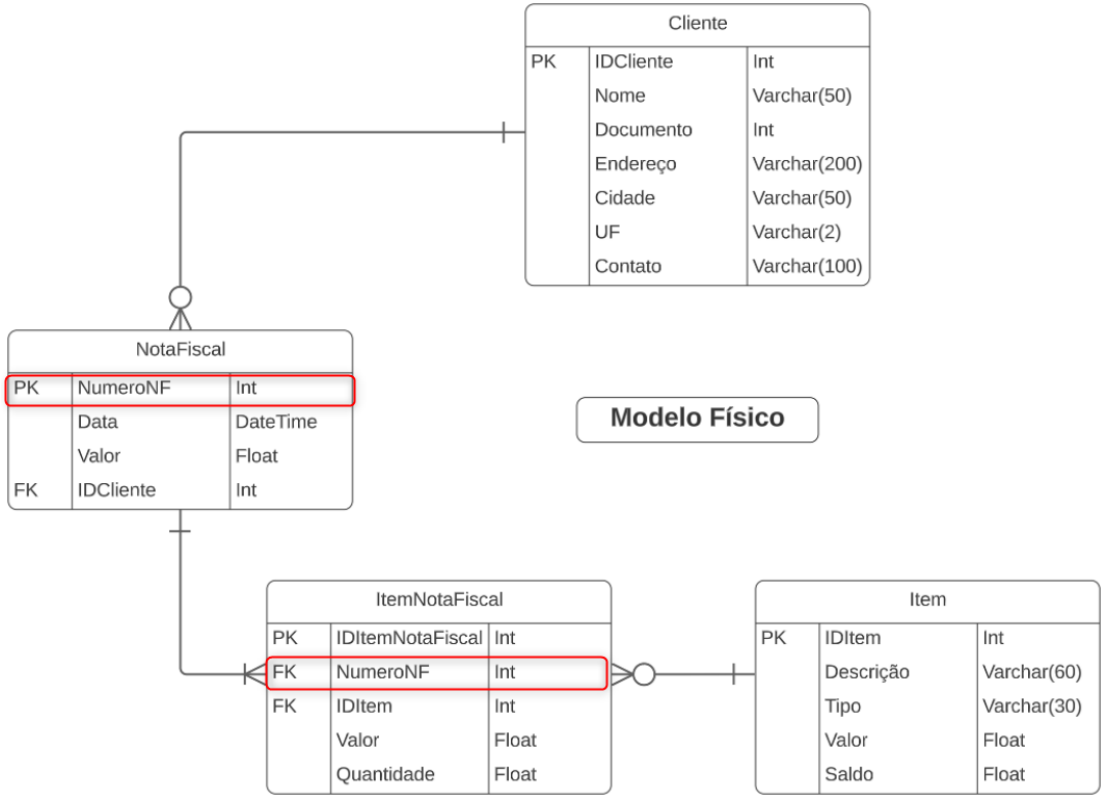
```
        -- Faz uma inserção em outra tabela utilizando os valores das variáveis
    INSERT INTO Log VALUES(@Data, @Operacao, @Observacao)
END
```

Agora, insira alguns dados na tabela Cliente e veja o resultado na Log

---

2.2 Exercício 2

Em banco de dados, existe algo que é chamado de **Integridade Referencial**.
Ou seja, **não** posso **excluir** um **dado** que está sendo **referenciado**(FK) em outra tabela.



Exemplo: Não é possível apagar uma Nota Fiscal sem antes apagar seus itens.

Crie uma trigger que ao excluir um dado, ele irá excluir primeiro a referencia desse dado, e depois o dado que queremos.
Basicamente "burlar" essa integridade do banco.

Trigger

```
CREATE TRIGGER tgApagaItens ON Notas_Fiscais
INSTEAD OF DELETE AS
BEGIN
    DECLARE
    @IDNotaFiscal VARCHAR(12)
```

```
    SELECT @IDNotaFiscal = Numero FROM DELETED
    DELETE FROM Itens_Notas_Fiscais WHERE Numero = @IDNotaFiscal
    DELETE FROM Notas_Fiscais WHERE Numero = @IDNotaFiscal
END
```

Agora, tente excluir alguns dados que possuam essa integridade referencial

3 STORED PROCEDURE

Um **conjunto** de **comandos** SQL, que ficam **salvos** no **servidor** e podem ser **executados** de **uma só vez**, como em uma função.
Além disso, é possível "esconder" seu código por meio de **encriptação**, assim é possível **preservar a inteligência** que há por trás dos comandos.

STORED PROCEDURE Vs **FUNÇÃO**

- A FUNÇÃO retorna um valor e pode ser usada em uma consulta SQL como parte de uma expressão.
- Enquanto a STORED PROCEDURE não retorna um valor e é usada para executar ações complexas no banco de dados.

Vantagens:

- Reduzir tráfego da rede.
- Melhor performance do que as função.
- Diminui os riscos.
- Rotinas de processamento.

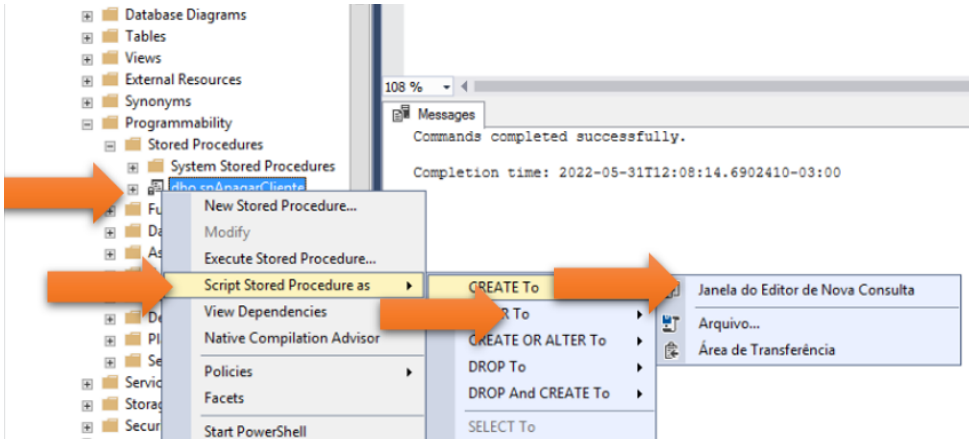
Geralmente utilizada para ações como INSERÇÕES, ou EXCLUSÕES, ou para executar lógica de negócios complexa.

Funcionamento de uma Procedure

```
CREATE PROCEDURE NomeDaStoredProc
    @Parametro1 TipoDeDados,
    @Parametro2 TipoDeDados
AS
BEGIN
    -- Corpo da Stored Procedure (comandos SQL e lógica de programação)
END
```

Chamando

```
EXEC NomeDaStoredProc @Parametro1 = Valor1, @Parametro2 = Valor2
-- Se você desejar, pode passar somente o valor sem o @Parametro
```



O SQL Server tem uma **função** de **gerar o código** para **criar**, **alterar**, ou **recriar** objetos por meio do objeto original.
Porém, em alguns casos você pode querer **preservar** (**esconder**) a **inteligência** de seu código SQL. Para isso é possível **criptografar** o código fonte.
Importante:
Uma vez criptografado, o código só pode ser alterado caso você tenha salvo em outro lugar o código fonte original!

Funcionamento de uma Procedure

```
CREATE PROCEDURE NomeDaStoredProc
WITH ENCRYPTION
    @Parametro1 TipoDeDados,
    @Parametro2 TipoDeDados
AS
BEGIN
    -- Corpo da Stored Procedure (comandos SQL e lógica de programação)
END
```

3.1 Exercício 3

CRIE uma **STORED PROCEDURE** que passado o **ID** de determinado **CLIENTE** como parâmetro deve:

- Excluir o Cliente
- Excluir todas as NotasFiscais desse Cliente
- Excluir todos os dados da tabela ItemNotaFiscal daquele Cliente

Resolução

Criar Procedure

```
CREATE PROCEDURE sp_ApagarCliente @IDCliente VARCHAR(12)
AS
BEGIN
    DELETE FROM Itens_Notas_Fiscais WHERE Numero IN
    (
        SELECT Numero FROM Notas_Fiscais WHERE CPF = @IDCliente
    )

    DELETE FROM Notas_Fiscais WHERE CPF = @IDCliente

    DELETE FROM Tabela_De_Clientes WHERE CPF = @IDCliente
END
```

Executar Procedure

```
EXEC sp_ApagarCliente 123
```

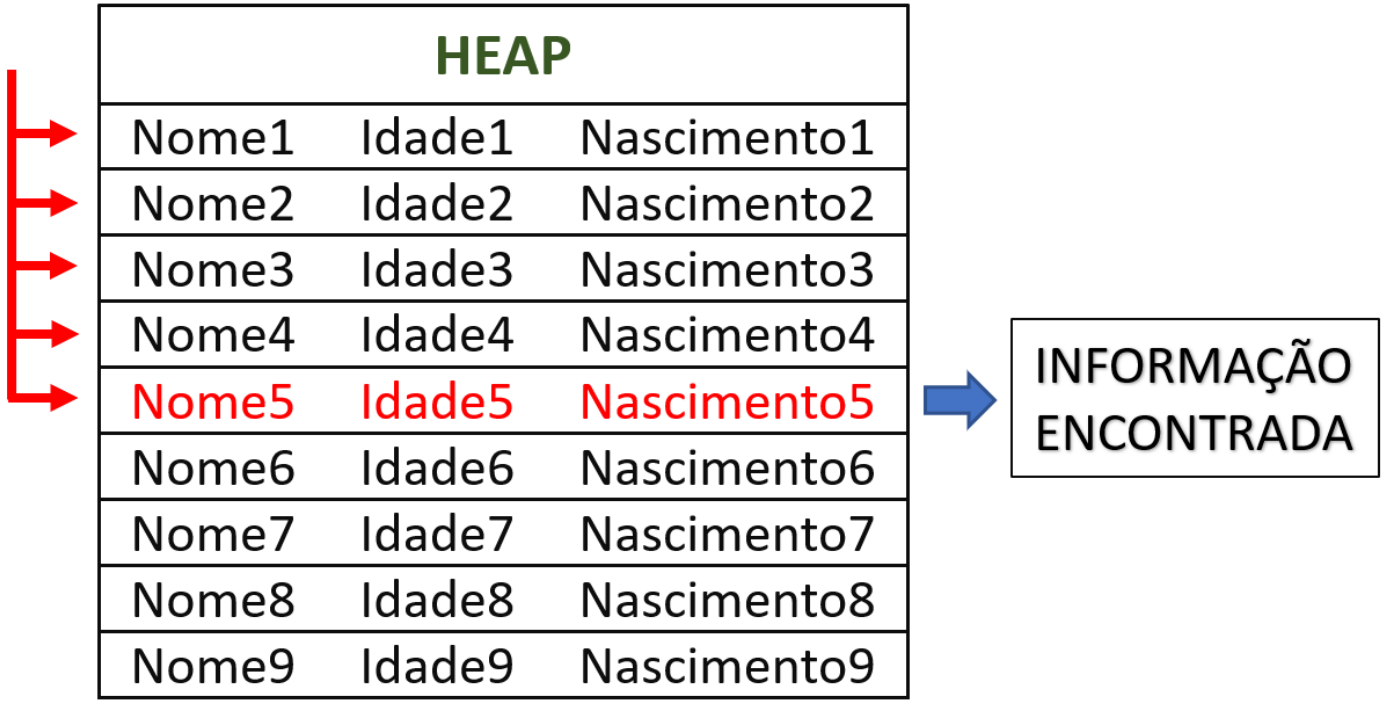
4 Índices (Index)

Utilizados para facilitar a busca de informações em uma tabela com o menor número possível de operações de leituras.
Tornar a busca mais rápida e eficiente.

Veremos o básico, pois é um assunto muito extenso.

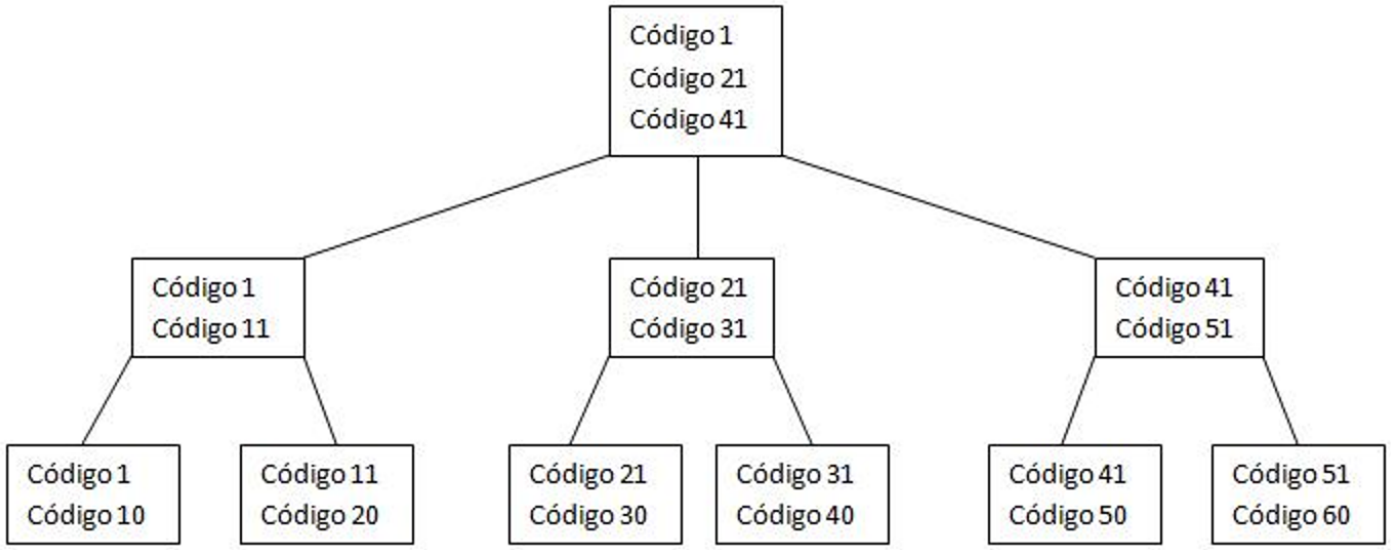
4.1 Busca sem utilizar ÍNDICES

É **necessário** a busca em **todos** os **dados** até **encontrar** a informação desejada.



4.2 ÍNDICE para um campo do tipo INTEIRO

Para **construir** os níveis **raiz** e **intermediário** pega-se o **primeiro valor** de cada **página** do **nível abaixo** junto com o ponteiro da página de onde o valor de dados veio.



4.2.1 Existem DOIS tipos de ÍNDICES no SQL Server

Clusterizados (clustered)

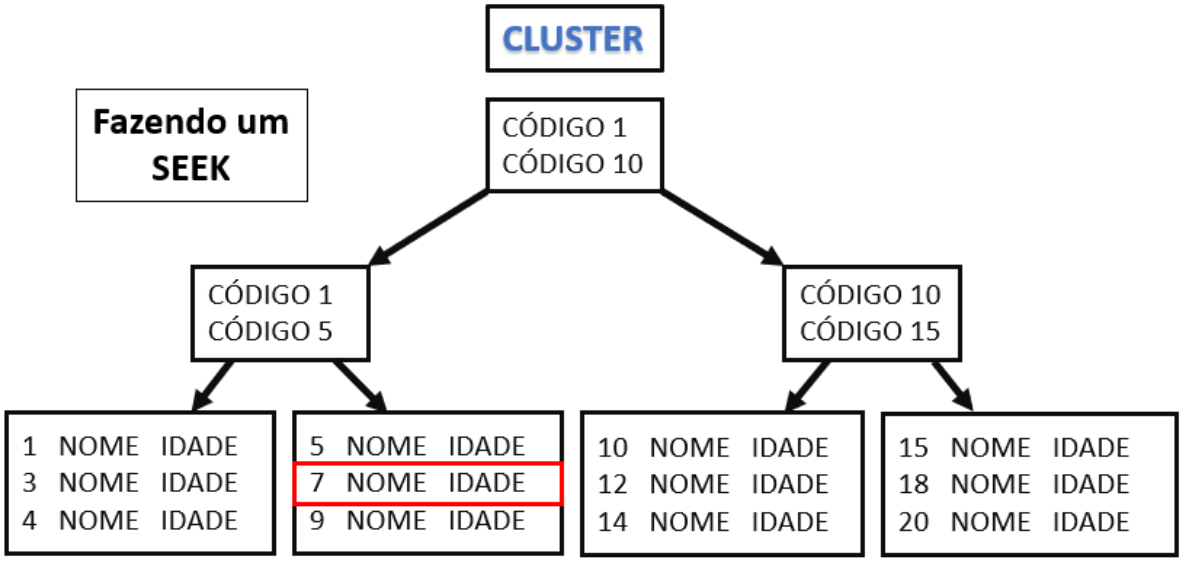
- Possível criar **apenas um** por tabela (Geralmente a **chave primária**).
- Se **tem chave primária**, a **árvore** será **montada** por **essa coluna**.

Não Clusterizados (nonclustered)

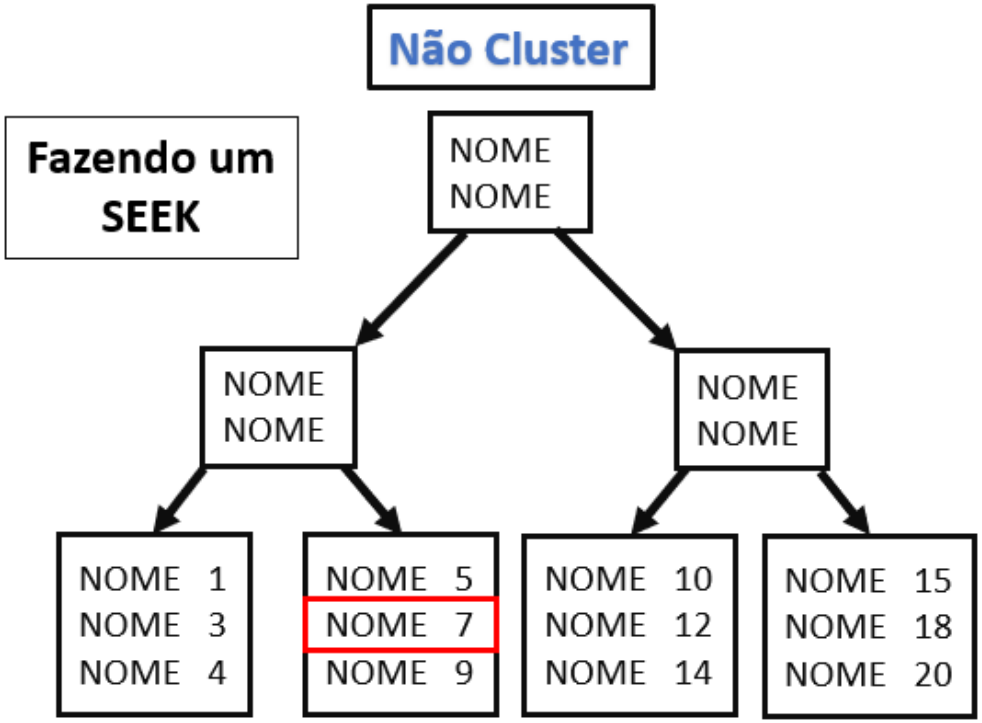
- Possível criar **até 1000** por tabela, com no máximo **900 bytes** cada e **16 colunas**.
- Pode ser em **qualquer coluna** (Indicado utilizar a **coluna** que **mais** costuma ser **utilizada** para se procurar os dados).

IMPORTANTE
No **ÍNDICE CLUSTER** estão **todas as colunas** no **nível folha**.
Enquanto **apenas** a **coluna** com **índice** estará nos **outros nós**.

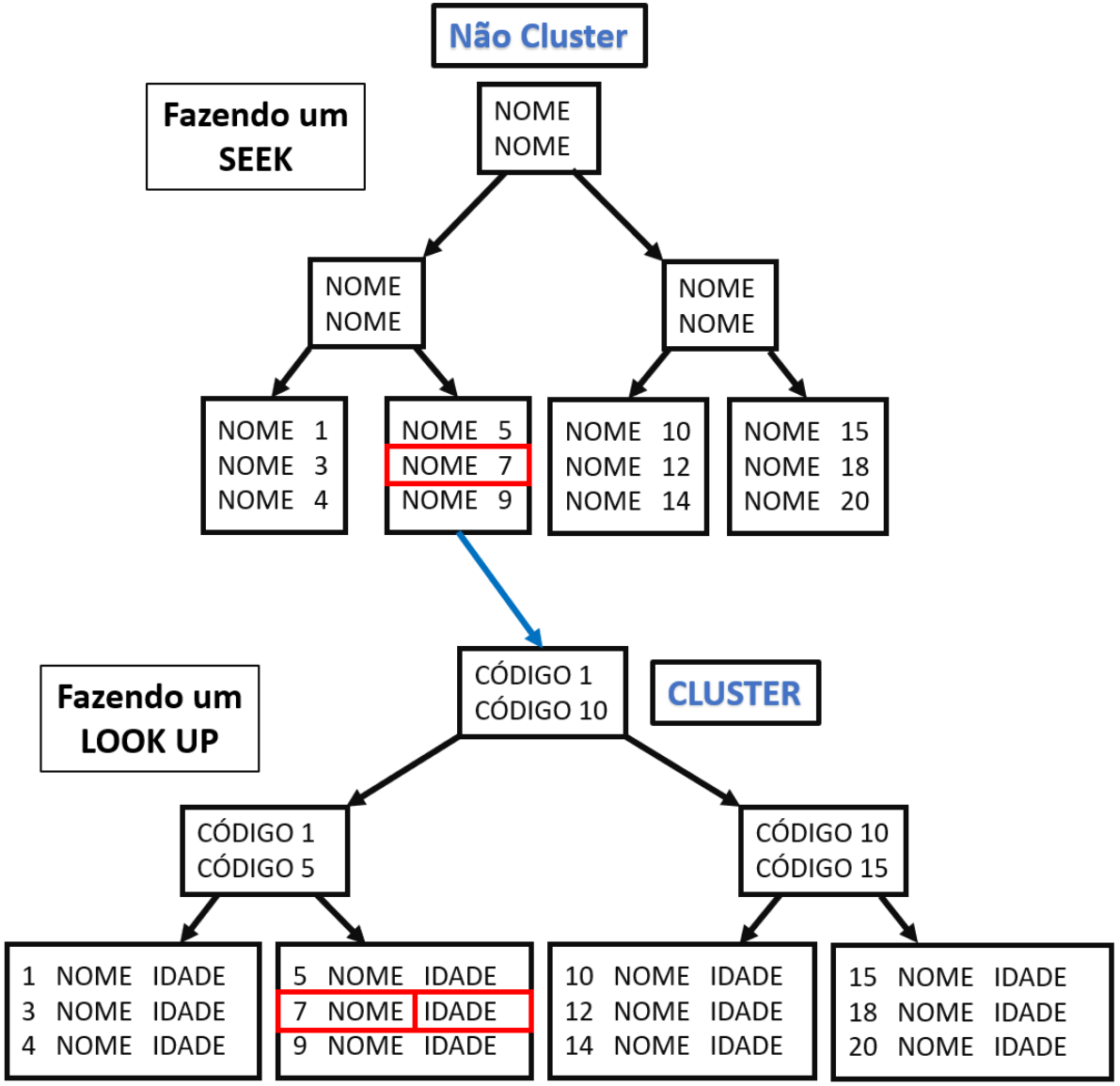
Procurando dados por um **ÍNDICE CLUSTER**



Procurando dados por um **ÍNDICE NÃO CLUSTER**



E quando eu quero saber a **IDADE** através de um **ÍNDICE NÃO CLUSTER**?

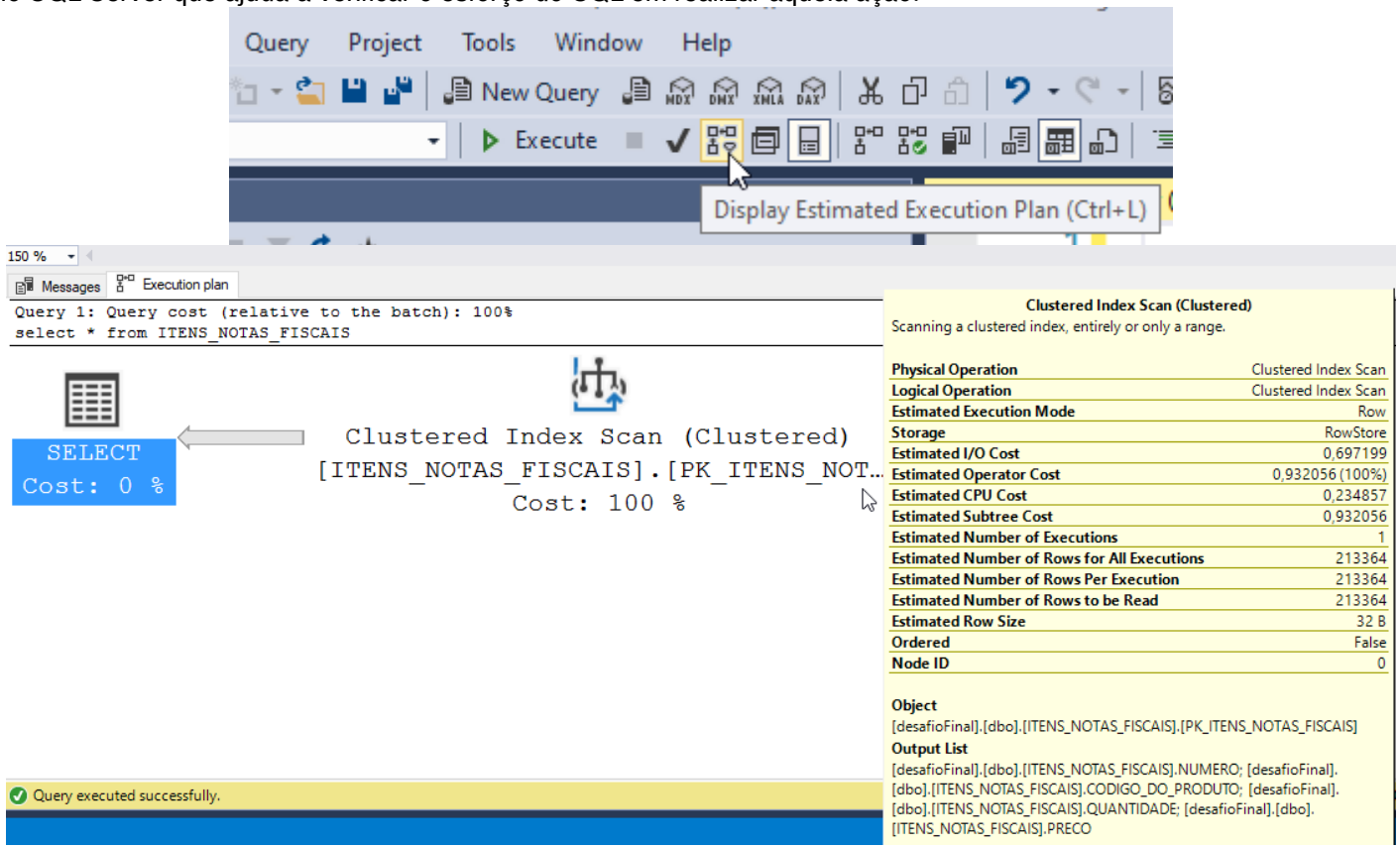


Ele vai pegar o valor do ÍNDICE CLUSTERIZADO e irá procura-lo na ÁRVORE CLUSTER

DESVANTAGENS em usar **ÍNDICES** em **SQL**

- Espaço em disco.
- Tempo em atualizar as tabelas (Modificar as tabelas).
- Manutenção do Banco.
- Tempo na inserção de novos dados.

Existe uma ferramenta no SQL Server que ajuda a verificar o esforço do SQL em realizar aquela ação.



Criando um ÍNDICE

```
CREATE INDEX IDX_NOME_INDEX ON NOME_TABELA(NOME_COLUNA)
```

Excluir INDEX

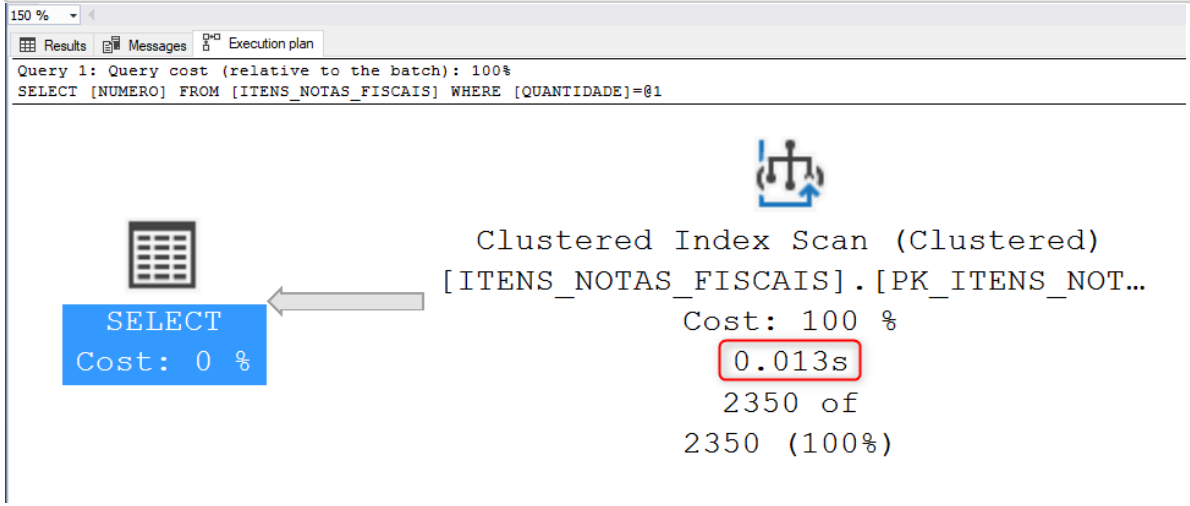
```
DROP INDEX NOME_TABELA.IDX_NOME_INDEX
```

Teste no **novo DataBase** e **encontre** um **INDEX** que **melhore** a **performance** da busca.
Cuidado, pois **dependendo** do **INDEX** que você criar, **pode aumentar o tempo** de **busca** de determinados dados.

Exemplo

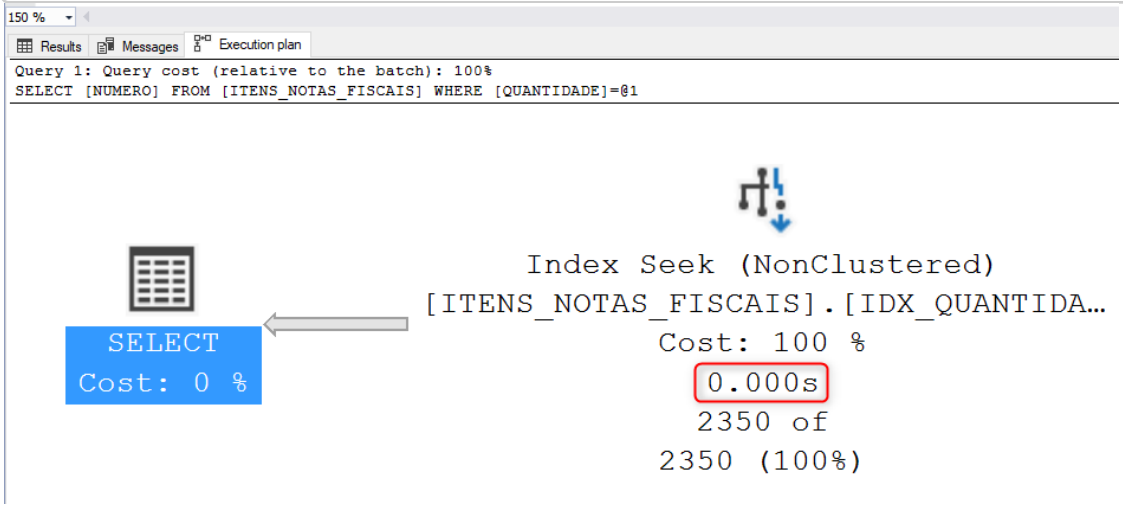
Busca sem INDEX

```
SELECT NUMERO FROM ITENS_NOTAS_FISCAIS
WHERE QUANTIDADE = 60
```

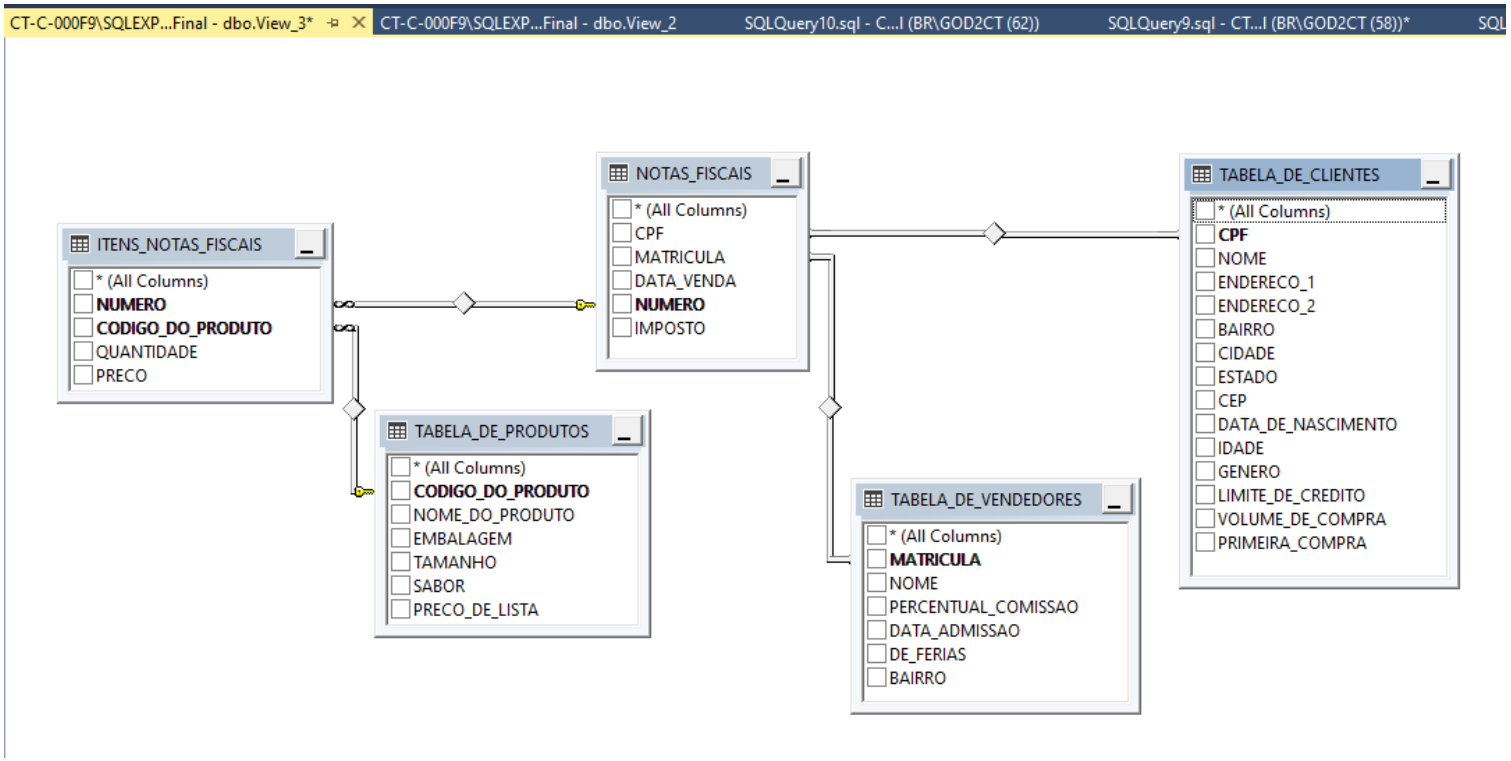


Criando INDEX

```
CREATE INDEX IDX_QUANTIDADE ON ITENS_NOTAS_FISCAIS(QUANTIDADE)
```



É possível ver o diagrama das Tabelas no Database



Para isso basta seguir os passos abaixo: