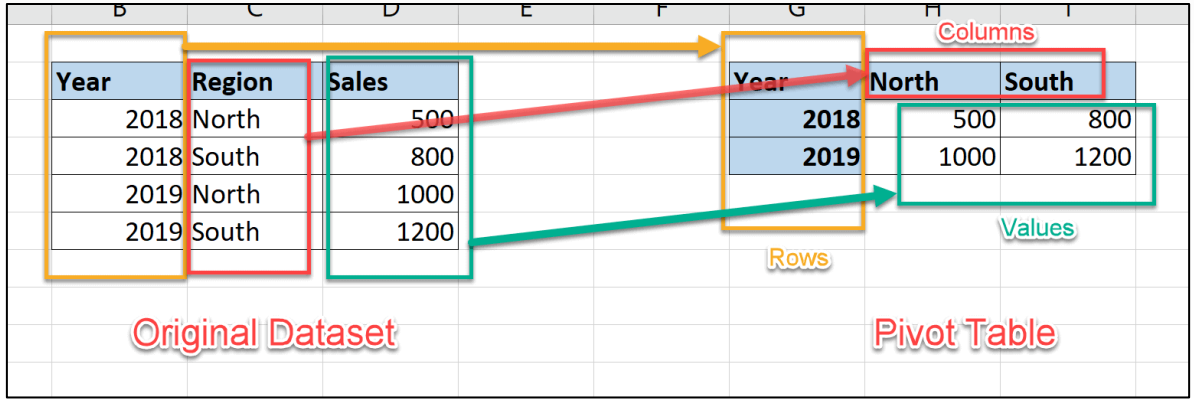
Banco de Dados

Módulo Intermediário

Tópicos Avançados
Manipulação de Dados

Pivoting

- ***Operador PIVOT***
  - *O **conceito** de **transformação** (**pivoting**) em **dados** refere-se a **reestruturá-los** em **outro formato***
  - *Para essa finalidade, no SQL Server, estão **disponíveis dois operadores** (**PIVOT** e **UNPIVOT**)*



Pivoting

- ***Operador PIVOT***
  - ***Imagine*** *que os* ***dados estejam armazenados*** *em um* ***formato semelhante*** *ao* ***apresentado*** *na* ***Tabela 1***

IDPedidoDeCompra	IDCliente	IDVendedor
43659	29825	279
43660	29672	279
43661	29734	282
43662	29994	276
43663	29565	280
43664	29898	283

Tabela 1 – Dados linha por linha

Pivoting

- ***Operador PIVOT***
  - ***Cada linha** da **tabela representa** um **pedido** **realizado** por um **cliente** e o **vendedor** que o **recebeu***
  - ***Em vez** de **exibir** essas **informações** **linha** **por** **linha**, podemos usar o **operador** **PIVOT** para **retornar** o **número** de **pedidos feitos** por **cada cliente**, **agrupados** por **vendedor***

IDVendedor	29825	29672	29734	29994	29565	29898
279	5	1	0	2	1	0
282	6	2	1	3	2	1
275	7	3	2	4	3	2
280	2	5	5	2	1	1
283	3	6	6	3	2	2

Tabela 2 – Transformação de dados

Pivoting

- ***Operador PIVOT***
  - *A **primeira coluna** da **tabela representa** o **vendedor** e **cada coluna subsequente representa** um **cliente** e **quantas vendas foram realizadas** por ele*
  - *A **primeira linha** de **cada coluna**, além da **coluna** que **contém IDVendedor**, **contém** um **IDCliente***

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 1***
    - ***Esse fragmento representa** a **base** da **transformação**, **retornando** os **dados linha** por **linha***

```
USE MyAdventureWorks;

SELECT SalesOrderID, CustomerID, SalesPersonID
FROM Sales.SalesOrderHeader
WHERE SalesPersonID IS NOT NULL;
```

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 2***
    - ***Esse outro fragmento representa*** *como os* ***dados serão retornados***

```
SELECT
   SalesPersonID, [29486] AS Cust1, [29487] AS Cust2,
   [29488] AS Cust3, [29491] AS Cust4,
   [29492] AS Cust5, [29512] AS Cust6
FROM
(
```

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 3***
    - ***Fragmento responsável** por **especificar** o **operador PIVOT**, o que **será agregado** e **quais clientes serão incluídos** no **conjunto** de **resultados***

```
) AS p
PIVOT
( COUNT(SalesOrderID)
  FOR CustomerID IN ([29486], [29487], [29488], [29491], [29492], [29512])
) AS pvt
ORDER BY SalesPersonID;
```

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 4***
    - ***Consulta completa*** *exemplificando o* ***uso*** *do* ***operador PIVOT***

```
SELECT
   SalesPersonID, [29486] AS Cust1, [29487] AS Cust2,
   [29488] AS Cust3, [29491] AS Cust4,
   [29492] AS Cust5, [29512] AS Cust6
FROM
( SELECT SalesOrderID, CustomerID, SalesPersonID
  FROM Sales.SalesOrderHeader
  WHERE SalesPersonID IS NOT NULL
) AS p
PIVOT
( COUNT(SalesOrderID)
  FOR CustomerID IN ([29486], [29487], [29488], [29491], [29492], [29512])
) AS pvt
ORDER BY SalesPersonID;
```

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 5***
    - ***Tabela resultante***

	SalesPersonID	Cust1	Cust2	Cust3	Cust4	Cust5	Cust6
1	274	0	0	0	1	0	0
2	275	6	6	0	3	5	0
3	276	0	0	0	0	0	0
4	277	6	5	0	0	1	0
5	278	0	0	0	0	0	0
6	279	0	0	0	0	0	0
7	280	0	0	0	0	0	0
8	281	0	0	0	0	0	0
9	282	0	0	0	0	0	0
10	283	0	0	0	0	0	0
11	284	0	0	0	0	0	0
12	285	0	0	1	0	0	3
13	286	0	0	3	0	0	1
14	287	0	0	0	0	0	0
15	288	0	0	0	0	0	0
16	289	0	0	0	0	0	0
17	290	0	0	0	0	0	0

Pivoting

- ***Operador PIVOT***
  - ***Exemplo (01): - Parte 5***
    - ***Tabela resultante***

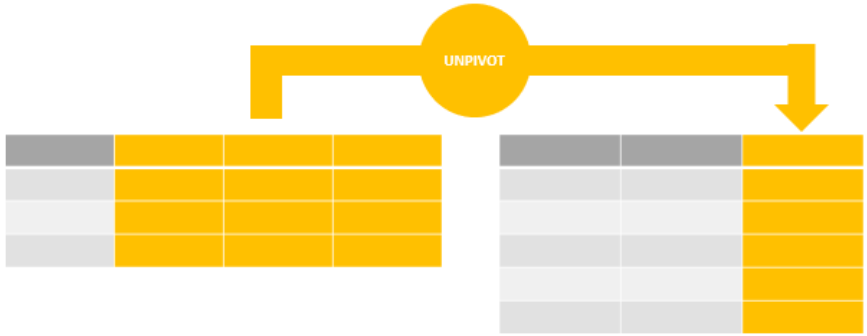
	SalesPersonID	Cust1	Cust2	Cust3	Cust4	Cust5	Cust6
1	274	0	0	0	1	0	0
2	275	6	6	0	3	5	0
3	276	0	0	0	0	0	0
4	277	6	5	0	0	1	0
5	278	0	0	0	0	0	0

Na lista transformada, cada linha se agrega para apresentar uma lista distinta de vendedores e a quantidade de vendas realizadas para cada cliente

13	286	0	0	3	0	0	1
14	287	0	0	0	0	0	0
15	288	0	0	0	0	0	0
16	289	0	0	0	0	0	0
17	290	0	0	0	0	0	0

Pivoting

- ***Operador UNPIVOT***
  - ***Se*** *os* ***dados*** *são* ***similares*** *com os* ***apresentados*** *anteriormente e, desejamos* ***visualizar*** *onde* ***cada linha represente uma venda*** *para o* ***cliente****, fazemos uso do* ***operador UNPIVOT*** *para* ***viabilizar*** *esse* ***tipo*** *de* ***transformação***
  - *O* ***conjunto*** *de* ***dados obtidos*** *seria* ***exatamente*** *como o* ***apresentado*** *na* ***Tabela 1***



Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (02): - Parte 1***
    - ***Apresentando*** *os* ***dados*** *que* ***precisam*** *ser* ***fornecidos*** *para* ***apresentar*** *um* ***exemplo prático*** *do* ***operador UNPIVOT***

```
USE MyAdventureWorks
GO
IF (OBJECT_ID('dbo.unPvt')) IS NOT NULL
   DROP TABLE dbo.unPvt
GO
CREATE TABLE dbo.unPvt
(
  SalesPersonID  INT,
  Cust1  INT,
  Cust2  INT,
  Cust3  INT,
  Cust4  INT,
  Cust5  INT,
  Cust6  INT
)
```

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (02): - Parte 2***
    - ***Apresentando*** *os* ***dados*** *que* ***precisam*** *ser* ***fornecidos*** *para* ***apresentar*** *um* ***exemplo prático*** *do* ***operador UNPIVOT***

```
GO
INSERT INTO dbo.unPvt
(
  SalesPersonID, Cust1, Cust2, Cust3, Cust4, Cust5, Cust6
)
VALUES
  (274, 5, 6, 4, 2, 6, 7),
  (275, 1, 7, 2, 3, 6, 8),
  (276, 0, 2, 8, 9, 6, 3),
  (277, 6, 3, 1, 7, 6, 1),
  (278, 5, 4, 9, 0, 2, 0),
  (279, 2, 1, 0, 1, 8, 9)
GO
```

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (02): - Parte 3***
    - ***Apresentando*** *os* ***dados*** *que* ***precisam*** *ser* ***fornecidos*** *para* ***apresentar*** *um* ***exemplo prático*** *do* ***operador UNPIVOT***

```
SELECT *
FROM dbo.unPvt;
```

	SalesPersonID	Cust1	Cust2	Cust3	Cust4	Cust5	Cust6
1	274	5	6	4	2	6	7
2	275	1	7	2	3	6	8
3	276	0	2	8	9	6	3
4	277	6	3	1	7	6	1
5	278	5	4	9	0	2	0
6	279	2	1	0	1	8	9

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (03): - Parte 1***
    - ***Fragmento responsável*** *pela* ***base*** *da* ***query*** *com* ***UNPIVOT*** *que* ***fornece*** *os* ***dados básicos***

```
USE MyAdventureWorks;

SELECT SalesPersonID, Cust1, Cust2, Cust3, Cust4, Cust5, Cust6
FROM unPvt;
```

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (03): - Parte 2***
    - ***Fragmento responsável*** *por* ***representar*** *os* ***dados*** *que* ***serão retornados*** *quando a* ***última query*** *for* ***executada***

```
USE MyAdventureWorks;

SELECT SalesPersonID, Customer, Sales
FROM
(
```

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (03): - Parte 3***
    - ***Fragmento** que **inclui** o **operador** **UNPIVOT** que, em vez de **agregar dados**, como **PIVOT**, **contém** uma **lista** de **clientes** que **serão incluídos** no **conjunto** de **resultados linha** por **linha***

```
) up
UNPIVOT
(
   Sales FOR Customer IN (Cust1, Cust2, Cust3, Cust4, Cust5, Cust6)
) AS unpvt;
GO
```

Pivoting

- ***Operador UNPIVOT***
  - ***Exemplo (03): - Parte 4***
    - ***Consulta completa*** *exemplificando o* ***uso*** *do* ***operador UNPIVOT***

```
USE MyAdventureWorks;
GO
SELECT SalesPersonID, Customer, Sales
FROM
(
  SELECT SalesPersonID, Cust1, Cust2, Cust3, Cust4, Cust5, Cust6
  FROM unPvt
) up
UNPIVOT
(
  Sales FOR Customer IN (Cust1, Cust2, Cust3, Cust4, Cust5, Cust6)
) AS unpvt;
GO
```

Paginação

- ***Paginando Dados***
  - *Uma **solicitação comum** de **aplicativo** é a **paginação** de **dados***
  - ***Em vez** de **retornar** o **conjunto** de **resultados inteiro**, muitas vezes é preferível ter **uma pequena lista**, **dividida** por uma **página** e **algum número** de **linhas***
  - *Os **desenvolvedores** e **DBA**s podiam **usar várias técnicas** para **simular** a **paginação**, mas com o **SQL Server** agora temos uma **verdadeira paginação nativa** no **lado** do **BD***

Paginação

- ***Paginando Dados***
  - *O **SQL Server introduziu** duas **palavras-chave** que **oferecem** uma **solução** de **paginação elegante** e **eficiente***
  - ***Usando OFFSET** e **FETCH** é possível **escrever uma única query** que **retorna dados uma página por vez** para um **aplicativo cliente** ou para o **usuário final***
  - ***OffSet:** denota **quantas linhas** devem ser **puladas, antes** que a **query comece** a **retornar linhas***
  - ***Fetch:** especifica **quantas linhas** devem ser **retornadas após** o **processamento** de **OFFSET***

Paginação

- ***Paginando Dados***
  - ***OFFSET*** *é* ***sinônimo*** *de* ***número*** *de* ***página*** *e* ***FETCH*** *é* ***sinônimo*** *de* ***número*** *de* ***linhas*** *a* ***serem exibidas*** *por* ***página***
  - ***Tanto OFFSET*** *como* ***FETCH*** *tem* ***mais alguns argumentos*** *que* ***devem*** *ser* ***incluídos*** *na* ***sintaxe***
  - ***Sintaxe:***

```
SELECT <lista de colunas>
FROM <nome da tabela>
ORDER BY <nome da coluna>
OFFSET <linhas começam em> ROWS
FETCH NEXT <número de linhas a retornar> ROW ONLY
```

Paginação

- ***Paginando Dados***
  - ***OFFSET*** *é* ***sinônimo*** *de* ***número*** *de* ***página*** *e* ***FETCH*** *é* ***sinônimo*** *de* ***número*** *de* ***linhas*** *a* ***serem exibidas*** *por* ***página***
  - ***Tanto OFFSET*** *como* ***FETCH*** *tem* ***mais alguns argumentos*** *que* ***devem*** *ser* ***incluídos*** *na* ***sintaxe***
  - ***Sintaxe:***

Com OFFSET, podemos fornecer um valor inteiro ou uma expressão especificando a linha inicial. A palavra-chave ROWS também deve ser incluída.
FETCH exige a palavra-chave NEXT, um inteiro ou uma expressão especificando o número de linhas a retornar e a palavra-chave ROWS

Paginação

- ***Paginando Dados***
  - ***Exemplo (04):***
    - ***Exibindo apenas 10 tuplas** a **partir** da **primeira tupla**, proveniente da tabela Product do **esquema** Production*

```
USE MyAdventureWorks;

SELECT
   ProductID,
   ProductNumber,
   Name AS ProductName,
   ListPrice
FROM Production.Product
ORDER BY ProductID
OFFSET 0 ROWS
FETCH NEXT 10 ROWS ONLY;
```

	ProductID	ProductNumber	ProductName	ListPrice
1	1	AR-5381	Adjustable Race	0,00
2	2	BA-8327	Bearing Ball	0,00
3	3	BE-2349	BB Ball Bearing	0,00
4	4	BE-2908	Headset Ball Bearings	0,00
5	316	BL-2036	Blade	0,00
6	317	CA-5965	LL Crankarm	0,00
7	318	CA-6738	ML Crankarm	0,00
8	319	CA-7457	HL Crankarm	0,00
9	320	CB-2903	Chainring Bolts	0,00
10	321	CN-6137	Chainring Nut	0,00

Paginação

- ***Paginando Dados***
  - *Um **requisito adicional** e, provavelmente o **mais importante**, é que **esse par deve ser precedido** por uma **cláusula ORDER BY***
  - *A **coluna especificada** na **cláusula ORDER BY determina** a **ordem** e **quais linhas** serão **retornadas***

Paginação

- ***Paginando Dados***
  - ***Exemplo (05):***
    - *A **consulta inicia** a **partir** da **10ª tupla**, ou seja, o **deslocamento** (offset) foi **alterado** para **10***

```
USE MyAdventureWorks;

SELECT
   ProductID,
   ProductNumber,
   Name AS ProductName,
   ListPrice
FROM Production.Product
ORDER BY ProductID
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY;
```

	ProductID	ProductNumber	ProductName	ListPrice
1	322	CR-7833	Chainring	0,00
2	323	CR-9981	Crown Race	0,00
3	324	CS-2812	Chain Stays	0,00
4	325	DC-8732	Decal 1	0,00
5	326	DC-9824	Decal 2	0,00
6	327	DT-2377	Down Tube	0,00
7	328	EC-M092	Mountain End Caps	0,00
8	329	EC-R098	Road End Caps	0,00
9	330	EC-T209	Touring End Caps	0,00
10	331	FE-3760	Fork End	0,00

Expressões

- ***Trabalhando com Expressões***
  - ***Muitas vezes** é **necessário combinar** os **valores** de **duas colunas** em **um único valor***
  - *Por exemplo, **concatenar** as **colunas** FirstName e LastName **produzindo** como **resultado** uma **única coluna***
  - ***Exemplo (06):***
    - ***Mesclando** os **valores** das **colunas** FirstName e LastName da **tabela** Person*

```
USE MyAdventureWorks;

SELECT
   FirstName + ' ' + LastName AS FullName
FROM Person.Person;
```

	FullName
1	Syed Abbas
2	Catherine Abel
3	Kim Abercrombie
4	Kim Abercrombie
5	Kim Abercrombie
6	Hazem Abolrous
7	Sam Abolrous
8	Humberto Acevedo
9	Gustavo Achong
10	Pilar Ackerman

Expressões

- ***Trabalhando com Expressões***
  - ***Exemplo (07):***
    - ***Aplicando** um **reajuste** de 5% aos **valores pertinentes** as **colunas** SubTotal e TaxAmt, ambas **constituintes** da **tabela** SalesOrderHeader*

```
USE MyAdventureWorks;

SELECT
   (SubTotal + TaxAmt) * 1.05 AS TotalDue
FROM Sales.SalesOrderHeader;
```

Variáveis

- ***Manipulando Variáveis***
  - *Durante a **recuperação** de **dados**, podemos **encontrar** uma **situação** na qual **precisamos armazenar** um **valor temporariamente**, para uso **posterior** em **nossa query***
  - *Pode ser um **valor** de uma **instrução SELECT** ou **um valor constante** que será **usado depois** em **uma query***
  - *Para **utilizar** uma **variável**, primeiro **precisamos declará-la**, na sequência, **prefixe** a **variável** com o **símbolo** de **arroba** (@)*
  - *Por fim, **especifique** o **tipo** de **dado** que será **armazenado** na **variável***

Variáveis

- ***Manipulando Variáveis***
  - ***Sintaxe:***

```
DECLARE @variavel INT
```

  - ***Uma vez declarada*** *a* ***variável****, podemos* ***atribuir valores*** *a ela*
  - ***Existem três métodos*** *para isso****:***
    - ***usar*** *a* ***palavra-chave SET*** *(****método preferido****)*
    - ***atribuir*** *um* ***valor utilizando*** *a* ***instrução SELECT***
    - ***atribuir*** *um* ***valor durante*** *a* ***declaração*** *da* ***variável***

Variáveis

- ***Manipulando Variáveis***
  - ***Sintaxe** para **atribuir valores** a **uma variável utilizando** os **três métodos possíveis***

```
-- Use esta sintaxe para atribuir um valor usando a palavra-chave SET
DECLARE @variavel INT
SET @variavel = <valor>

-- Use esta sintaxe para atribuir um valor usando uma instrução SELECT
SELECT @variavel <coluna ou expressão>
FROM <nome da tabela>

-- Use esta sintaxe para atribuir um valor a uma variável quando ela é declarada
DECLARE @variavel INT = <valor>
```

Variáveis

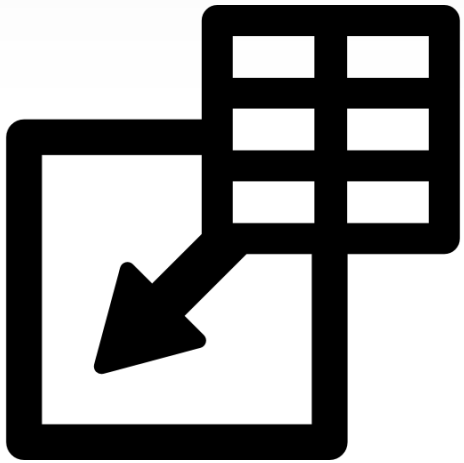
- ***Manipulando Variáveis***
  - ***Exemplo (08):***
    - ***Selecionando informações*** *de* ***produto utilizando*** *como* ***filtro*** *um* ***valor*** *de uma determinada* ***variável***

```
USE MyAdventureWorks;

DECLARE @ProductID INT = 1;

SELECT
   ProductID,
   ProductNumber,
   Name AS ProductName
FROM Production.Product
WHERE ProductID = @ProductID;
```

Modificação de Dados

Manipulação de Dados

- ***Inserindo Dados** em **Tabelas***
  - ***Embora** seja possível **usar várias técnicas** e **métodos** para **inserir dados** em **tabelas** no **SQL Server**, a melhor **estratégia** é começar com as **duas maneiras mais simples***
    - ***INSERT INTO***
    - ***SELECT INTO***



Manipulação de Dados

- ***Instrução INSERT INTO***
  - ***Permite adicionarmos uma** ou **várias linhas** em uma **única instrução***
  - ***Com esse método** é **possível inserir dados** em **todas** as **colunas**, em **colunas específicas**, em **colunas** de **identificação** e **diversas outras variações***
  - ***Exemplo (09):***
    - ***Inserindo** uma **nova tupla** na **tabela** Department, **esquema** HumanResources*

```
USE MyAdventureWorks;

INSERT INTO HumanResources.Department(Name, GroupName, ModifiedDate)
VALUES
('Payroll', 'Executive General and Administration', '06/12/2012');
```

Manipulação de Dados

- ***Instrução INSERT INTO***
  - *A **lista** de **colunas** é **opcional** na **instrução**, mas, por clareza, ela é sempre **recomendada***
  - *Se **não** for **incluída**, os **valores** serão **inseridos** na **tabela** com **base** na **ordem** das **colunas***
  - *As **colunas** de **identificação** **não** são **incluídas** na **ordem***
  - ***Nem toda coluna** da **tabela** precisa **aparecer** na **instrução** **INSERT***
  - ***É preciso especificar** um **valor** na **cláusula** **VALUES** para **todas** as **colunas listadas***

Manipulação de Dados

- ***Instrução INSERT INTO***
  - ***Exemplo (10):***
    - *Na **consulta anterior**, DepartmentID foi **omitido** na **lista** de **colunas***

```
USE MyAdventureWorks;

SELECT DepartmentID, Name, GroupName, ModifiedDate
FROM HumanResources.Department
ORDER BY DepartmentID DESC;
```

	DepartmentID	Name	GroupName	ModifiedDate
1	17	Payroll	Executive General and Administration	2012-06-12 00:00:00.000
2	16	Executive	Executive General and Administration	2002-06-01 00:00:00.000
3	15	Shipping and Receiving	Inventory Management	2002-06-01 00:00:00.000
4	14	Facilities and Maintenance	Executive General and Administration	2002-06-01 00:00:00.000
5	13	Quality Assurance	Quality Assurance	2002-06-01 00:00:00.000
6	12	Document Control	Quality Assurance	2002-06-01 00:00:00.000
7	11	Information Services	Executive General and Administration	2002-06-01 00:00:00.000
8	10	Finance	Executive General and Administration	2002-06-01 00:00:00.000
9	9	Human Resources	Executive General and Administration	2002-06-01 00:00:00.000
10	8	Production Control	Manufacturing	2002-06-01 00:00:00.000
11	7	Production	Manufacturing	2002-06-01 00:00:00.000
12	6	Research and Development	Research and Development	2002-06-01 00:00:00.000
13	5	Purchasing	Inventory Management	2002-06-01 00:00:00.000
14	4	Marketing	Sales and Marketing	2002-06-01 00:00:00.000
15	3	Sales	Sales and Marketing	2002-06-01 00:00:00.000
16	2	Tool Design	Research and Development	2002-06-01 00:00:00.000
17	1	Engineering	Research and Development	2002-06-01 00:00:00.000

Manipulação de Dados

- ***Instrução INSERT INTO***
  - ***Exemplo (11):***
    - *Como DepartmentID é uma **coluna** de **identificação**, um **valor** foi **inserido automaticamente** nela. Se quiser **inserir** um **valor** em uma **coluna** de **identificação**, use a **instrução SET IDENTITY_INSERT***

```
USE MyAdventureWorks;

SET IDENTITY_INSERT HumanResources.Department ON

INSERT INTO HumanResources.Department (DepartmentID, Name, GroupName, ModifiedDate)
VALUES
(18, 'International Marketing', 'Sales and Marketing', '05/26/2012');

SET IDENTITY_INSERT HumanResources.Department OFF
```

Manipulação de Dados

- ***Instrução INSERT INTO***
  - ***Exemplo (11):***
    - *Como DepartmentID é uma **coluna** de **identificação**, um **valor** foi **inserido automaticamente** nela. Se quiser **inserir** um **valor** em uma **coluna** de **identificação**, use a **instrução SET IDENTITY_INSERT***

```
USE MyAdventureWorks;

SET IDEN

INSERT I                                                    Date)
VALUES
(18, 'In

SET IDENTITY_INSERT HumanResources.Department OFF
```

As duas estratégias mencionadas inserem apenas uma linha.

Aproveitando a cláusula VALUES é possível inserir várias linhas em uma tabela com apenas uma instrução.

Sequências

- ***Instrução INSERT INTO***
  - *O **único mecanismo** para a **geração automática** de **números** no **SQL Server** era por meio de **uma coluna** de **identificação***
  - *Na **versão mais recente**, a **Microsoft introduziu** a **sequência** de **números***
  - ***Uma sequência** de **número** é um **objeto definido** pelo **usuário** que se **comporta** como uma **coluna** de **identificação**, pois **gera valores numéricos automaticamente***

Sequências

- ***Instrução INSERT INTO***
  - ***Todavia**, ao **contrário** da **coluna** de **identificação**, ela **não** está **associada** a um **objeto específico** no **BD***
  - ***Como resultado**, os **números gerados** pela **sequência** podem ser **usados** em **várias tabelas***

Sequências

- ***Instrução INSERT INTO***
  - *As **sequências** têm **algumas limitações** que podem fazer com que **alguns desenvolvedores** de **BD não** as **utilizem***
    - ***Não há garantia** de que os **números gerados** pela **sequência sejam únicos***
    - ***Esses números podem ser alterados**, o que é **uma grande diferença** em **relação** às **colunas** de **identificação***
    - *A **probabilidade** de **haver lacunas** entre os **números** é **maior**, especialmente quando a **sequência** é **compartilhada** por **várias tabelas***

Sequências

- ***Instrução INSERT INTO***
  - ***Sintaxe:***

```
CREATE SEQUENCE <esquema>.<nome da sequência>
AS INT
START WITH <algum número>
INCREMENT BY <algum número>;
```

  - *O **tipo** de **dados especificado** pode ser **qualquer inteiro** de **BD** e, por padrão, é **bigint***
  - *As **palavras-chave START WITH** são **opcionais***

Sequências

- ***Instrução INSERT INTO***
  - ***Exemplo (12): - Parte 1***
    - ***Criação** de uma **nova table** para **testar** o **uso** da **sequence***

```
USE MyAdventureWorks;

GO
IF (OBJECT_ID('dbo.States')) IS NOT NULL
   DROP TABLE dbo.States

GO
CREATE TABLE dbo.States(
StateID         INT CONSTRAINT pkStatesStatesID PRIMARY KEY,
StateName       VARCHAR(50),
StateAbbrev     CHAR(2)
)
GO
```

Sequências

- ***Instrução INSERT INTO***
  - ***Exemplo (12): - Parte 2***
    - ***Criando** um **objeto** **sequence** intitulado de StateSeq*

```
USE MyAdventureWorks;

GO
CREATE SEQUENCE dbo.StateSeq
AS INT
START WITH 1
INCREMENT BY 1

GO
```

Sequências

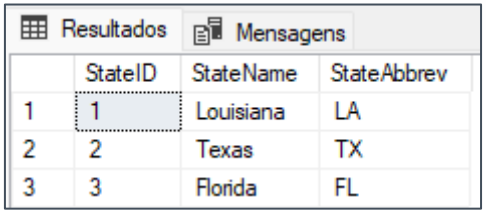
- ***Instrução INSERT INTO***
  - ***Exemplo (12): - Parte 3***
    - ***Utilizando** a **sequence** na **prática** com a **instrução INSERT***

```
USE MyAdventureWorks;

GO
INSERT INTO dbo.States(StateID, StateAbbrev, StateName)
VALUES
(NEXT VALUE FOR dbo.StateSeq, 'LA', 'Louisiana')
INSERT INTO dbo.States(StateID, StateAbbrev, StateName)
VALUES
(NEXT VALUE FOR dbo.StateSeq, 'TX', 'Texas')
INSERT INTO dbo.States(StateID, StateAbbrev, StateName)
VALUES
(NEXT VALUE FOR dbo.StateSeq, 'FL', 'Florida')
```

Sequências

- ***Instrução INSERT INTO***
  - ***Exemplo (12): - Parte 4***
    - ***Consultando a tabela States***

```
USE MyAdventureWorks;

GO
SELECT *
FROM dbo.States;
```



Resultados | Mensagens

	StateID	StateName	StateAbbrev
1	1	Louisiana	LA
2	2	Texas	TX
3	3	Florida	FL

INSERT

- ***Instrução INSERT INTO***
  - ***Exemplo (13):***
    - ***Inserindo várias linhas*** *simultaneamente na* ***tabela*** *Department*

```
USE MyAdventureWorks;

INSERT INTO HumanResources.Department
VALUES
('International Sales', 'Sales and Marketing', '05/26/2012'),
('Media Control', 'Quality Assurance', '05/26/2012');
```

Resultados | Mensagens

	DepartmentID	Name	GroupName	ModifiedDate
1	20	Media Control	Quality Assurance	2012-05-26 00:00:00.000
2	19	International Sales	Sales and Marketing	2012-05-26 00:00:00.000
3	18	International Marketing	Sales and Marketing	2012-05-26 00:00:00.000
4	17	Payroll	Executive General and Administration	2012-06-12 00:00:00.000
5	16	Executive	Executive General and Administration	2002-06-01 00:00:00.000
6	15	Shipping and Receiving	Inventory Management	2002-06-01 00:00:00.000
7	14	Facilities and Maintenance	Executive General and Administration	2002-06-01 00:00:00.000

INSERT

- ***Instrução INSERT INTO***
  - ***Exemplo (14):***
    - *Inserindo **várias linhas** de uma **tabela existente** em **outra tabela** usando a **instrução SELECT***

```
USE MyAdventureWorks;

INSERT INTO HumanResources.Department(Name, GroupName, ModifiedDate)
SELECT
   Name + ' USA', GroupName, ModifiedDate
FROM HumanResources.Department
WHERE DepartmentID IN (20, 19);
```

Resultados | Mensagens

	DepartmentID	Name	GroupName	ModifiedDate
1	22	Media Control USA	Quality Assurance	2012-05-26 00:00:00.000
2	21	International Sales USA	Sales and Marketing	2012-05-26 00:00:00.000
3	20	Media Control	Quality Assurance	2012-05-26 00:00:00.000
4	19	International Sales	Sales and Marketing	2012-05-26 00:00:00.000
5	18	International Marketing	Sales and Marketing	2012-05-26 00:00:00.000
6	17	Payroll	Executive General and Administration	2012-06-12 00:00:00.000

INSERT

- ***Instrução SELECT INTO***
  - *O **segundo método** que podemos **utilizar** para **inserir dados** em **tabelas** do **SQL Server** é por meio da **instrução SELECT INTO***
  - *Esse **método cria uma nova tabela** e **insere todas** as **linhas** da **instrução SELECT** nessa **tabela recentemente criada***

INSERT

- ***Instrução SELECT INTO***
  - ***Exemplo (15):***
    - ***Utilizando** a **instrução SELECT INTO** para **criar** um **novo objeto table***

```
USE MyAdventureWorks;

SELECT
   DepartmentID, Name, GroupName, ModifiedDate
INTO dbo.Departmet
FROM HumanResources.Department;
```

INSERT

- ***Instrução SELECT INTO***
  - ***Exemplo (15):***
    - ***Utilizando a instrução SELECT INTO para criar um novo objeto table***

A nova tabela pode ter características semelhantes ao esquema da tabela que atua como origem.

Contudo, o comprimento das colunas (tanto string como numéricas) pode mudar, e as chaves, os índices e as restrições não serão criados na nova tabela (não devemos utilizar SELECT INTO para uma tabela permanente).

UPDATE

- ***Atualizando Dados***
  - *Uma **instrução UPDATE** pode ser usada para **modificar uma** ou **várias linhas***
  - ***É necessário cautela** ao **executar** uma **instrução UPDATE**, pois é **altamente improvável** que todas as linhas de uma **tabela** precise ser **atualizada***
  - ***Todavia**, sempre pense na **possibilidade** de **incluir** a **cláusula WHERE** com **cada instrução UPDATE***
  - *Se a **cláusula WHERE não** for incluída, será atualizada todas as linhas acidentalmente*

UPDATE

- ***Atualizando Dados***
  - ***Exemplo (16):***
    - ***Essa instrução** utiliza uma **expressão** para **anexar** Europe na **coluna** Name em **uma linha** da **tabela** Department*

```
USE MyAdventureWorks;

UPDATE HumanResources.Department
   SET Name = Name + ' Europe'
WHERE DepartmentID = 19;
```

	DepartmentID	Name	GroupName	ModifiedDate
1	22	Media Control USA	Quality Assurance	2012-05-26 00:00:00.000
2	21	International Sales USA	Sales and Marketing	2012-05-26 00:00:00.000
3	20	Media Control	Quality Assurance	2012-05-26 00:00:00.000
4	19	International Sales Europe	Sales and Marketing	2012-05-26 00:00:00.000
5	18	International Marketing	Sales and Marketing	2012-05-26 00:00:00.000
6	17	Payroll	Executive General and Administration	2012-06-12 00:00:00.000
7	16	Executive	Executive General and Administration	2002-06-01 00:00:00.000

UPDATE

- ***Atualizando Dados***
  - ***Exemplo (17):***
    - *O **filtro utiliza** o **LIKE** para **garantir** que, se a instrução for **executada novamente**, **não** terá **efeito sobre** os **dados** que **foram atualizados***

```
USE MyAdventureWorks;

UPDATE HumanResources.Department
   SET Name = Name + ' Europe'
WHERE DepartmentID = 19
AND Name NOT LIKE '% Europe';
```

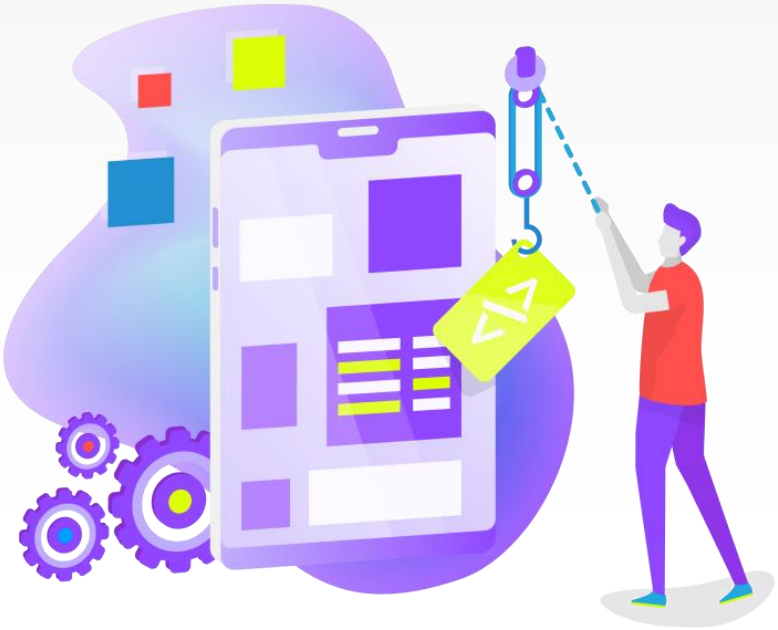
UPDATE

- ***Atualizando Dados***
  - ***Podemos encontrar*** *uma* ***situação*** *na qual* ***precisamos referenciar tabelas adicionais*** *enquanto* ***atualizamos*** *uma* ***linha***
  - *As* ***tabelas adicionais*** *poderiam ser* ***usadas*** *para* ***limitar*** *as* ***linhas*** *que* ***serão atualizadas*** *ou* ***poderiam fornecer*** *um* ***valor*** *a* ***ser usado*** *em* ***uma expressão*** *como* ***parte*** *da* ***atualização***
  - *Com a* ***cláusula JOIN*** *é* ***possível referenciar tabelas adicionais, exatamente*** *como* ***se faz*** *com* ***uma instrução SELECT***

UPDATE

- ***Atualizando Dados***
  - ***Exemplo (18):***
    - ***Utilizando** a **tabela** ProductSubCategory para **limitar** o **número** de **linhas atualizadas** a **somente** as que **estão incluídas** na **subcategoria** Socks. O ListPrice de **cada linha** que está nesse **subconjunto** é **incrementado** em 5%*

```
USE MyAdventureWorks;

UPDATE Production.Product
  SET ListPrice = p.ListPrice * 1.05
FROM Production.Product p
INNER JOIN Production.ProductSubcategory ps
   ON (p.ProductSubcategoryID = ps.ProductSubcategoryID)
WHERE
   ps.name = 'Socks';
```



EXERCÍCIOS

Referências

Noble, E.; Pro T-SQL 2019 Toward Speed, Scalability, and Standardization for SQL Server Developers. Apress, 2020.

Ben-Gan, I.; Microsoft SQL Server 2012 T-SQL Fundamentals. Pearson Education. 2012.

Lahoud, P.; Lopes, P.; T-SQL Querying: A guide to developing efficient and elegant T-SQL code. Packt Publishing. 2019.