

A Linguagem SQL

Capítulo 1. Introdução à Linguagem SQL

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 1. AULA 1.0. INTRODUÇÃO

PROF. GUSTAVO AGUILAR

Nesta Aula



- Apresentação do Professor
- Apresentação da Disciplina

Apresentação do Professor



Gustavo Aguilar



Atuação



- Administração de Bancos de Dados
- Ambientes de missão crítica em diversas plataformas
- Modelagem, Arquitetura e Engenharia de Dados
- Persistência e Pesquisa de Dados
- Sistemas Distribuídos e Cloud Computing
- Metodologias Ágeis e DevOps
- Professor e Instrutor Certificado Microsoft (MCT)



Formação



- Bacharelado em Ciência da Computação (PUC-Minas)
- Pós-Graduação em Administração de Banco de Dados
- Especialização em Docência do Ensino Superior
- MBA em Ciência de Dados (IGTI)

Apresentação da Disciplina



- Módulo de apresentação da Linguagem SQL que desenvolve as habilidades necessárias para criação de scripts que:
 - Consultam dados em um banco de dados
 - Criam, alteram e excluem dados e estruturas
- Apresenta o conceito de ferramentas de mapeamento objeto relacional (ORM);
- Apresenta ferramentas para controle de versão de bancos de dados.

Apresentação da Disciplina



Espera-se que o aluno consiga, ao final deste módulo:

- ✓ Compreender os conceitos básicos de banco de dados relacional;
- ✓ Compreender a aplicabilidade da Linguagem SQL;
- ✓ Usar instruções das classes DML, DDL, DCL e TCL da Linguagem SQL, no SQL Server;
- ✓ Criar e popular o schema físico de um banco de dados;
- ✓ Compreender os conceitos e funcionamento de queries distribuídas;
- ✓ Ser capaz de usar uma ferramenta de mapeamento objeto relacional (ORM);
- ✓ Implementar controle de versão em banco de dados.

Apresentação da Disciplina



- **Material Didático**

- ✓ Apostila
- ✓ Aulas gravadas
- ✓ Slides das aulas gravadas
- ✓ Aulas interativas ao vivo (*gravação e slides são disponibilizados)

- **Atividades dos Alunos**

- ✓ Fórum de dúvidas e fórum de debates
- ✓ Trabalho prático
- ✓ Desafio final
- ✓ Questões de reposição da aula interativa (*para quem não assistiu a aula)

Apresentação da Disciplina



- **Distribuição de pontos**

- ✓ **10 pontos** de participação na 1ª Aula Interativa (*presença computada através de resposta à enquete)
 - ✓ **25 pontos** do Trabalho Prático
 - ✓ **10 pontos** de participação no Fórum de Debates do módulo 2
 - ✓ **10 pontos** de participação na 2ª Aula Interativa (*presença computada através de resposta à enquete)
 - ✓ **40 pontos** do Desafio do módulo 2
 - ✓ **5 pontos** do Feedback do aluno em relação ao conteúdo e professor do módulo.
-
- **TOTAL: 100 pontos**

Próxima Aula



- Banco de Dados Relacional: Uma Breve Revisão

A Linguagem SQL

CAPÍTULO 1. AULA 1.1. BANCO DE DADOS RELACIONAL: UMA BREVE REVISÃO

PROF. GUSTAVO AGUILAR

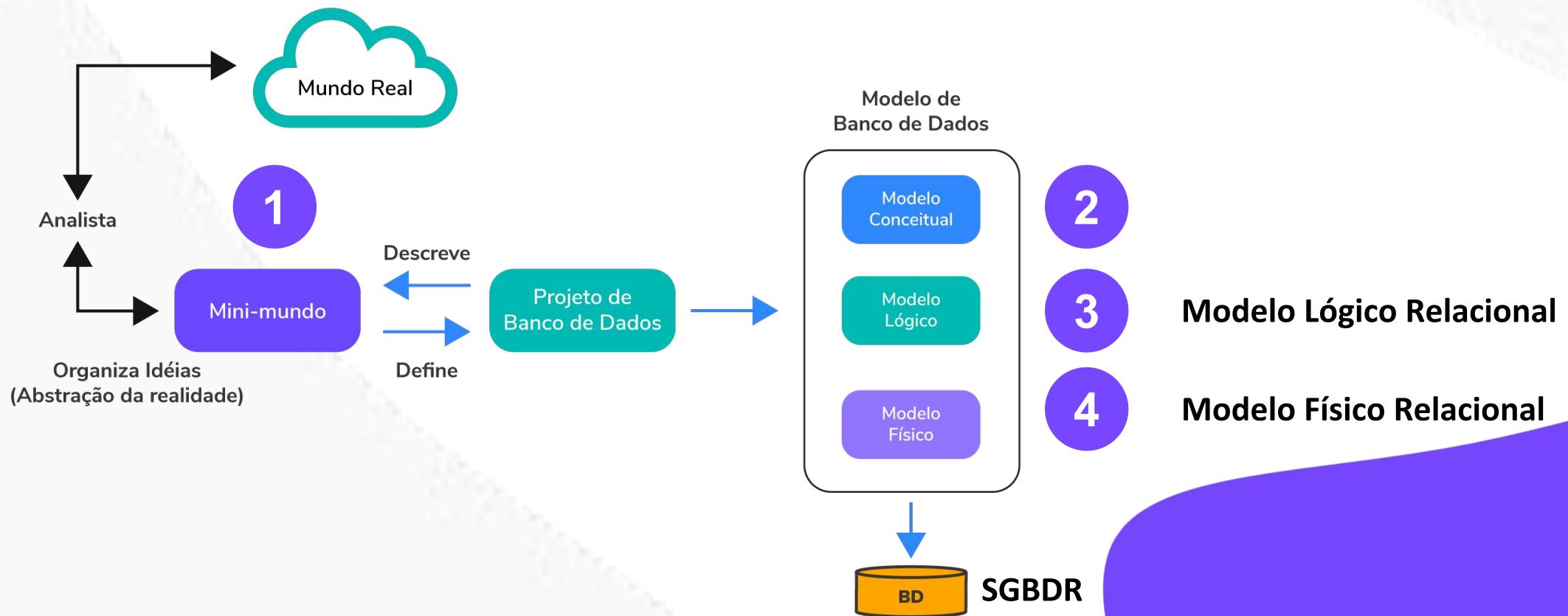
Nesta Aula



- O Modelo Relacional
- Elementos do Modelo de Dados Relacional
- Modelo de Dados Físico
- As Doze Regas de Codd

O Modelo Relacional

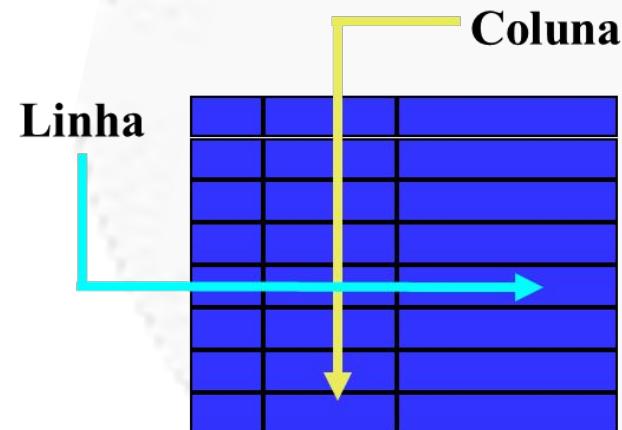
- Visão Geral do Processo de Modelagem de Dados (abordagem relacional)



Modelo Relacional

IGTI

- Criado por Edgar Frank Codd, em junho de 1970
 - Artigo *A Relational Model of Data for Large Shared Data Banks.*
- Representação dos dados é independente de como eles são organizados internamente nos servidores
- Representação tabular dos dados □
- Baseado na Álgebra Relacional



O Modelo Lógico Relacional

- Dados sendo vistos como uma **tupla** □ linha formada por uma lista ordenada de colunas.

COD_CLIENTE	NOM_CLIENTE	NUM_TEL_CLIENTE	IND_SEXO_CLIENTE
1	JOÃO	33333333	M
2	MARIA	99999999	F
3	JOSÉ	88888888	M
4	TIAGO	66666666	M
5	PEDRO	44444444	M
6	MATEUS	11111111	M
7	ANA	77777777	F
8	TEREZA	55555555	F

- Facilidade para manipular e visualizar dados □ “boom” do modelo.

Elementos do Modelo de Dados Relacional



- **Tabela:** objeto correspondente à entidade do modelo conceitual
- **Domínio:** natureza dos valores. Exemplos:
 - CODIGO CLIENTE □ NUMBER
 - NOME CLIENTE □ STRING
 - DATA NASCIMENTO □ DATE
- **Coluna (campo):** atributo (modelo conceitual) + domínio do dado

Elementos do Modelo de Dados Relacional



- **Chave Primária (“Primary Key” / “PK”):** coluna(s) que identificam unicamente uma linha (tupla) em uma tabela □ Ex.: ID ÚNICO, CPF.
- **Chave Candidata:** não repetem valor e podem ser PK
 - Exemplos: TÍTULO DO ELEITOR, RG, CPF.
- **Chave Estrangeira (“Foreign Key” / “FK”):** elo de ligação.

Elementos do Modelo de Dados Relacional



- **Cardinalidade:** além de representar a quantidade de elementos das entidades envolvidos na relação, indicam também as **restrições de nulidade das chaves estrangeiras**.
 - Notação *Information Engineering (IE)*

0,1 : 0,N 

1 : 0,N 

N : N 

0,1 : 0,1 

1 : 0,1 

Elementos do Modelo de Dados Relacional

IGTI

- Cardinalidades **0,1** e **0,N** □ chaves estrangeiras nulas.

0,1:

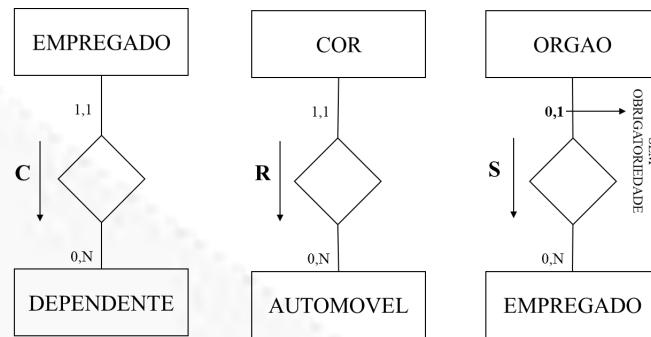
- Cardinalidades **1:0,N** e **1:0,1** □ chaves estrangeiras não nulas.

1:

Elementos do Modelo de Dados Relacional

IGTI

- **Restrições de Integridade:** regra que garante a consistência dos dados em um relacionamento entre duas tabelas (*definida na constraint da FK)
 - **Cascade (C):** deleção em cascata
 - **Restrict (R):** não permite deletar um registro que seja chave estrangeira
 - **Set Null (S):** seta chave estrangeira para nulo



Elementos do Modelo de Dados Relacional



- **Restrições de Integridade:** algoritmo para auxiliar a definição.

Posso Eliminar Chave Primária (registro pai)?

Se não puder → RESTRICT

Se puder:

Preciso Manter os Filhos (registros na tabela da FK)?

Se precisar → SET NULL

Se não precisar → CASCADE

Modelo de Dados Físico Relacional



- Modelo de dados físico de um Sistema de Agência de Turismo para o SGBDR Oracle.

As Doze Regras de Codd



- Publicadas em 1981, com o propósito de definir o que seria necessário para um SGBD ser considerado relacional.
 - **Proteger a visão relacional, dos fornecedores de SGBDs** □
- Numeradas de 1 à 12
- **REGRA 0:** regra base para todas as outras 12
 - Gerenciamento exclusivamente pelas capacidades relacionais do SGBD.
- **REGRA 1:** A Regra da Informação
 - Representada explicitamente no nível lógico e por valores em tabelas.

As Doze Regras de Codd



- **REGRA 2:** Regra de Garantia de Acesso
 - Logicamente acessível com nome da tabela + chave + coluna.
- **REGRA 3:** Tratamento Sistemático de Valores Nulos
 - Possibilidade de valores nulos para as colunas.
- **REGRA 4:** Catálogo Online e Dinâmico Baseado no Modelo Relacional
- **REGRA 5: Regra da Linguagem de Dados Compreensiva**
 - Linguagem de alto nível: abstrair os detalhes internos da engine do banco de dados e do “bit-a-bit” do armazenamento dos dados para os usuários.
 - Um dos pilares para o sucesso e uso em larga escala da **Linguagem SQL**.



As Doze Regras de Codd



- **REGRA 6:** Regra da Atualização de Visões
- **REGRA 7:** Inserções, Atualizações e Deleções de Alto Nível
- **REGRA 8:** Independência Física dos Dados
- **REGRA 9:** Independência Lógica dos Dados
- **REGRA 10:** Independência de Integridade
- **REGRA 11:** Independência de Distribuição
- **REGRA 12:** Regra de Não Subversão

Conclusão



- ✓ Processo de Modelagem de Dados como meio para adicionar gradativamente detalhes e complexidade em Projetos de BDs.
- ✓ Minimundo Mod. Conceitual Mod. Lógico Mod. Físico
- ✓ Facilidade para manipular e visualizar dados no Modelo de Dados Relacional como grande responsável para o “boom” do modelo.
- ✓ 5^a Regra das 12 Regas de Codd como grande pilar para o sucesso e uso em larga escala da Linguagem SQL.

Próxima Aula



- ❑ História da Linguagem SQL

A Linguagem SQL

CAPÍTULO 1. AULA 1.2. HISTÓRIA DA LINGUAGEM SQL

PROF. GUSTAVO AGUILAR

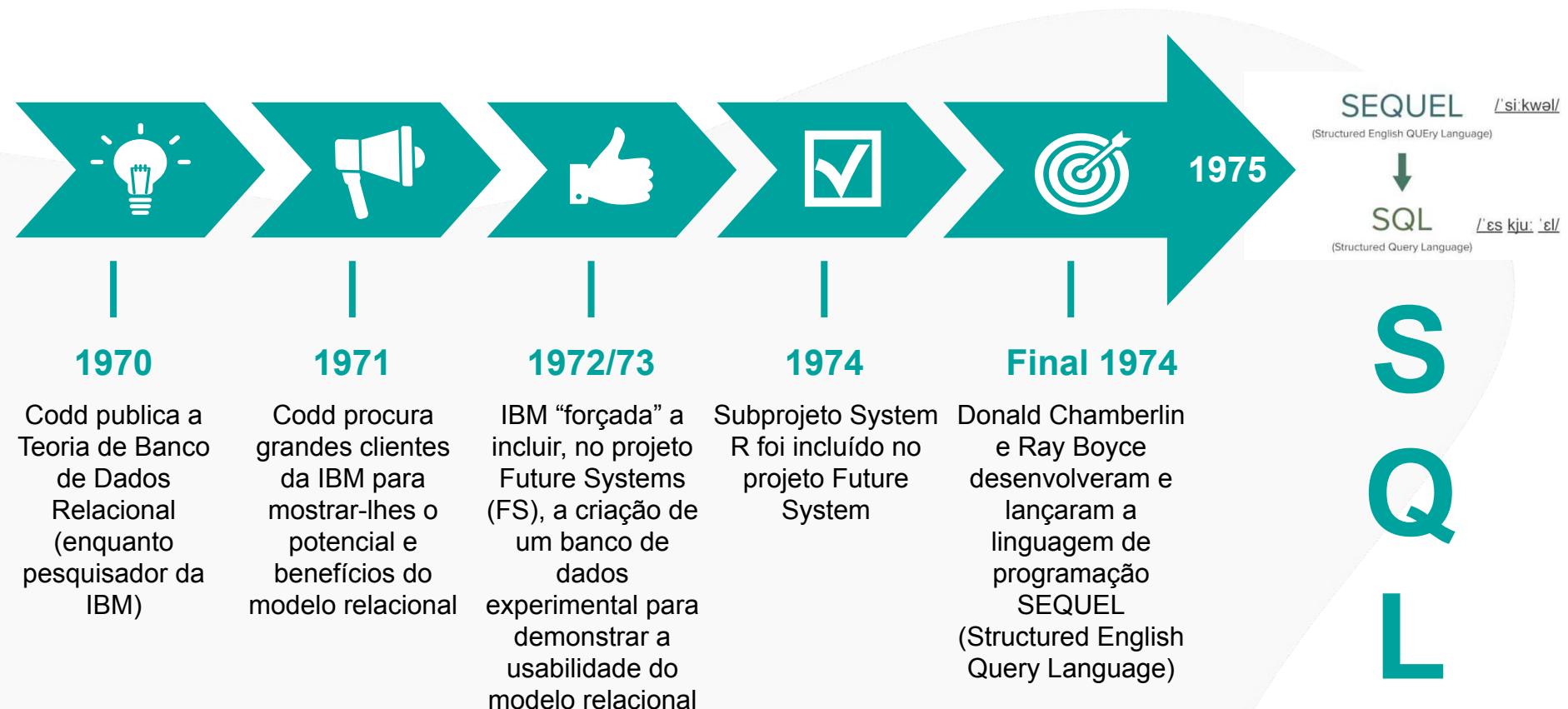
Nesta Aula



- ❑ Surgimento da Linguagem SQL
- ❑ Características da Linguagem SQL

Surgimento da Linguagem SQL

IGTI



Características da Linguagem SQL



- **Linguagem Declarativa**

- Mais simples e com curva de aprendizado menor
 - Linguagens procedurais da época requeriam que o usuário especificasse o caminho (navegação, posição etc.) para se chegar ao resultado.

Repita para cada linha de dados:

Se a cidade for São Paulo, retorne a linha; caso contrário, não faça nada.

Mova para a próxima linha.

Final do Loop.

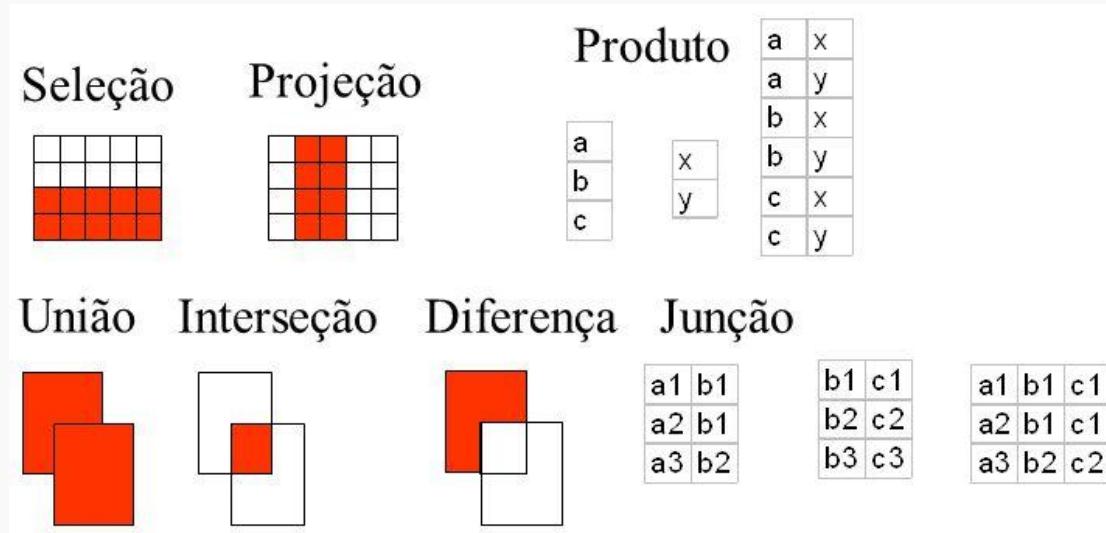
- Especificar a forma (conteúdo desejado) do resultado e não o caminho para se chegar a ele.

Retorne todos os clientes cuja cidade é São Paulo.

Características da Linguagem SQL

IGTI

- Baseada na **Álgebra Relacional e Teoria de Conjuntos**
 - Codd utilizou as já conhecidas operações da Álgebra Relacional, e as operações da Teoria de Conjuntos para sustentar Sua fundamentação teórica.



Características da Linguagem SQL



- Usando-se dos operadores da Álgebra Relacional, Codd conseguiu fundamentar a viabilidade e usabilidade do modelo relacional e seu armazenamento de dados no formato tabular, em tuplas.

OPERAÇÃO	SÍMBOLO	SINTAXE
Projeção	π ("pi")	π <lista de campos> (Tabela)
Seleção/ Restrição	σ ("sigma")	σ <condição de seleção> (Tabela)
União	\cup	(Tabela 1) \cup (Tabela 2)
Interseção	\cap	(Tabela 1) \cap (Tabela 2)
Diferença	-	(Tabela 1) - (Tabela 2)
Produto Cartesiano	X	(Tabela 1) X (Tabela 2)
Junção	$ X $	(Tabela 1) $ X $ <condição de junção> (Tabela 2)
Divisão	\div	(Tabela 1) \div (Tabela 2)
Renomeação	ρ ("rho")	ρ Nome(Tabela)
Atribuição	\leftarrow	Variável \leftarrow Tabela

Características da Linguagem SQL

IGTI

- **Restrição (σ):** retornar somente as tuplas que satisfaçam à uma condição.

$\sigma_{\text{salario} < 2000} (\text{Empregado})$

Empregado

codEmp	Nome	Salario	idade	codDep
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

Resultado

codEmp	Nome	Salario	idade	codDep
203	Ana	1.800,00	25	002

Características da Linguagem SQL

IGTI

- **Projeção (π):** retornar um ou mais atributos desejados.

π nome, idade (Empregado)

Empregado

codEmp	Nome	Salario	idade	codDep
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

Resultado

Nome	idade
Pedro	45
Paulo	43
Maria	38
Ana	25

Características da Linguagem SQL



- **Interseção (\cap):** retornar tuplas comuns entre duas tabelas.

Exemplo: retornar o nome e o CPF dos funcionários que estão internados como pacientes.

$$\pi \text{ nome,CPF } (\text{FUNCIONARIO}) \cap \pi \text{ nome,CPF } (\text{PACIENTE})$$

- **União (u):** retornar a união das tuplas de duas tabelas.

Exemplo: retornar o nome e o CPF dos médicos e pacientes cadastrados no hospital.

$$\pi \text{ nome,CPF } (\text{MEDICO}) \cup \pi \text{ nome,CPF } (\text{PACIENTE})$$

Conclusão



- Após pressão de grandes clientes que tiveram contato com a Teoria Relacional de Codd, a IBM criou o Projeto System R para desenvolver um banco relacional experimental.
- A Linguagem SEQUEL foi criada dentro desse projeto, em 1974 por Donald Chamberlin e Ray Boyce.
- Por questões de patente teve o nome alterado para SQL.
- Por ser uma linguagem declarativa, mais simples, baseada nas operações da Álgebra Relacional e Teoria de Conjuntos, e com menor curva de aprendizado, teve grande aceitação.

Próxima Aula



- Padronização da Linguagem SQL, Classes de Instruções SQL e Dialetos

A Linguagem SQL

CAPÍTULO 1. AULA 1.3. PADRONIZAÇÃO DA LINGUAGEM SQL, CLASSES DE INSTRUÇÕES SQL E DIALETOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Padronização da Linguagem SQL
- Dialetos
- Classes de Instruções SQL

Padronização da Linguagem SQL



- Linguagem SQL originalmente criada pela IBM
- Após a sua publicação, rapidamente surgiram várias linguagens desenvolvidas pelos fornecedores de SGBDs relacionais, cada uma com suas adaptações à Linguagem SQL original
 - Ex.: PL/SQL da Oracle.
 - Levou à necessidade da criação de um padrão para a Linguagem SQL.
- Publicação do **padrão ANSI X3.135-1986** pelo American National Standards Institute, em 1986



Padronização da Linguagem SQL



- Após alguns meses ao lançamento do ANSI X3.135-1986, a International Organization for Standardization (ISO) publicou um padrão tecnicamente idêntico, o **ISO 9075-1987**
 - Para o **cenário internacional**
- Esses padrões foram revisados em conjunto:
 - Primeiramente em 1989: **ANSI X3.135-1989 e ISO / IEC 9075: 1989**
 - Novamente em 1992: **ANSI X3.135-1992 e ISO / IEC 9075: 1992.**
- Essas edições se tornaram conhecidas como **SQL-86, SQL-89 e SQL-92**

Padronização da Linguagem SQL



- O padrão foi revisado novamente em 1999 (**SQL3**), 2003, 2008, 2011 e 2016 (**edição atual ISO / IEC 9075: 2016**)
- A norma internacional ISO / IEC 9075 para SQL é desenvolvida pelo ISO/IEC Joint Technical Committee (JTC) for Information Technology
- Desde a edição de 2003 foi subdividida em 9 partes, cada uma cobrindo um aspecto do padrão geral, sob o título **Tecnologia da Informação – Linguagens de Banco de Dados – SQL**.

Dialectos



- Fornecedores de SGBDs continuaram fazendo suas próprias implementações da Linguagem SQL
- Grande parte da implementação era baseada nos padrões da Linguagem SQL, mas com pequenas adaptações ou personalizações:
 - PL/SQL da Oracle
 - **T-SQL da Microsoft (usada no SQL Server)**
 - SQL PL da IBM

Dialectos



- **Resultado:** pequenas diferenças na sintaxe de implementação para um mesmo tipo de comando SQL entre os diversos dialectos (SGBDs)
 - Ex: retornar os **N** registros de uma tabela com a cláusula **TOP**

- Sintaxe de uso no dialeto SQL Server:

```
SELECT TOP [QTDE DE REGISTROS] [COLUNA(S)] FROM [TABELA]
```

- Sintaxe de uso nos dialectos MySQL e PostgreSQL:

```
SELECT [COLUNA(S)] FROM [TABELA] LIMIT [QTDE DE REGISTROS]
```

- Sintaxe de uso no dialeto Oracle:

```
SELECT [COLUNA(S)] FROM [TABELA] WHERE ROWNUM < [QTDE REGISTROS]
```

- Sintaxe de uso no dialeto Firebird:

```
SELECT FIRST [QTDE DE REGISTROS] [COLUNA(S)] FROM [TABELA]
```

Classes de Instruções SQL

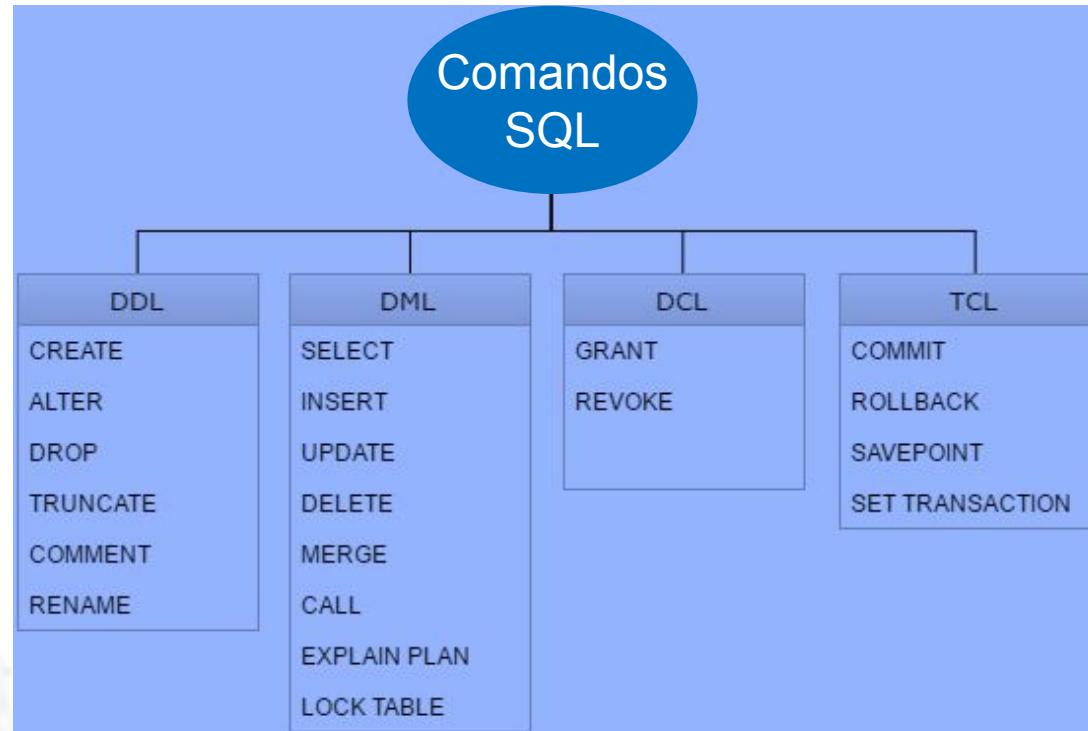


- Dentro do padrão ISO SQL: **quatro classes de comandos** da Linguagem SQL.

DDL	DML	DCL	TCL
Linguagem de Definição de Dados	Linguagem de Manipulação de Dados	Linguagem de Controle de Dados	Linguagem de Controle de Transação
Comandos para a criação do banco de dados e dos objetos no banco de dados, como tabelas, índices, constraints etc	Comandos para inserir, atualizar, deletar e consultar dados	Comandos para gerenciar permissões de acesso para usuários e objetos	Comandos para realizar controle das transações de dados no banco de dados
Comandos para alteração e exclusão de objetos		Permissões para executar comandos DDL, DML, DCL e TCL	
<i>Data Definition Language</i>	<i>Data Manipulation Language</i>	<i>Data Control Language</i>	<i>Transaction Control Language</i>

Classes de Instruções SQL

IGTI



Conclusão



- Publicação do primeiro padrão da Linguagem SQL pelo American National Standards Institute, em 1986: ANSI X3.135-1986.
- International Organization for Standardization (ISO) publicou um padrão tecnicamente idêntico, o ISO 9075-1987.
- Pequenas diferenças de sintaxes entre os diversos dialetos.
- Várias revisões feitas depois, sendo a SQL-86, SQL-89 e SQL-92 as mais conhecidas, e a ISO / IEC 9075: 2016 a atual.
- Dentro do padrão ISO SQL existem quatro classes de comandos da Linguagem SQL: DDL, DML, DCL e TCL.

Próxima Aula



- Uma “Sopa de Letras”

A Linguagem SQL

CAPÍTULO 1. AULA 1.4. UMA “SOPA DE LETRAS”

PROF. GUSTAVO AGUILAR

Nesta Aula



- A SQL x O SQL
- NoSQL x NOSQL
- UnQL
- NewSQL

A SQL x O SQL



- A SQL □ A **Linguagem SQL**.
- O SQL □ menção ao **SQL Server**, sistema gerenciador de banco de dados relacional desenvolvido pela Microsoft.
- Encontramos também o nome de outros SGBDs fazendo menção à Linguagem SQL, de forma a **indicar que são relacionais**:
 - **MySQL** fornecido pela Oracle (www.mysql.com)
 - **PostgreSQL** desenvolvido pela PostgreSQL Global Development Group e de código aberto (www.postgresql.org)
 - **SQLite**: banco de dados portátil e de código aberto (www.sqlite.org)

NoSQL x NOSQL



- “NOSQL” pode se referenciar à duas tecnologias diferentes, mas nenhuma delas relacionadas diretamente à Linguagem SQL.
- A diferença entre essas duas tecnologias é feita na grafia:
 - **NoSQL** versus **NOSQL**
- O termo **NoSQL** surgiu em 1998, quando **Carlo Strozzi** batizou seu novo produto de *NoSQL*:
 - Um banco de dados relacional
 - Não fornecia nenhum tipo de linguagem SQL para consulta
 - Era pronunciado como *noseequel* (*No + SQL*)

NoSQL x NOSQL



- Os dados do *NoSQL* eram armazenados em arquivos *ASCII UNIX*:
 - Manipulados por utilitários *UNIX* comuns: *ls*, *mv*, *cp* e editores como o '*vi*'.
- Manipulação dos dados sem a linguagem SQL ☐ “*non SQL*”
 - No sentido de realmente não ser um banco de dados SQL
- Formato dos arquivos: baseado em relação (tabela, linhas e colunas) ☐ **família de SGBD relacional.**
- Logo:



NoSQL x NOSQL



- Strozzi afirmava categoricamente que seu produto “*nada tinha a ver com o recém-nascido Movimento NOSQL*”:
 - “...é um banco de dados relacional para todos os efeitos e apenas intencionalmente não usa SQL como uma linguagem de consulta...”
 - “...o recém-chegado é principalmente um conceito, que se afasta do modelo relacional...”
 - “....deveria ter sido chamado de ‘NoREL’, , já que não ser baseado em SQL é apenas uma consequência óbvia de não ser relacional, e não o contrário”.

NoSQL x NOSQL

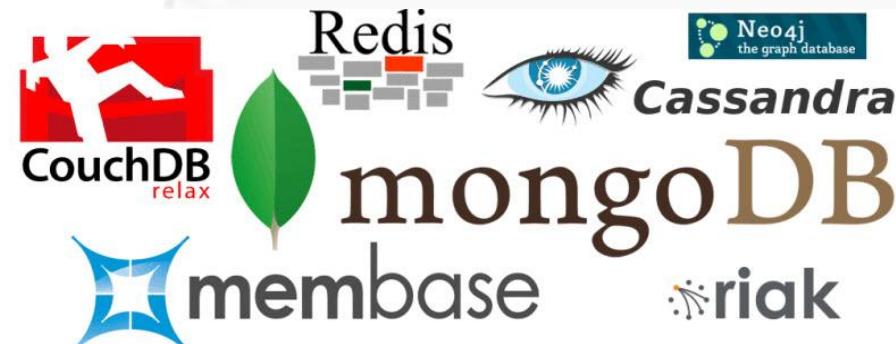


- No início de 2009 ☐ termo NOSQL voltou à tona
 - Johan Oskarsson, desenvolvedor na *Last.fm*
 - Organizou um evento para discutir "bancos de dados distribuídos, não relacionais e de código aberto"
 - Termo tentava rotular o surgimento de um número crescente de repositórios de dados distribuídos, não relacionais
 - Clones de código aberto do *Google Bigtable/MapReduce* e do *DynamoDB* da *Amazon*.
 - Desde então ☐ termo NOSQL tem sido atribuído aos **SGBDs não relacionais** (e não somente mais aos “não SQL”):
 - Mecanismo de armazenamento de dados que não é baseado no formato tabular (linhas x colunas).

NoSQL x NOSQL

IGTI

- Termo *NoSQL* passou a ser escrito como **NOSQL**;
- SGBDs “Não Apenas SQL” ☐ **Not Only SQL**;
- Enfatizar que suportavam outras linguagens de consulta, que não a SQL.



UnQL

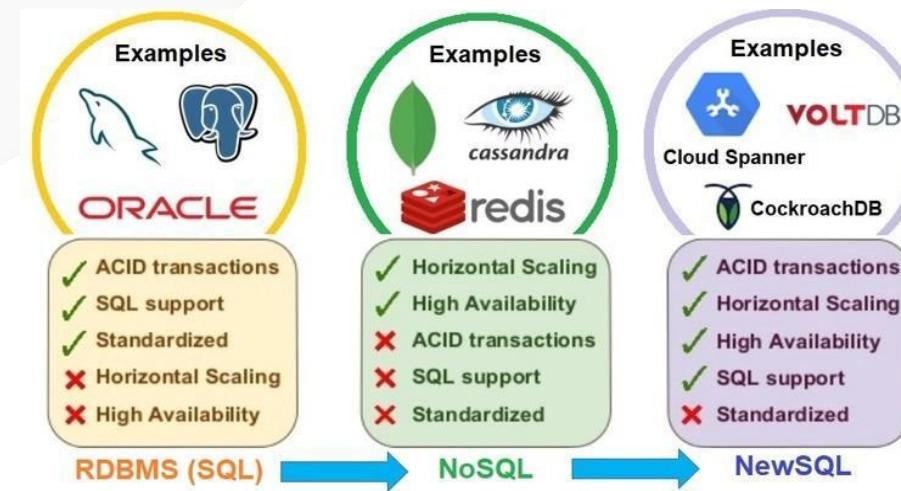


- Em 29 de julho de 2011:
 - Fundador da Couchbase e inventor do CouchDB, Damien Katz
 - +
 - Richard Hipp, inventor do SQLite
 - ‘UNQL QUERY LANGUAGE UNVEILED BY COUCHBASE AND SQLITE’
 - Linguagem de consulta padrão NOSQL ☐ **UnQL**
 - Unstructured Query Language;
 - Pronúncia “Uncle”;
 - Tentar se popularizar da mesma forma como o *SQL* do movimento relacional se popularizou;
 - Site do projeto: <http://unql.sqlite.org/index.html/wiki?name=UnQL>.

NewSQL

IGTI

- Não é uma nova versão da Linguagem SQL
- Trata-se de uma **nova geração de SGBDs relacionais distribuídos**
- **Novos players**
 - Amazon Aurora, VoltDB
 - CockroachDB, CosmosDB
 - Google Spanner
- SGBDRs já conhecidos
 - Com a **engine reformulada**
 - MySQL Cluster, SQL Server Column Store



Conclusão



- O termo “a SQL” refere-se a Linguagem SQL e o termo “o SQL” faz menção do SGBD SQL Server da Microsoft.
- Muitos SGBDs usam SQL no nome para se identificarem como relacionais.
- Surgimento do termo NoSQL inicialmente como um SGBDR sem suporte à linguagem SQL e o termo NOSQL concretizado para designar os SGBDs não relacionais.
- UNQL linguagem aspirante à padrão para SGBDs NOSQL e o termo NewSQL para nomear a nova geração de SGBDRs distribuídos.

Próxima Aula



- Capítulo 2. Introdução ao SQL Server

A Linguagem SQL

Capítulo 2. Introdução ao SQL Server

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 2. AULA 2.1. INTRODUÇÃO AO SQL SERVER

PROF. GUSTAVO AGUILAR

Nesta Aula



- Introdução ao SQL Server
- Produtos da Família SQL Server

Introdução ao SQL Server



- Microsoft SQL Server é um sistema gerenciador de banco de dados **relacional**, desenvolvido pela Microsoft.
 - Será usado para aplicarmos na prática os conceitos e instruções da Linguagem SQL.
- Primeira versão (1.0): lançada em 1989.
 - Para sistema operacional OS/2.
- Suporte para Windows: a partir da 4.2.1.
 - Suporte para Linux desde a versão 2017
- Versão atual: 2019.

Versão	Ano de Lançamento
2019	2019
2017	2017
2016	2016
2014	2014
2012	2012
2008 R2	2010
2008	2008
2005	2005
2000	2000
7.0	1998
6.5	1996
6.0	1995
4.2.1	1994
4.2	1992
1.1	1991
1.0	1989

Introdução ao SQL Server

IGTI



Produtos da Família SQL Server

The Microsoft SQL Server logo, featuring a red and white stylized network icon next to the text 'Microsoft SQL Server'.

Database Engine

Bancos OLTP
Cluster Failover
Snapshot
Database Mirror
Replicação
Always On

SQL

The Microsoft SQL Server Analysis Services logo, featuring a red and white stylized network icon next to the text 'Microsoft SQL Server Analysis Services'.

Analytical Engine

Bancos OLAP
Cubos
Tabelas Fato / Dimensão

SSAS

The Microsoft SQL Server Reporting Services logo, featuring a red and white stylized network icon next to the text 'Microsoft SQL Server Reporting Services'.

Report Engine

Pastas
Relatórios
Dashboards
Assinaturas

SSRS

The Microsoft SQL Server Integration Services logo, featuring a red and white stylized network icon next to the text 'Microsoft SQL Server Integration Services'.

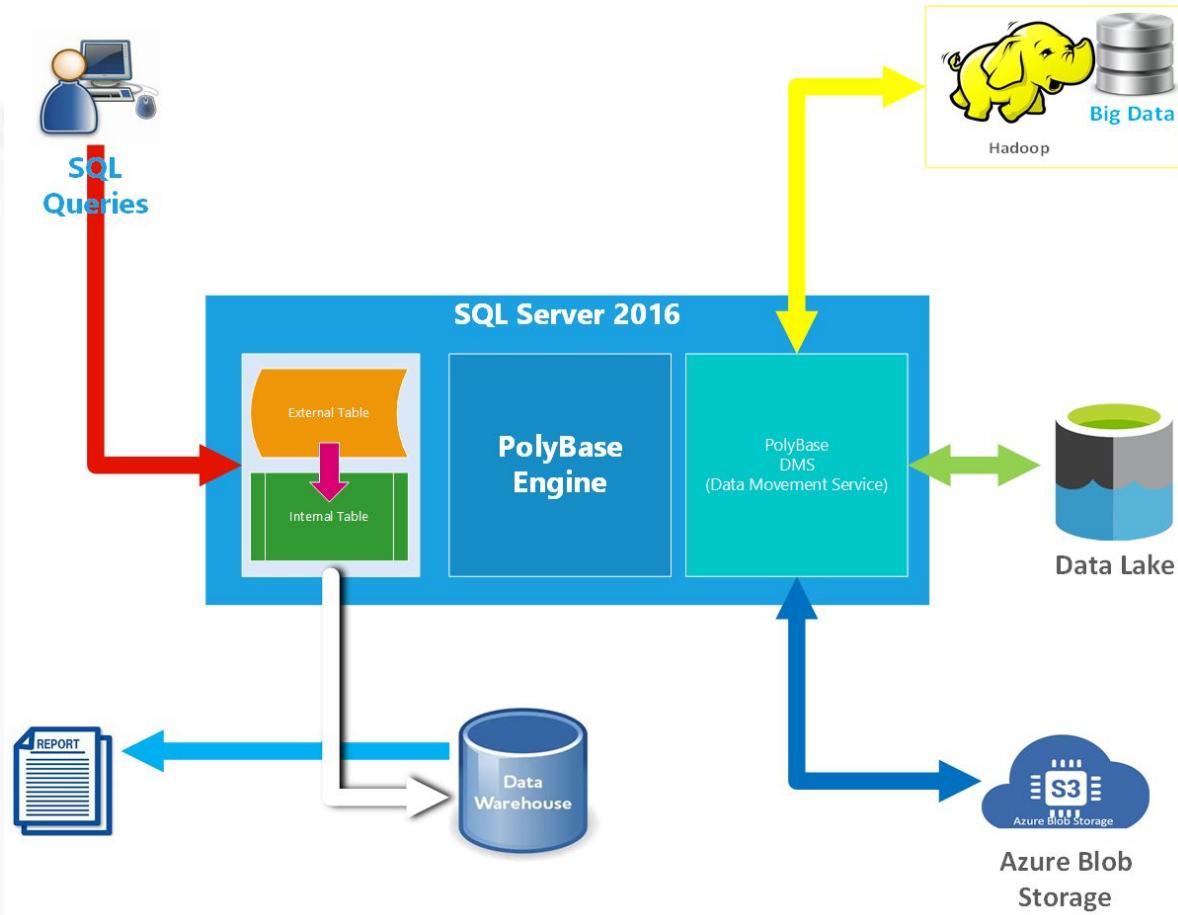
ETL Engine

Catálogos
Pacotes
Extração
Transformação
Carga de Dados

SSIS

Produtos da Família SQL Server

IGTI



Próxima Aula



- Conceitos Básicos do SQL Server

A Linguagem SQL

CAPÍTULO 2. AULA 2.2. CONCEITOS BÁSICOS DO SQL SERVER

PROF. GUSTAVO AGUILAR

Nesta Aula



- Conceitos e Macroarquitetura
- Segurança de Autenticação

Conceitos e Macroarquitetura

IGTi

Servidor



Instância



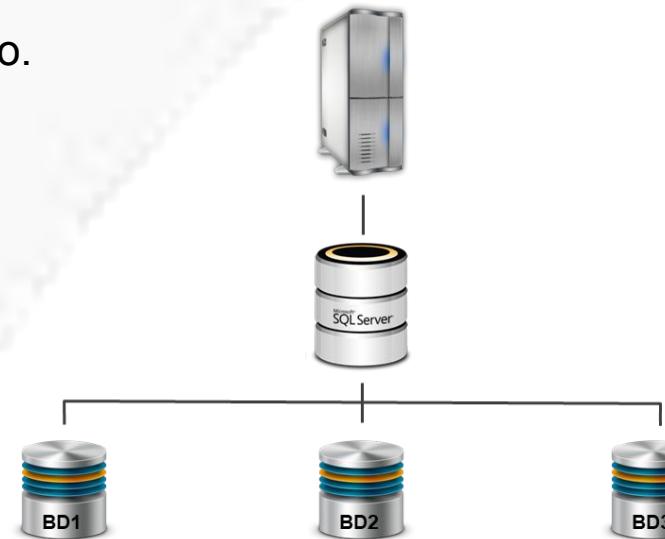
Banco de Dados / Pasta / Catálogo



Conceitos e Macroarquitetura

IGTI

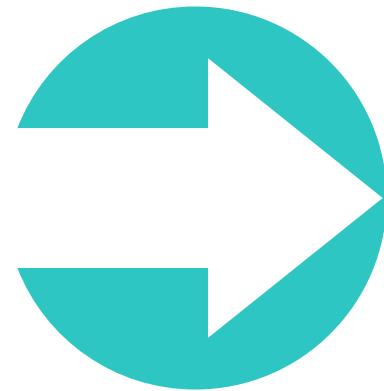
- Um servidor pode conter 1 ou mais Instâncias SQL/SSAS/SSRS
 - Da mesma versão ou não.
 - SSIS não é multi-instância, apenas multiversão.
- As instâncias são diferentes entre si
 - Conjunto próprio de bancos/pastas/catálogos.
- Uma Instância pode conter vários BDs
 - Até 32.767 bancos de dados.
- Os bancos são diferentes entre si
 - Cada um possui seu conjunto de tabelas e usuários.



Segurança de Autenticação



SQL Authentication

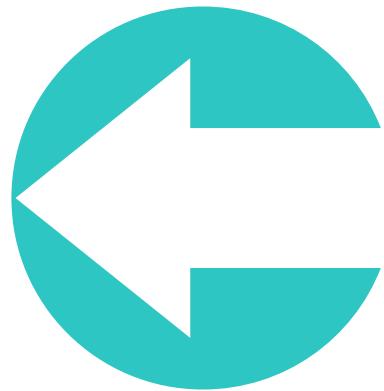


Login interno do SQL

Nome do login e senha
são armazenados no SQL.
Necessário informar login e
senha para fazer o logon.



Windows Authentication

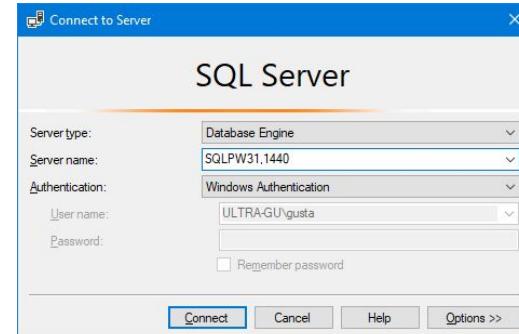


*Não disponível para SSAS SSRS.



Login externo ao SQL

Nome do login e senha
são armazenados no SO/AD.
Quando já autenticado no SO,
necessário informar somente
login para fazer o logon.



Próxima Aula



- Instalação do SQL Server

A Linguagem SQL

CAPÍTULO 2. AULA 2.3. INSTALAÇÃO DO SQL SERVER

PROF. GUSTAVO AGUILAR

Nesta Aula

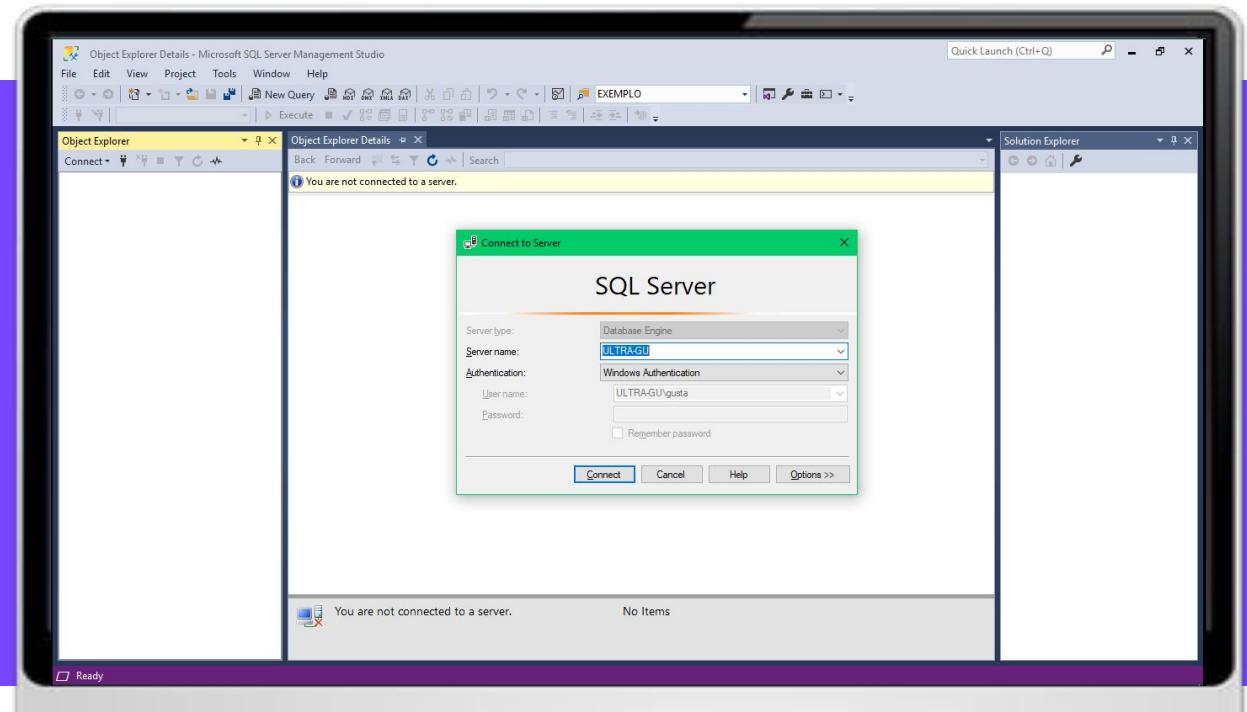


- ❑ Instalação do SQL Server

Instalação do SQL Server



Demo



Próxima Aula



- Introdução à T-SQL

A Linguagem SQL

CAPÍTULO 2. AULA 2.4. INTRODUÇÃO À T-SQL

PROF. GUSTAVO AGUILAR

Nesta Aula



- T-SQL
- SQL Server Management Studio
- Azure Data Studio
- Scripts, Projetos e Soluções no SSMS

T-SQL

IGTI

- **Transact SQL (T-SQL):** implementação da Microsoft, no SQL Server, para a Linguagem SQL ISO / IEC 9075.
- Linguagem declarativa.
- Possui todas as opções e instruções existentes no padrão ISO
 - Variáveis, loop, controle de fluxo e decisões, funções, etc.
 - Possui opções e instruções que não estão no padrão.
 - Algumas instruções com sintaxe diferente do padrão e/ou de outros SGBDs.



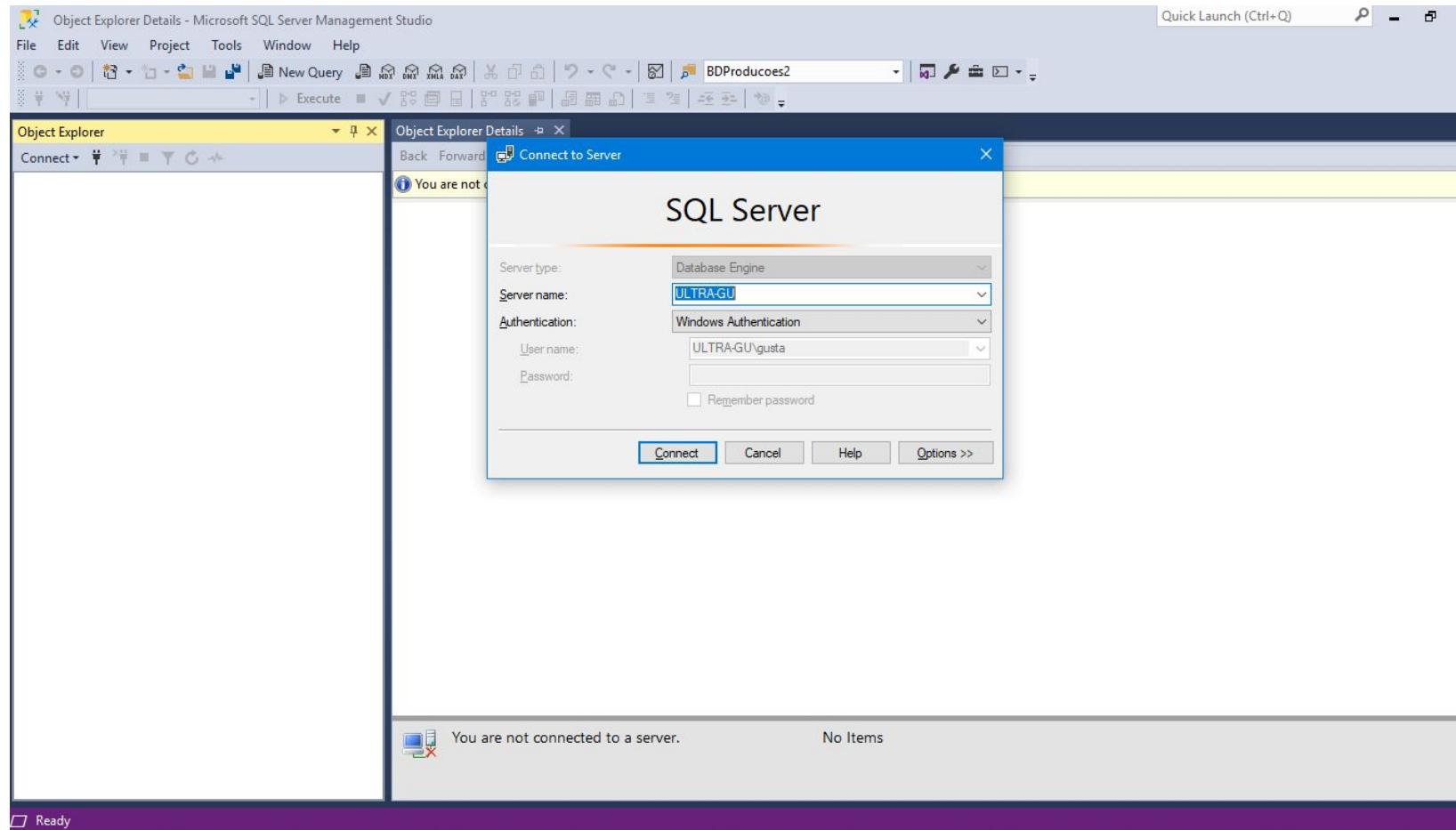
SQL Server Management Studio



- Interface gráfica (client) oficial da Microsoft para se conectar e interagir com o SQL Server.
- Também chamado de **SSMS**.
<https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2019>
- Permite administrar o SQL Server e os bancos de dados.
- Possui editor de queries e vários *Wizards*.
- Construtor de modelos de dados (diagramas) ☐ integrado com o BD.
- Relatórios padrões pré-definidos / customizados.
- Outros clients: HeidiSQL, Toad for SQL Server.

SQL Server Management Studio

iG Ti



Azure Data Studio



- Interface gráfica (client) da Microsoft para se conectar e interagir com o SQL Server e outras plataformas de dados.
- Também chamado de **ADS**.
- Permite administrar o SQL Server e os bancos de dados.
- Possui editor de queries, mas bem menos *Wizards* que o SSMS.
- Não possui construtor de modelos de dados (diagramas).
- Trabalha com o conceito de *Jupyter Notebooks*

Azure Data Studio Notebook



The screenshot shows the Azure Data Studio interface with a notebook titled "deploy-bdc-aks".

Prerequisites:
Ensure the following tools are installed and added to PATH before proceeding.

Tools	Description	Installation
Azure CLI	Command-line tool for managing Azure services. Used to create AKS cluster	Installation
kubectl	Command-line tool for monitoring the underlying Kuberentes cluster	Installation
azdata	Command-line tool for installing and managing a big data cluster	Installation

Check dependencies:

```
1 import pandas,sys,os,getpass,time,json,html
2 pandas_version = pandas.__version__.split('.')
3 pandas_major = int(pandas_version[0])
4 pandas_minor = int(pandas_version[1])
5 pandas_patch = int(pandas_version[2])
6 if not (pandas_major > 0 or (pandas_major == 0 and pandas_minor > 24) or (pandas_major == 0 and pandas_minor == 24 and pandas_patch > 0)):
7     sys.exit('Please upgrade the Notebook dependency before you can proceed, you can do it by running the "Reinstall Notebook depen...')
```

Azure Data Studio Notebook



The screenshot shows the Azure Data Studio Notebook interface. The top navigation bar includes File, Edit, View, Help, and a title bar for 'deploy-bdc-aks - Azure Data Studio'. The left sidebar has icons for File, Connections, Servers, and Azure. The main area has tabs for 'SQLQuery_1 - disconnected' and 'deploy-bdc-aks'. Below the tabs are buttons for Code, Text, Kernel (Loading kernels...), Attach To (Loading contexts...), Trusted, Run Cells, and Clear Results.

Required information

```
1 env_var_flag = "AZDATA_NB_VAR_BDC_CONTROLLER_PASSWORD" in os.environ
2 if env_var_flag:
3     mssql_password = os.environ["AZDATA_NB_VAR_BDC_CONTROLLER_PASSWORD"]
4 else:
5     mssql_password = getpass.getpass(prompt = 'SQL Server 2019 big data cluster controller password')
6     if mssql_password == "":
7         sys.exit(f'Password is required.')
8     confirm_password = getpass.getpass(prompt = 'Confirm password')
9     if mssql_password != confirm_password:
10        sys.exit(f'Passwords do not match.')
11 print('You can also use the same password to access Knox and SQL Server.')

You can also use the same password to access Knox and SQL Server.
```

Azure settings

Subscription ID: visit [here](#) to find out the subscriptions you can use, if you leave it unspecified, the default subscription will be used.

VM Size: visit [here](#) to find out the available VM sizes you could use.

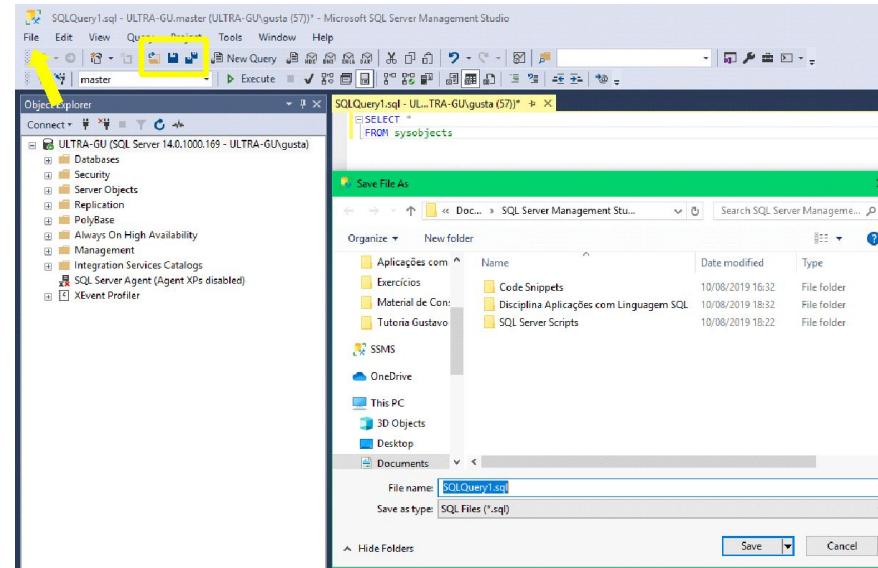
Region: visit [here](#) to find out the Azure regions where the Azure Kuberettes Service is available.

```
[9] 1 if env_var_flag:
2     azure_subscription_id = os.environ["AZDATA_NB_VAR_BDC_AZURE_SUBSCRIPTION"]
3     azure_vm_size = os.environ["AZDATA_NB_VAR_BDC_AZURE_VM_SIZE"]
4     azure_region = os.environ["AZDATA_NB_VAR_BDC_AZURE_REGION"]
5     azure_vm_count = int(os.environ["AZDATA_NB_VAR_BDC_VM_COUNT"])
6 else:
7     azure_subscription_id = "1b294139-ea87-49f9-a09e-a530e697a5f0"
8     azure_vm_size = "Standard_E2s_v3"
9     azure_region = "eastus"
10    azure_vm_count = int(5)
```

Scripts, Projetos e Soluções



- Comandos T-SQL podem ser salvos em um arquivo texto: **script**.
- Normalmente com a extensão “**.sql**”.
- No SSMS, isso é feito no menu **File** ou nos **botões** de salvar da barra de ferramentas.
- Scripts podem ser abertos de dentro do SSMS.
- Scripts podem ser organizados em **Projetos e Soluções**
 - Uma solução pode conter vários projetos, scripts e conexões



Próxima Aula



- Instalação das Ferramentas Client

A Linguagem SQL

CAPÍTULO 2. AULA 2.5.1. INSTALAÇÃO E OVERVIEW DO SQL SERVER MANAGEMENT STUDIO

PROF. GUSTAVO AGUILAR

Nesta Aula

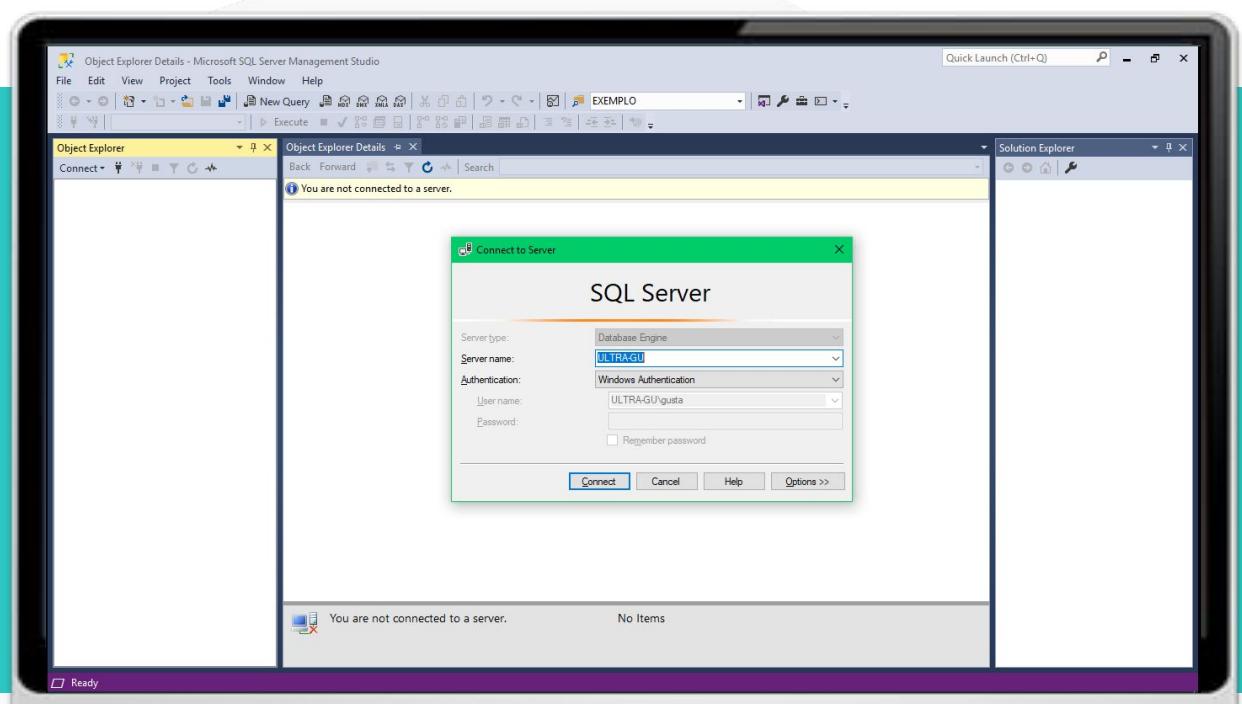


- Instalação do SQL Server Management Studio
- Overview da Utilização do SSMS

Instalação e Overview do Management Studio



Demo



Próxima Aula



- Instalação e Overview do Azure Data Studio

A Linguagem SQL

CAPÍTULO 2. AULA 2.5.1. INSTALAÇÃO E OVERVIEW DO AZURE DATA STUDIO

PROF. GUSTAVO AGUILAR

Nesta Aula

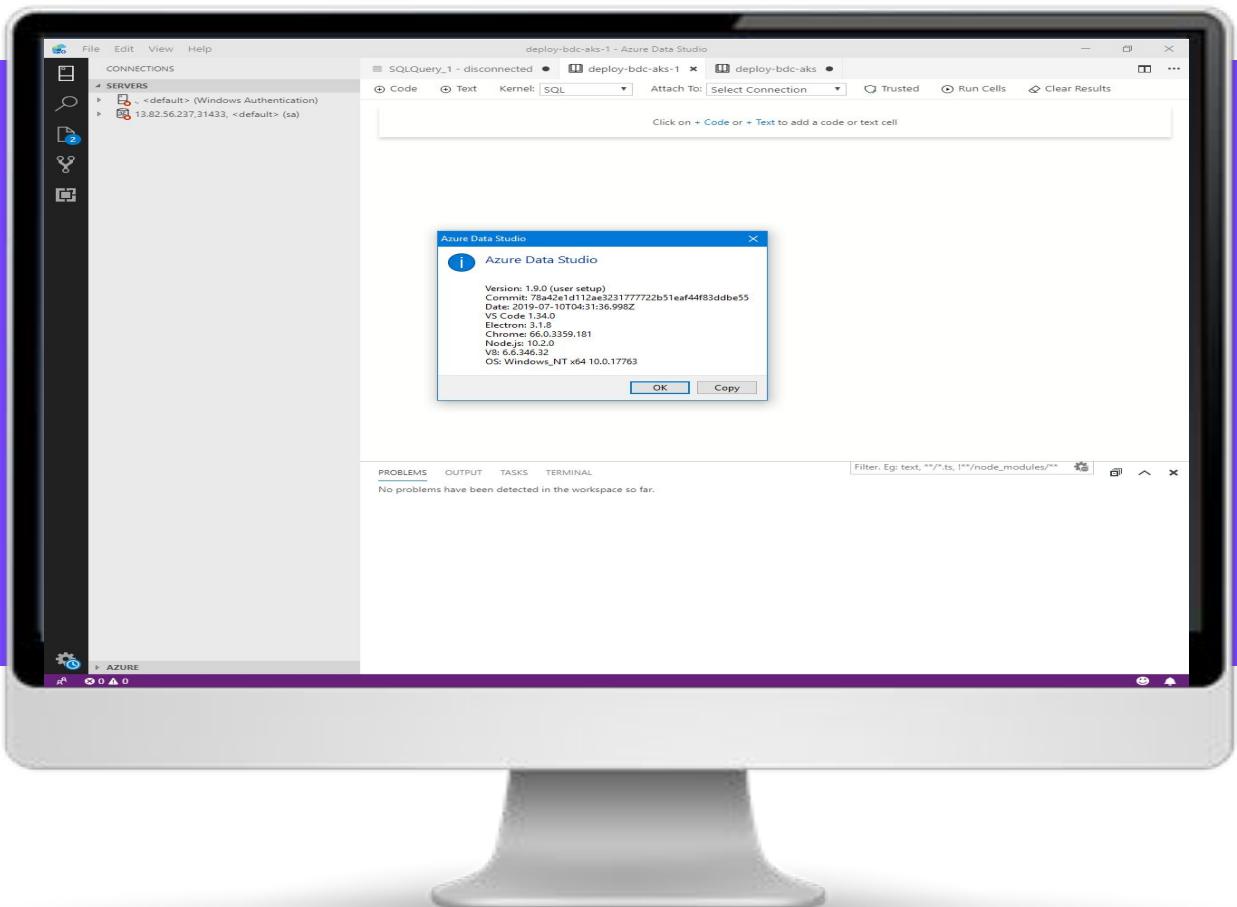


- Instalação do Azure Data Studio
- Overview da Utilização do Azure Data Studio

Instalação e Overview do Azure Data Studio



 Demo



Próxima Aula



- ❑ Banco de Dados de Exemplo AdventureWorks

A Linguagem SQL

CAPÍTULO 2. AULA 2.6. BANCO DE DADOS DE EXEMPLO ADVENTUREWORKS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Banco de Dados de Exemplo AdventureWorks
- Restore do AdventureWorks2019

Banco de Dados de Exemplo

AdventureWorks



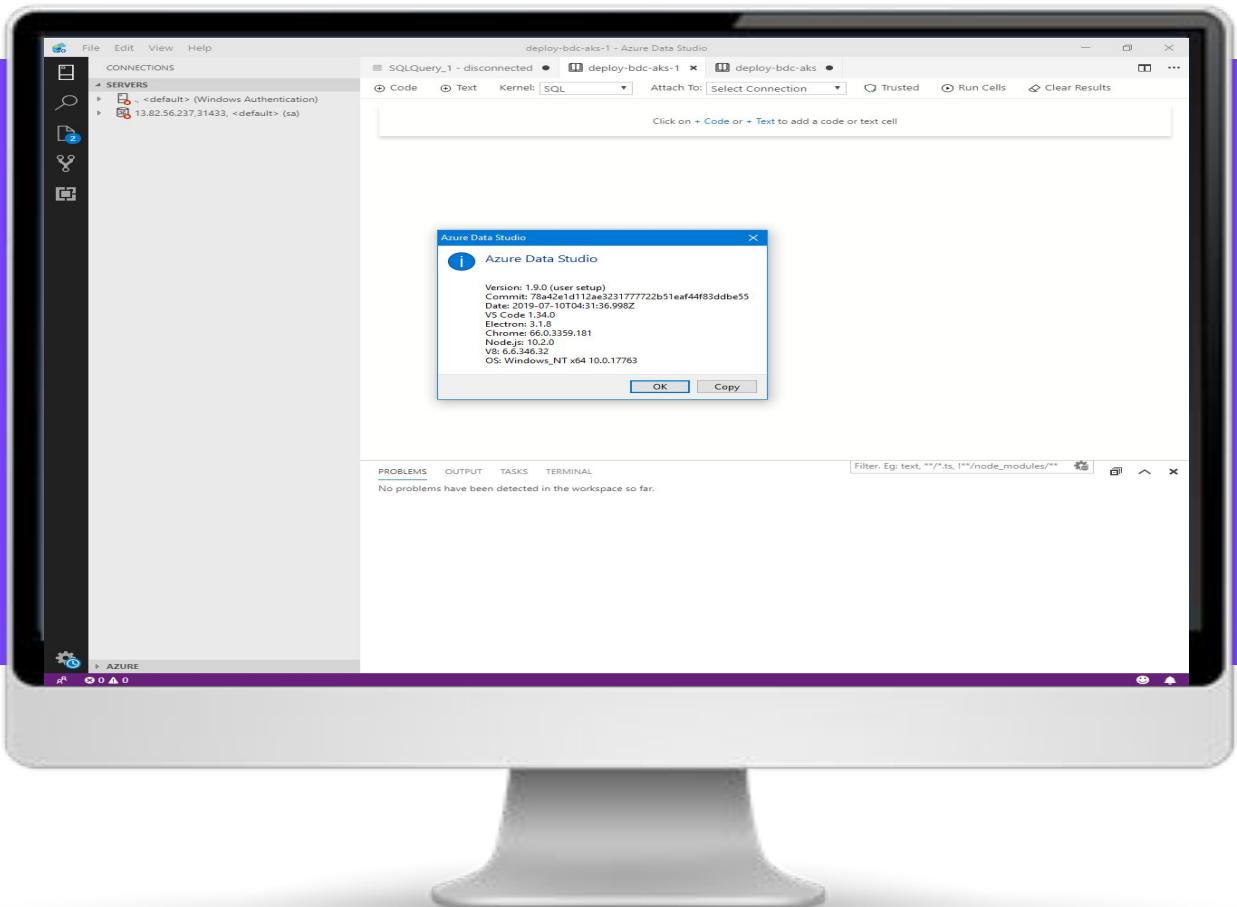
- Banco de dados de exemplo disponibilizado pela Microsoft em
<https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks>
- Empresa de produção e venda de produtos para aventuras.
- Já populado com dados fictícios.
- O modelo de dados deste banco de dados pode ser encontrado em
<https://improveandrepeat.com/2019/02/use-the-adventureworks-sample-database-for-your-examples>.

Restore do AdventureWorks2019

IGTI



Demo



Próxima Aula



- Capítulo 3. Linguagem de Definição de Dados (DDL)

A Linguagem SQL

Capítulo 3. Linguagem de Definição de Dados (DDL)

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 3. AULA 3.1. CRIAÇÃO DE ESTRUTURAS DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Linguagem de Definição de Dados (DDL)
- Script DDL
- Criação de Estruturas de Dados
- Criação de Outros Tipos de Objetos

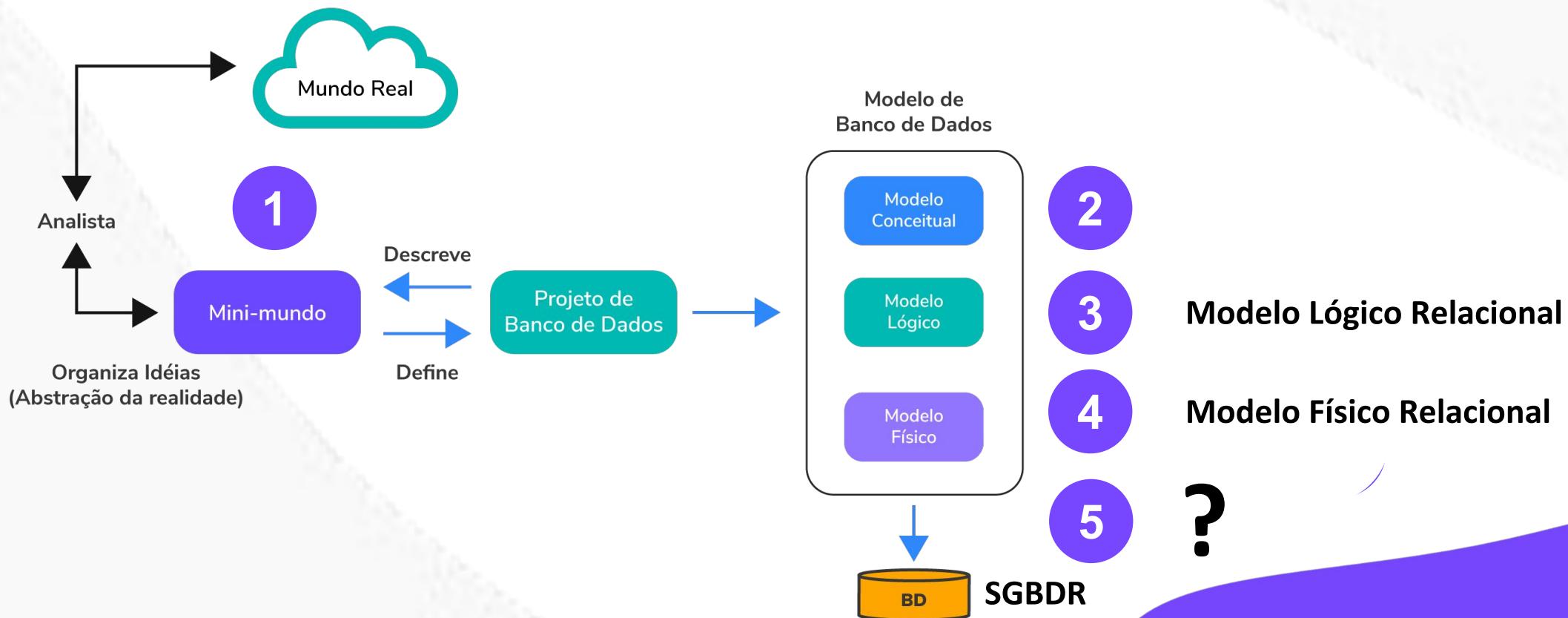
Linguagem de Definição de Dados



- **Data Definition Language** ☐ DDL
- **Classe da Linguagem SQL** com comandos que permitem criar, alterar e excluir objetos de banco de dados.
- Em um **Projeto de Banco de Dados** ☐ usada na etapa de geração do esquema físico no banco de dados.

Linguagem de Definição de Dados

IGTI



Script DDL



- Conjunto de várias instruções DDL.

The screenshot shows a Windows application window titled "ORACLE Schema Generation Preview". The main area contains Oracle SQL DDL code. The code creates two tables: "ATENDENTE" and "CLIENTE". The "ATENDENTE" table has columns "COD_ATENDENTE" (NUMBER(3) NOT NULL) and "NOM_ATENDENTE" (VARCHAR2(50) NOT NULL). It also includes an ALTER TABLE statement to add a primary key constraint named "PK_ATEND" on the "COD_ATENDENTE" column. The "CLIENTE" table has columns "COD_CLIENTE" (NUMBER(5) NOT NULL), "NOM_CLIENTE" (VARCHAR2(50) NOT NULL), "DSC_ENDERECO_COBRANCA" (VARCHAR2(100) NOT NULL), and "NUM_TELEFONE" (NUMBER(12) NOT NULL). The bottom of the window shows a "Table Filter: 7/7" and buttons for "Generate..." and "Close".

```
CREATE TABLE ATENDENTE (
    COD_ATENDENTE      NUMBER(3) NOT NULL,
    NOM_ATENDENTE       VARCHAR2(50) NOT NULL
);

ALTER TABLE ATENDENTE
    ADD ( CONSTRAINT PK_ATEND PRIMARY KEY (COD_ATENDENTE)
) ;

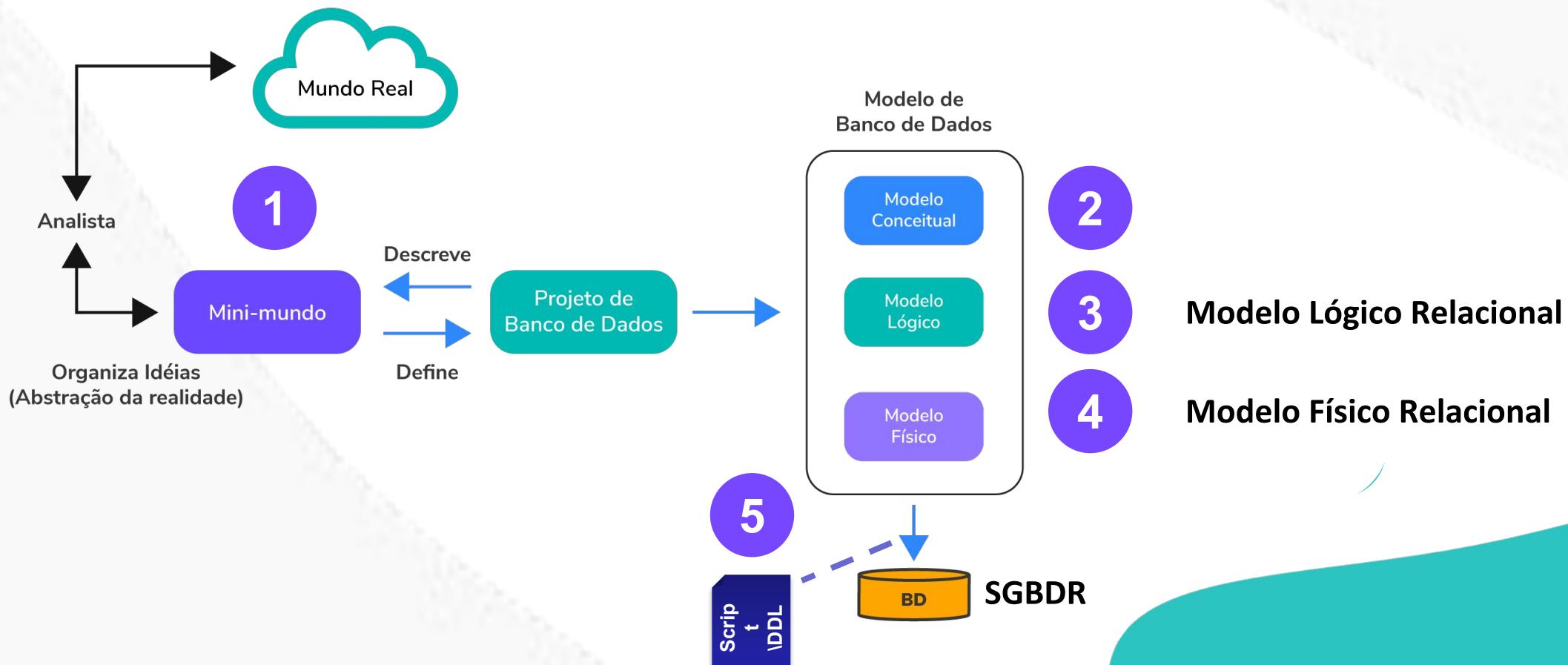
CREATE SEQUENCE SQ_ATEND_COD_ATENDENTE
NOCACHE
NOCYCLE;

CREATE TABLE CLIENTE (
    COD_CLIENTE         NUMBER(5) NOT NULL,
    NOM_CLIENTE          VARCHAR2(50) NOT NULL,
    DSC_ENDERECO_COBRANCA VARCHAR2(100) NOT NULL,
    NUM_TELEFONE         NUMBER(12) NOT NULL
);

Table Filter: 7/7
```

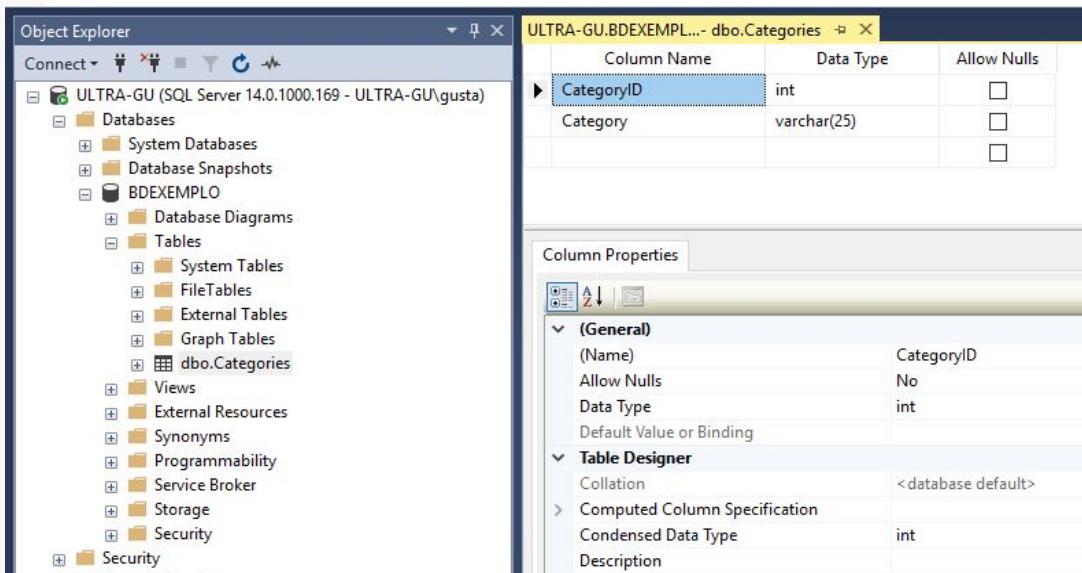
Script DDL

IGTI



Script DDL

- A maioria dos SGBDs fornece **interface gráfica** para implementação do modelo de dados físico.
 - Usam scripts DDL em background.



Criação de Estruturas de Dados



- Através da instrução SQL **CREATE**
- Criação de banco de dados
 - CREATE DATABASE <Nome_do_Banco_de_Dados>
 - Cada SGBD implementa os parâmetros adicionais necessários e específicos da sua plataforma.

```
CREATE DATABASE [SQLServer] ON PRIMARY
( NAME = N'SQLServer', FILENAME = N'D:\DATA\SQLServer.mdf',
  SIZE=102400KB, MAXSIZE=1024000KB, FILEGROWTH=102400KB)
LOG ON
( NAME = N'SQLServer_log', FILENAME = N'E:\LOG\SQLServer_log.ldf',
  SIZE=51200KB, MAXSIZE=3072000KB, FILEGROWTH=51200KB)
GO
ALTER DATABASE [SQLServer] SET COMPATIBILITY_LEVEL = 140
GO
ALTER DATABASE [SQLServer] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [SQLServer] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [SQLServer] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [SQLServer] SET RECOVERY FULL
GO
```

```
create database Oracle
controlfile reuse
logfile 'D:\ORAWIN95\DATABASE\log1orcl.ora'
size 200K reuse,
'D:\ORAWIN95\DATABASE\log2orcl.ora'
size 200K reuse
datafile 'D:\ORAWIN95\DATABASE\sys1orcl.ora'
size 20M reuse autoextend on
next 10M maxsize 200M
character set WE8ISO8859P1;
```

Criação de Estruturas de Dados



- Criação de tabelas

```
CREATE TABLE <Nome_da_Tabela>
(
    column1 datatype [ NULL | NOT NULL ],
    column2 datatype [ NULL | NOT NULL ], ....
);
```

- Cada SGBD implementa os parâmetros adicionais necessários e específicos para a criação de tabelas no seu banco de dados.

Criação de Estruturas de Dados



- Também através da instrução SQL **CREATE**
 - Usuários
 - Índices
 - Visões
 - Chaves estrangeiras
 - Constraints
 - Sequences
 - Sinônimos, etc.

CREATE SYNONYM <Nome_do_Sinônimo> FOR <Nome_do_Objeto>;

Próxima Aula



- Demonstração: Criação de Estruturas de Dados

A Linguagem SQL

CAPÍTULO 3. AULA 3.1.1. DEMONSTRAÇÃO: CRIAÇÃO DE ESTRUTURAS DE DADOS

PROF. GUSTAVO AGUILAR

Criação de Estrutura de Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTR)'. The Solution Explorer on the right lists various SQL scripts such as '01-Criar Banco.sql', '02-Criar Tabelas.sql', etc. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database and enabling full-text search. The status bar at the bottom indicates 'Connected (1/1)'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'AdventureWorks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\DATA\AdventureWorks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\LOG\AdventureWorks2017.ldf' , SIZE = 64 , MAXSIZE = 1024 , FILEGROWTH = 64 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Alteração de Estruturas de Dados

A Linguagem SQL

CAPÍTULO 3. AULA 3.2. ALTERAÇÃO DE ESTRUTURAS DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Alteração de Estruturas de Dados
- Demonstração

Alteração de Estruturas de Dados



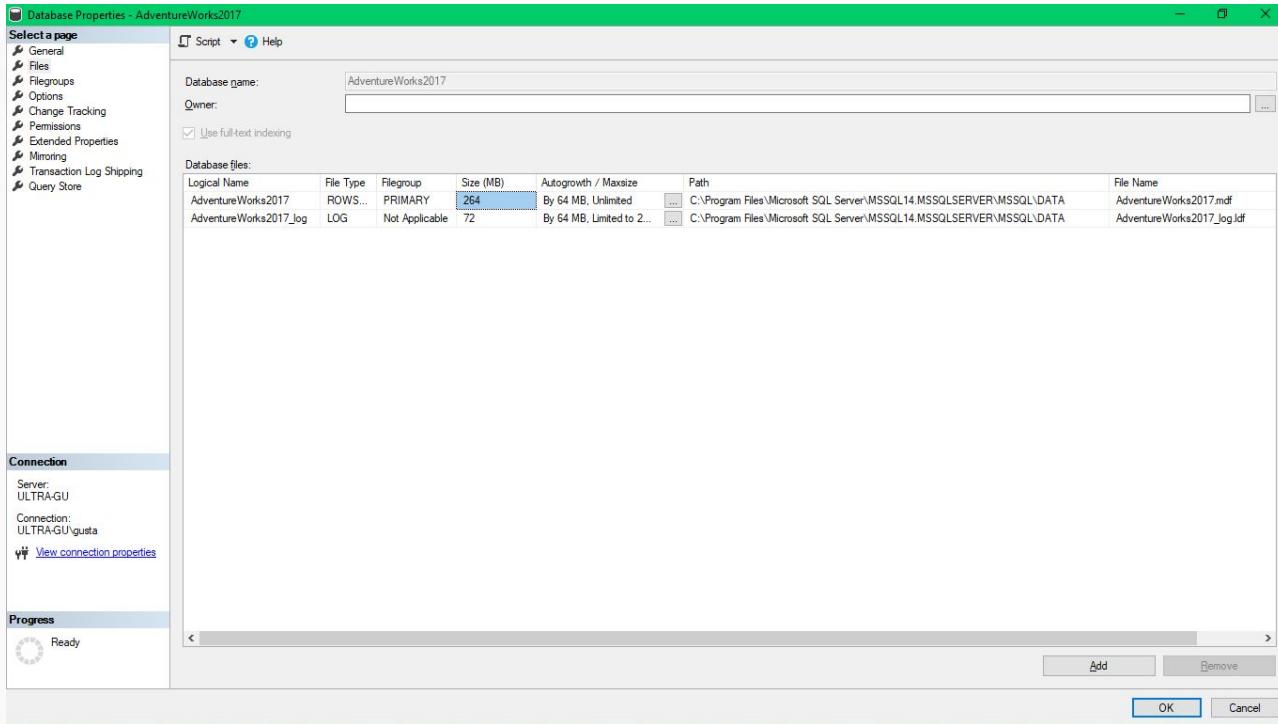
- Comando DDL **ALTER**

- Evitar recriação das estruturas de dados: bancos de dados, tabelas, etc.
- Alterar as estruturas de dados que já estejam criadas.
- Desde que não viole as regras do SGBD / modelo de dados
 - Regras de unicidade: alterar campo que possua valores nulos para NOT NULL
 - Regras de integridade referencial: setar para NULL chaves estrangeiras NOT NULL
 - Adicionar coluna NOT NULL em uma tabela já populada
 - Alterar tipo de dados de uma coluna para um tipo de dados incompatível
 - Reduzir o tamanho do tipo de dados de uma coluna que possua valores que extrapolem esse tamanho
 - Etc.

Alteração de Estruturas de Dados



- SGBDs fornecem diversas interfaces gráficas para fazer alterações.



Alteração de Estruturas de Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The Object Explorer on the left shows the connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU.gusta (56))'. The Solution Explorer on the right lists various SQL scripts such as '01-Criar Banco.sql' through 'SQLQuery4.sql'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\MDF\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LFN\Adventureworks2017_log.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Remoção de Estruturas de Dados

A Linguagem SQL

CAPÍTULO 3. AULA 3.3. REMOÇÃO DE ESTRUTURAS DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Remoção de Estruturas de Dados
- Demonstração

Remoção de Estruturas de Dados

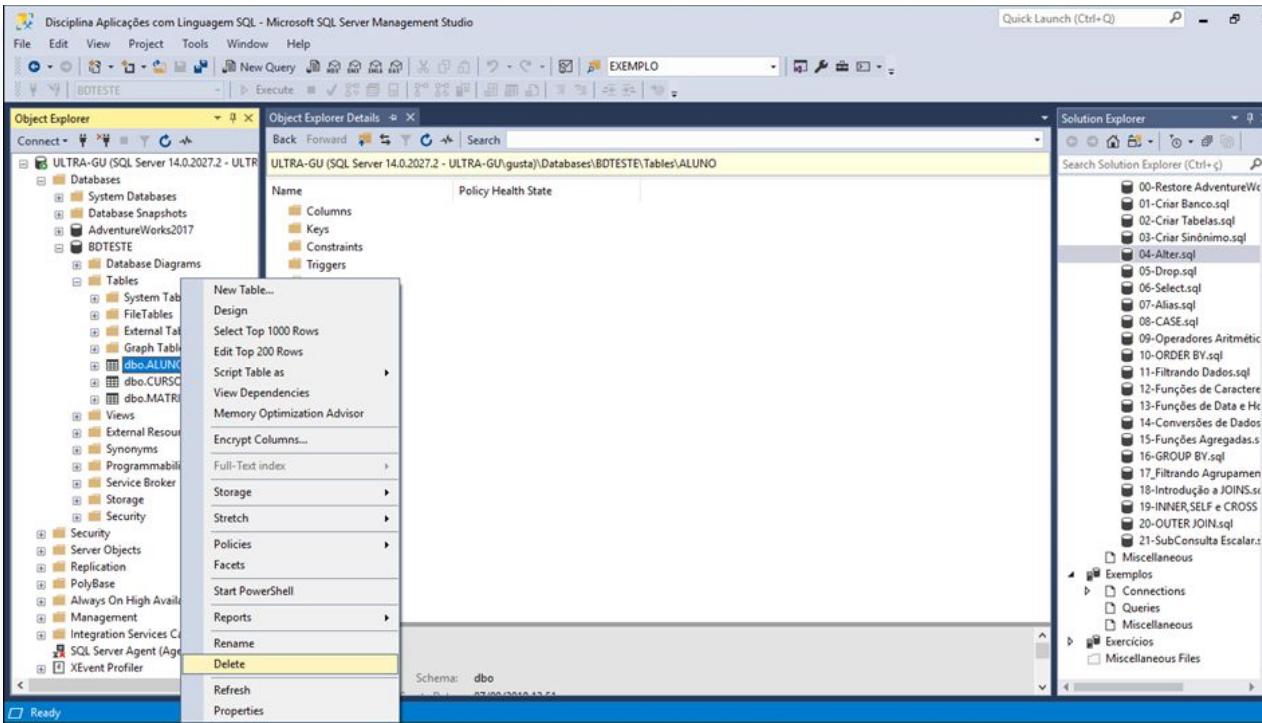


- Comando DDL **DROP**

- Excluir estruturas de dados e outros objetos do banco de dados
- Deve ser usado com **muita cautela**, para não remover objetos por engano
- Para ocorrer a remoção, não pode haver violação das regras do SGBD / modelo de dados
 - Excluir um banco de dados / tabela que esteja sendo usada
 - Regras de integridade referencial: excluir uma coluna que seja referenciada por uma chave estrangeira com constraint FK
 - Etc.

Remoção de Estruturas de Dados

- SGBDs fornecem diversas interfaces gráficas para fazer remoções.



Remoção de Estruturas de Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU)'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 4. Linguagem de Manipulação de Dados (DML)

A Linguagem SQL

Capítulo 4. Linguagem de Manipulação de Dados (DML)

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 4. AULA 4.1. SELECIONANDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Introdução à Linguagem de Manipulação de Dados (DML)
- A Instrução SELECT
- Alias de Coluna
- Alias de Tabela
- Expressão CASE

Introdução à Linguagem de Manipulação de Dados



- **Data Manipulation Language** ☐ **DML**
- **Classe da Linguagem SQL** com comandos que permitem:
 - Inserir dados (novas linhas / tuplas)
 - Atualizar dados (linhas / tuplas existentes)
 - Excluir dados: remover linhas / tuplas existentes
 - Selecionar dados: recuperar dados armazenados para exibição e/ou processamento.

A Instrução SELECT



- Pertencente à classe **DML** da Linguagem SQL
- No padrão **ODMG** □ classe **DQL** (*Data Query Language*)
- Utilizada para retornar (consultar) os dados armazenados no banco
- Baseada na operação de **Projeção** da **Álgebra Relacional**
- Em sua forma mínima é composta de 2 cláusulas:

SELECT <relação de colunas ou * >

FROM

Projeção

■	■	■
■	■	■
■	■	■

Alias de Coluna

- Nome alternativo (apelido) para colunas
- Existentes apenas em tempo de execução (não é persistido)
- Criado usando-se a **cláusula AS** após o nome da coluna, colocando-se o alias desejado em seguida.

```
SELECT Name AS Nome_do_Produto, ProductNumber AS Número_Produto  
FROM Production.Product;
```

Alias de Coluna

IGTI

- Efeito prático ☐ modificar o cabeçalho (label) da coluna que é exibido no resultado da query.

	Name	Owner	Type	Created_datetime	Column_name	Type	Computed	Length	Prec	Scale	Nullable
1	Product	dbo	user table	2017-10-27 14:33:01.813	ProductID	int	no	4	10	0	no
2	Name		Name		Name	name	no	100			no
3	ProductNumber		nvarchar		ProductNumber	nvarchar	no	50			no
4	MakeFlag		Flag		MakeFlag	flag	no	1			no
5	FinishedGoodsFlag		Flag		FinishedGoodsFlag	flag	no	1			no
6	Color		nvarchar		Color	nvarchar	no	30			yes

```
SELECT Name AS Nome_do_Produto,
       ProductNumber AS Número_do_Produto
    FROM Production.Product
GO
```

Nome_do_Produto	Número_do_Produto
Adjustable Race	AR-5381
Bearing Ball	BA-8327
BB Ball Bearing	BE-2349
Headset Ball Bearings	BE-2908
Blade	BL-2036

Alias de Tabela



- Nome alternativo (apelido) para tabelas
- Existentes apenas em tempo de execução (não é persistido)
 - Diferente do sinônimo que é persistido (criado como objeto)
- Uso bem mais amplo: usado em outras cláusulas da instrução SELECT, como por exemplo, nas cláusulas WHERE, GROUP BY, ORDER BY etc., que serão vistas mais à frente.

Alias de Tabela



- Criado usando-se a cláusula AS (ou apenas espaço) após o nome da tabela na cláusula FROM, colocando-se o alias desejado em seguida.

--Alias de Tabela Criado **com a Cláusula AS**

```
SELECT Name, ProductNumber FROM Production.Product AS P;
```

--Alias de Tabela Criado **sem a Cláusula AS**

```
SELECT Name, ProductNumber FROM Production.Product P;
```

Alias de Tabela



- Uma vez definido o alias para uma tabela, ele pode ser referenciado na relação de colunas da cláusula SELECT:

```
SELECT P.Name, P.ProductNumber FROM Production.Product P;
```

- Ao se definir alias de tabelas, não é obrigatório usá-los na relação de colunas, desde que não haja ambiguidade nos nomes delas.
- Alias de tabela pode ser usado em conjunto com alias de colunas:

```
SELECT P.Name AS Nome_do_Produto, P.ProductNumber AS Número_Produto  
FROM Production.Product P;
```

Expressão CASE

- Pode ser utilizada na instrução SELECT para tratamento dos dados
 - Não altera os dados persistidos no BD, **apenas na exibição da query.**
- Compara um valor de entrada a uma lista de possíveis valores
 - Se uma correspondência for encontrada, o primeiro valor correspondente será retornado como resultado da expressão CASE **múltiplas correspondências não são permitidas;**
 - Se nenhuma correspondência for encontrada, a expressão CASE retorna o valor encontrado em uma **cláusula ELSE, se existir;**
 - Se nenhuma correspondência for encontrada e nenhuma cláusula ELSE existir, a expressão CASE retornará **NULL.**

Expressão CASE

IGTI

- Sintaxe:

- Expressão CASE Simples:

CASE Expressão_de_Estrada

*WHEN expressão_de_comparação THEN expressão_resultado [...n]
[ELSE expressão_resultado_else]*

END

- Expressão CASE com Pesquisa Booleana:

CASE

WHEN Expressão_Booleana THEN expressão_resultado [...n]

[ELSE expressão_resultado_else]

END

Próxima Aula



- Demonstração: Selecionando Dados

A Linguagem SQL

CAPÍTULO 4. AULA 4.1.1. DEMONSTRAÇÃO: SELECIONANDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Demonstrações:
 - Instrução SELECT
 - Alias de Coluna
 - Alias de Tabela
 - Expressão CASE

Selecionando Dados

IGTI



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTR...'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database, setting its properties, and enabling full-text search. The status bar at the bottom indicates 'Connected (1/1)'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\...LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\...
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Operadores Aritméticos e de Concatenação

A Linguagem SQL

CAPÍTULO 4. AULA 4.2. OPERADORES ARITMÉTICOS E DE CONCATENAÇÃO

PROF. GUSTAVO AGUILAR

Nesta Aula



- Expressão Calculada
- Operadores Aritméticos
- Operador de Concatenação
- Ordem de Avaliação de Operadores

Expressão Calculada

- Instrução SELECT pode realizar cálculos matemáticos e manipulações dos valores selecionados, em tempo de execução
- Resultado é uma nova coluna no output □ **Expressão Calculada**
 - Inicialmente sem nome: pode-se definir um alias para ela
 - Não altera os dados persistidos
 - Não existe fisicamente na tabela (e nem é criada em tempo de execução)
 - Deve ser escalar (retornar apenas um valor por linha)
- As expressões calculadas são construídas utilizando-se **operadores aritméticos ou de concatenação**

Operadores Aritméticos



- Recursos da Linguagem SQL para realizar **operações aritméticas**
- Usados com colunas do tipo de dados não string
 - Numéricos, datas, hora, moeda etc.
- Operadores do SQL Server:

Operador	Descrição
+	Adição / Concatenação
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)

Operadores Aritméticos



- Ex.: Expressão Calculada Multiplicando Duas Colunas

```
SELECT UnitPrice, OrderQty, (UnitPrice * OrderQty) AS TotalValue
```

```
FROM Sales.SalesOrderDetail;
```

Unit Price	OrderQty	TotalValue
2024,994	1	2024,994
2024,994	3	6074,982
2024,994	1	2024,994
2039,994	1	2039,994
2039,994	1	2039,994
2039,994	2	4079,988
2039,994	1	2039,994
28,8404	3	86,5212
28,8404	1	28,8404
5,70	6	34,20

Operador de Concatenação



- Operador +
- Usado com colunas do tipo de dados string
- Útil para junção de colunas ou construção dinâmica de código
- Ex.: Retornar o NomeCompleto das pessoas

```
SELECT FirstName, MiddleName, LastName,  
FirstName + ' ' + MiddleName + ' ' + LastName AS NomeCompleto  
FROM Person.Person;
```

FirstName	MiddleName	LastName	NomeCompleto
Xavier	C	Adams	Xavier C Adams
Ronald	L.	Adina	Ronald L. Adina
Osarum...	Uwaifiokun	Agbonile	Osarumwense U...
Samuel	N.	Agcaoili	Samuel N. Agcaoili
James	T.	Aguilar	James T. Aguilar
Robert	E.	Ahlering	Robert E. Ahlering
François	P	Ajenstat	François P Ajenstat
Kim	NULL	Nguyen	NULL

Ordem de Avaliação de Operadores



- Quando há mais de 1 operador na expressão a ser executada, o SGBD precisa saber qual operação deve ser realizada primeiro
 - Para isso, existe a **Ordem em que os operadores são avaliados**
 - Cada SGBD possui a sua, mas na grande maioria a ordem é igual
 - Pode-se usar parênteses para “alterar” (indicar) a ordem de avaliação
- No SQL Server:

1º ➔ / (divisão)

2º ➔ * (multiplicação)

3º ➔ % (módulo)

4º ➔ + (adição / concatenação)

5º ➔ - (subtração)

Próxima Aula



- Demonstração: Operadores Aritméticos e de Concatenação

A Linguagem SQL

CAPÍTULO 4. AULA 4.2.1. DEMONSTRAÇÃO: OPERADORES ARITMÉTICOS E DE CONCATENAÇÃO

PROF. GUSTAVO AGUILAR

Demonstração: Operadores

Aritméticos e de Concatenação



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU)'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

The Solution Explorer on the right lists various SQL scripts, including '01-Criar Banco.sql', '02-Criar Tabelas.sql', and '09-Operadores Aritméticos e Concatenação.sql'. The status bar at the bottom indicates 'Connected (1/1)'.

Próxima Aula



- Ordenando Dados

A Linguagem SQL

CAPÍTULO 4. AULA 4.3. ORDENANDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- A Cláusula ORDER BY
- Tipos de Ordenação
- ORDER BY Composto
- Alias de Coluna na Cláusula ORDER BY

A Cláusula ORDER BY



- Além das cláusulas *SELECT* e *FROM*, uma instrução *SELECT* pode ter também a cláusula **ORDER BY**
- Utilizada para **ordenar** os dados retornados pela instrução *SELECT*
- Coluna(s) na cláusula *ORDER BY* não necessariamente precisa(m) estar na cláusula *SELECT*
- Sintaxe: a última cláusula em uma instrução *SELECT*

SELECT FirstName, MiddleName, LastName

FROM Person.Person

ORDER BY FirstName

Tipos de Ordenação

- **Default:** ordenação feita de forma **ascendente** (A à Z, 0 à 9)
- **DESC:** cláusula para fazer ordenação **descendente** (Z à A, 9 à 0)
- **ASC:** cláusula para explicitar a ordenação **ascendente**
- **Ex.:** classificando descendemente pelo primeiro nome

SELECT FirstName, MiddleName, LastName

FROM Person.Person

ORDER BY FirstName DESC;

FirstName	MiddleName	LastName
Zoe	L	Bailey
Zoe	L	Bell
Zoe	L	Brooks
Zoe	NULL	Cook
Zoe	L	Cooper
Zoe	A	Cox

FirstName	MiddleName	LastName
A.	Francesca	Leonetti
A.	Scott	Wright
A. Scott	NULL	Wright
Aaron	L	Wright
Aaron	C	Yang
Aaron	M	Young

ORDER BY FirstName ASC

ORDER BY Composto

- Pode-se usar mais de uma coluna na cláusula ORDER BY
- Cada campo com o tipo de ordenação (ASC / DESC) desejada
- Ex.: classificando ascendentemente pelo primeiro nome e descendente pelo último nome

```
SELECT FirstName, MiddleName, LastName  
FROM Person.Person  
ORDER BY FirstName ASC, LastName DESC
```

Ordenação descendente pela coluna LastName

FirstName	MiddleName	LastName
A.	Scott	Wright
A.	Francesca	Leonetti
A. Scott	NULL	Wright
Aaron	A	Zhang
Aaron	M	Young
Aaron	C	Yang
Aaron	L	Wright
Aaron	L	Washington
Aaron	V	Wang
Aaron	NULL	Simmons
Aaron	J	Shama
Aaron	NULL	Shan

Alias de Coluna na Cláusula



ORDER BY

- Pode-se usar o alias da(s) coluna(s) na cláusula ORDER BY
 - Ao invés de usar o nome das colunas.

--Usando *Alias* de Coluna no ORDER BY

```
SELECT Name AS Nome_do_Produto, ProductNumber AS Número_do_Produto  
FROM Production.Product  
ORDER BY Nome_do_Produto ASC;
```

- Possível devido à **ordem de execução de comandos** no SQL Server

5.	SELECT	<select list>
1.	FROM	<table source>
2.	WHERE	<search condition>
3.	GROUP BY	<group by list>
4.	HAVING	<search condition>
6.	ORDER BY	<order by list>

Próxima Aula



- Demonstração: Ordenando Dados

A Linguagem SQL

CAPÍTULO 4. AULA 4.3.1. DEMONSTRAÇÃO: ORDENANDO DADOS

PROF. GUSTAVO AGUILAR

Demonstração: Ordenando Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right pane shows a Solution Explorer with various SQL files listed, and the bottom right corner indicates the script is ready to execute.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\M...LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\...GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO
ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Filtrando Dados

A Linguagem SQL

CAPÍTULO 4. AULA 4.4. FILTRANDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula

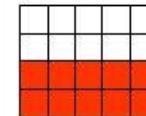


- A Cláusula WHERE
- A Cláusula TOP
- A Cláusula DISTINCT
- Operadores de Comparaçāo
- Operadores Lógicos
- Ordem de Avaliação dos Operadores
- Alias de Coluna na Cláusula WHERE

A Cláusula WHERE

- Cláusula usada para realizar **operações de seleção (restrições)**
 - Restringir as tuplas (linhas/dados) retornados pela operação de projeção (cláusula SELECT).
- **Sintaxe:** usada após a cláusula FROM.
- Utiliza-se de operadores lógicos ou de comparação para filtrar os resultados retornados.

Seleção



--Listar Produtos da Cor Preta (Black)

SELECT Name, Color

FROM Production.Product

WHERE Color = 'Black'

ORDER BY Name;

Name	Color
Chainring	Black
Full-Finger Gloves, L	Black
Full-Finger Gloves, M	Black
Full-Finger Gloves, S	Black
Half-Finger Gloves, L	Black
Half-Finger Gloves, M	Black
Half-Finger Gloves, S	Black
HL Crankarm	Black
HL Crankset	Black
HL Mountain Frame - Black, 38	Black
HL Mountain Frame - Black, 42	Black

A Cláusula TOP

- Usada em conjunto com a cláusula SELECT para limitar a quantidade de linhas retornadas
- Sintaxe T-SQL (SQL Server)
 - **SELECT TOP (N)**: retorna as N primeiras linhas encontradas.
 - A cláusula ORDER BY impacta diretamente no resultado retornado
- Ex.: retornar os 5 primeiros produtos (na ordem alfabética):

```
SELECT TOP (5) Name AS Nome_Produto  
FROM Production.Product  
ORDER BY Nome_Produto;
```

Nome_Produto
Adjustable Race
All-Purpose Bike Stand
AWC Logo Cap
BB Ball Bearing
Bearing Ball

A Cláusula DISTINCT

- Elimina os resultados repetidos retornados (tuplas / linhas repetidas) pela instrução SELECT
- Atua em todas as colunas da cláusula SELECT
- **Sintaxe:** *SELECT DISTINCT Coluna1, Coluna2, etc / **
- Ex.: cores de produtos existentes

```
SELECT DISTINCT Color AS Cores_de_Produtos  
FROM Production.Product  
WHERE Color IS NOT NULL  
ORDER BY Cores_de_Produtos;
```

Cores_de_Produtos
Black
Blue
Grey
Multi
Red
Silver
Silver/Black
White
Yellow

Operadores de Comparação



- Usados para comparar colunas ou expressões nas cláusulas de filtro da Linguagem SQL
- Cada SGBD procura manter o caractere indicado no padrão ISO
 - Nada impede que existam outros operadores ou operadores representados por caracteres diferentes em alguns SGBDs.

Operador de Comparação	Significado
=	Igual a
>	Maior que
<	Menor que
>=	Maior que ou igual a
<=	Menor que ou igual a
<>	Diferente de
!=	Diferente de (não é padrão ISO)
>!	Não é maior que (não é padrão ISO)
<!	Não é menor que (não é padrão ISO)

Operadores Lógicos

- Testam a verdade de alguma condição, retornando um tipo de dados booleano com o valor TRUE, FALSE ou UNKNOWN
- Permitem que mais de uma condição (filtro) possa ser feita na operação de restrição
- Podem ser usados em conjunto com os operadores de comparação.

Operadores Lógicos do SQL Server □

Operador	Significado
ALL	TRUE se todas as comparações forem verdadeiras
AND	TRUE se ambas as expressões booleanas forem verdadeiras
ANY	TRUE se qualquer um de um conjunto de comparações for verdadeiro
BETWEEN	TRUE se o operando/expressão estiver dentro de um intervalo
EXISTS	TRUE se uma subconsulta contiver quaisquer linhas
IN	TRUE se o operando for igual a um de uma lista de valores/expressões
LIKE	TRUE se o operando corresponder a um padrão
NOT	Inverte o valor de qualquer outro operador booleano
OR	TRUE se qualquer expressão booleana for TRUE
SOME	TRUE se algumas das comparações forem verdadeiras

Operadores Lógicos

- Ex.: Listar Produtos da Cor Preta (Black) ou Prata (Silver)

SELECT Name, Color

FROM Production.Product

WHERE Color = 'Black'

OR Color = 'Silver'

ORDER BY Name;

Name	Color
Chain	Silver
Chainring	Black
Chainring Bolts	Silver
Chainring Nut	Silver
Freewheel	Silver
Front Brakes	Silver
Front Derailleur	Silver
Front Derailleur Cage	Silver
Front Derailleur Linkage	Silver
Full-Finger Gloves, L	Black
Full-Finger Gloves, M	Black

Ordem de Avaliação dos Operadores



- Da mesma forma que as cláusulas da Linguagem SQL e os operadores aritméticos / de comparação possuem uma ordem pré-definida de processamento, os operadores lógicos e de comparação também a tem.
- Operadores lógicos e de comparação são avaliados após os operadores aritméticos e de concatenação.
- Entre si, os operadores lógicos e de comparação são avaliados na seguinte ordem (sequência) □

1º ➔ =, >, <, <=, <, !=, !, !<, !> (operadores de comparação)
2º ➔ NOT
3º ➔ AND
4º ➔ ALL, ANY, BETWEEN, IN, LIKE, OR, SOME

Operadores de Comparaçāo



- Pode-usar parênteses para determinar a ordem desejada de avaliação dos operadores lógicos ou de comparação
- Ex.: operador OR sendo avaliado antes do operador AND

-- Produtos com Nome que Iniciam com 'Chain' e que Sejam da Cor Preta ou Prata

SELECT Name,Color

FROM Production.Product

WHERE Name LIKE 'Chain%'

AND (Color = 'Black' OR Color = 'Silver')

ORDER BY Name;

Alias de Coluna na Cláusula WHERE



- Não é possível usar alias de coluna na cláusula WHERE
 - Cláusula WHERE é processada antes da cláusula SELECT

5.	SELECT	<select list>
1.	FROM	<table source>
2.	WHERE	<search condition>
3.	GROUP BY	<group by list>
4.	HAVING	<search condition>
6.	ORDER BY	<order by list>

- O alias de coluna ainda não está definido quando o filtro é aplicado
 - Será definido para o contexto de execução da query, após a cláusula SELECT ser processada.

Próxima Aula



- Demonstração: Filtrando Dados

A Linguagem SQL

CAPÍTULO 4. AULA 4.4.1. DEMONSTRAÇÃO: FILTRANDO DADOS

PROF. GUSTAVO AGUILAR

Demonstração: Filtrando Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right-hand pane shows a Solution Explorer with various SQL files listed under a folder named 'SQL Query4.sql'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 512 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 512 , FILEGROWTH = 10 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Funções de Caracteres, de Data e Hora

A Linguagem SQL

CAPÍTULO 4. AULA 4.5. FUNÇÕES DE CARACTERES, DE DATA E HORA

PROF. GUSTAVO AGUILAR

Nesta Aula



- ❑ Introdução à Funções de Caracteres
- ❑ Exemplos de Funções de Caracteres
- ❑ Uso de Funções de Caracteres em Conjunto
- ❑ Tipo de Dados de Data e Hora
- ❑ Funções Básicas de Data e Hora
- ❑ Funções Aritméticas para Data e Hora

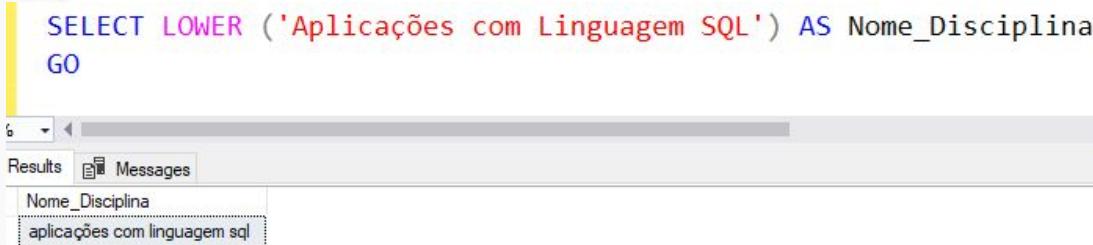
Introdução à Funções de Caracteres



- Recurso de programação da Linguagem SQL utilizado para modificar ou realizar um tratamento específico em uma cadeia de caracteres (string).
- Nos SGBDs, essas funções já vêm prontas para serem utilizadas junto às queries SQL □ chamadas de **built-in string functions**.
- Recebem como **parâmetro de entrada** uma **string**, uma **coluna** ou uma **expressão**, e retornam os dados alterados / tratados em um formato do tipo texto.

LOWER e UPPER

- **LOWER (string)**: converte a string passada como parâmetro em caracteres **minúsculos**.



```
SELECT LOWER ('Aplicações com Linguagem SQL') AS Nome_Disciplina
GO
```

The screenshot shows a SQL query being run in SSMS. The query uses the LOWER function to convert the string 'Aplicações com Linguagem SQL' to its lowercase equivalent, 'aplicações com linguagem sql'. The results are displayed in a table with one column, 'Nome_Disciplina', containing the converted value.

Nome_Disciplina
aplicações com linguagem sql

- **UPPER (string)**: converte a string passada como parâmetro em caracteres **maiúsculos**.

CHARINDEX

- **CHARINDEX (string_a_encontrar, string_onde_procurar, [início])**

- Procura uma string em uma outra string, retornando a posição inicial na string onde foi feita a busca (se encontrada).
- Opcionalmente, pode-se especificar a posição de início a partir da qual a busca na string deve ser feita.

```
SELECT CHARINDEX ('SQL', 'Aplicações com Linguagem SQL') AS Inicio_String_SQL,  
      CHARINDEX ('T-SQL', 'Aplicações com Linguagem SQL') AS String_Inexiste  
GO  
--Especificando uma Posição de Início para a Busca  
SELECT CHARINDEX ('SQL', 'Aplicações com Linguagem SQL',27) AS A_Partir_da_27ª  
GO
```

Início_String_SQL	String_Inexiste
26	0

A_Partir_da_27ª
0

Funções de Caracteres em Conjunto

IGTI

- As funções de caracteres podem ser usadas em conjunto, com a saída (retorno) de uma determinada função sendo o parâmetro de entrada de outra função.

```
SELECT FirstName, MiddleName, LastName,
       REPLACE(UPPER(CONCAT(FirstName, ' ', MiddleName, ' ', LastName)), ' ', '_') AS NomeMaiúsculoCom_
FROM Person.Person
GO
```

FirstName	MiddleName	LastName	NomeMaiúsculoCom_
Syed	E	Abbas	SYED_E_ABBAS
Catherine	R.	Abel	CATHERINE_R._ABEL
Kim	NULL	Abercrombie	KIM_ABERCROMBIE
Kim	NULL	Abercrombie	KIM_ABERCROMBIE
Kim	B	Abercrombie	KIM_B_ABERCROMBIE
Hazem	E	Abolrous	HAZEM_E_ABOLROUS
Sam	NULL	Abolrous	SAM_ABOLROUS
Humberto	NULL	Acevedo	HUMBERTO_ACEVEDO
Gustavo	NULL	Achong	GUSTAVO_ACHONG
Pilar	NULL	Ackerman	PILAR_ACKERMAN

Tipo de Dados de Data e Hora



- A grande maioria dos SGBDs não oferece meios para que sejam inseridas datas e horas como **valores literais**
- Usam sequências de caracteres (**strings literais**) delimitadas com aspas simples
- Seguem o formato de data e hora especificado nas configurações regionais do sistema operacional
 - Formato Brasileiro: '**DD-MM-AAAA hh:mm:ss[.nnnnnnnn]**'
 - Formato Americano: '**YYYY-MM-DD hh:mm:ss[.nnnnnnnn]**'
- As strings literais são convertidas em data e hora

Funções Básicas de Data e Hora



- Retornam parte da informação de um dado do tipo data e/ou hora
- **DAY (data)**: retorna um inteiro representando o dia da data.
- **MONTH (data)**: retorna um inteiro representando o mês da data.
- **YEAR (data)**: retorna um inteiro representando o ano da data.

```
SELECT DISTINCT DueDate AS Data_e_Hora_Vencimento,  
    DAY(DueDate) AS Dia_Vencimento,  
    MONTH(DueDate) AS Mês_Vencimento,  
    YEAR(DueDate) AS Ano_Vencimento  
FROM Purchasing.PurchaseOrderDetail  
ORDER BY DueDate DESC;
```

Data_e_Hora_Vencimento	Dia_Vencimento	Mês_Vencimento	Ano_Vencimento
2014-10-22 00:00:00.000	22	10	2014
2014-09-22 00:00:00.000	22	9	2014
2014-08-17 00:00:00.000	17	8	2014
2014-08-16 00:00:00.000	16	8	2014
2014-08-15 00:00:00.000	15	8	2014
2014-08-14 00:00:00.000	14	8	2014
2014-08-13 00:00:00.000	13	8	2014

Funções Básicas de Data e Hora



- Função **DATENAME**

- Retorna o dia, mês e ano de uma data-hora
- Retorna outras partes / informações, como o dia da semana, o nome por extenso do mês, a hora, minutos, segundos etc

- **Sintaxe:** *DATENAME (datepart, date)*

- **Date:** um campo ou uma cadeia de caracteres no formato do tipo data-hora.
- **datepart:** opções de formato da data/hora

```
SELECT DATENAME (weekday,GETDATE())
```

datepart	Informação Extraída	Exemplo do Retorno
year, yyyy, yy	Ano	2019
month, mm, m	Mês	October
dayofyear, dy, y	Dia do Ano	303
day, dd, d	Dia do Mês	30
week, wk, ww	Número da Semana	44
weekday, dw	Dia da Semana	Tuesday
hour, hh	Hora	12
minute, n	Minutos	15
second, ss, s	Segundos	32
millisecond, ms	Milisegundos	123
microsecond, mcs	Microsegundos	123456
nanosecond, ns	Nanosegundos	123456700

Funções Aritméticas para Data e Hora



- **DATEDIFF:** calcula a diferença entre duas datas.
 - Utiliza as opções de *datepart*, para indicar a unidade de medida da diferença a ser retornada.
 - **Sintaxe:** *DATEDIFF (datepart , Data_Inicial , Data_Final)*
- **DATEADD:** fazer uma adição (de dias, horas, minutos etc.)
 - Também utiliza as opções de *datepart*, para indicar a unidade de medida do valor que está sendo somado à data .
 - **Sintaxe:** *DATEADD (datepart , Quantidade, Data)*

Próxima Aula



- Demonstração: Funções de Caracteres, de Data e Hora

A Linguagem SQL

CAPÍTULO 4. AULA 4.5.1. DEMONSTRAÇÃO: FUNÇÕES DE CARACTERES, DE DATA E HORA

PROF. GUSTAVO AGUILAR

Demonstração: Funções de Caracteres, de Data e Hora



Play icon

Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU.gusta (56))'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Tipos de Dados e Conversões Explícitas x Implícitas

A Linguagem SQL

CAPÍTULO 4. AULA 4.6. TIPOS DE DADOS E CONVERSÕES EXPLÍCITAS X IMPLÍCITAS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Conversão Implícita de Dados
- Conversão Explícita de Dados

Conversão Implícita de Dados

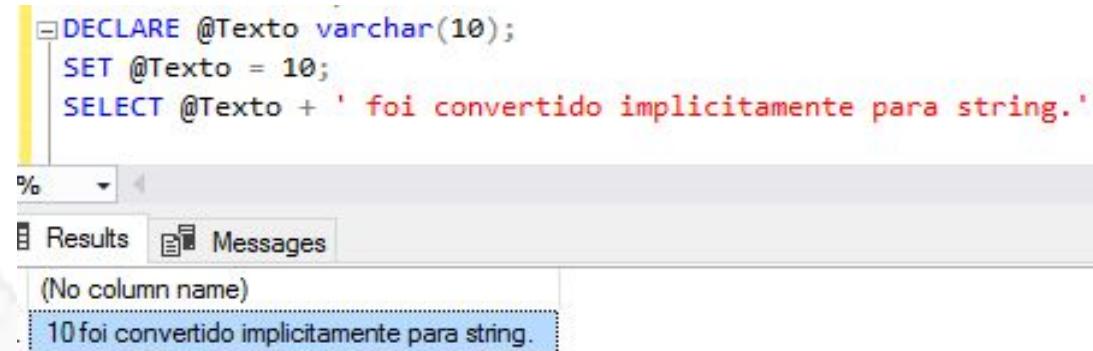


- Além da **conversão implícita de strings literais, no formato de data/hora para o tipo de dados *datetime***, os SGBDs realizam outras conversões implícitas de forma totalmente **transparente** para o usuário.
- Em comparações entre dois tipos de dados diferentes
 - Ex.: quando um *smallint* é comparado a um *int*, e o *smallint* é implicitamente convertido para *int* antes que a comparação prossiga.

Conversão Implícita de Dados

IGTI

- Na atribuição de valores à uma variável ou à uma coluna
 - Tipo de dados resultante é o definido na declaração da variável ou o da coluna.
 - Exemplo:



The screenshot shows a SQL query window in SSMS. The query is:

```
DECLARE @Texto varchar(10);
SET @Texto = 10;
SELECT @Texto + ' foi convertido implicitamente para string.'
```

The results pane shows a single row with the value:

(No column name)
10 foi convertido implicitamente para string.

- O valor inteiro 10 é convertido implicitamente em um varchar, retornando, na instrução SELECT, uma string de caracteres.

Conversão Implícita de Dados



▪ Problema?

- Transparente para quem executa a query
- Gera overhead para a engine do banco de dados
- Causa problemas de performance quando o volume de dados é alto

▪ Onde mais acontece?

- JOINS (relacionamentos) □ tipo de dados da PK e da FK diferentes

Conversão Explícita de Dados



- Além das conversões implícitas, é possível fazer também **conversões explícitas**
- Na T-SQL, existem duas funções para conversão explícita
 - **CAST()**: padrão ISO
 - **CONVERT()**: proprietária do SQL Server

```
--Convertendo Explicitamente Tipo de Dados MONEY para VARCHAR  
SELECT CAST ( $1090.50 AS VARCHAR(10) ) AS VALOR
```

VALOR
1090.50

Conversão Explícita de Dados

IGTI

- A grande maioria dos SGBDs fornece um mapa de conversões
 - Com as conversões implícitas e explícitas permitidas entre os tipos de dados existentes no produto.
 - Conversões não possíveis.
 - Mapa de conversões de tipos de dados do SQL Server □

From	To	binary	varbinary	char	nchar	nvarchar	nvarchar	smalldatetime	datetime	time	datetimeoffset	datetime2	decimal	numeric	float	real	bigin	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	text	sql_variant	xml	CLR UDT	hierarchyid
binary	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
varbinary	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
char	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
varchar	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
nchar	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
nvarchar	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
datetime	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smalldatetime	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
date	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
time	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
datetimeoffset	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
datetime2	binary	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
decimal	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
numeric	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
float	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
real	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
bigin	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
int(INT4)	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smallint(INT2)	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
tinyint(INT1)	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
money	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smallmoney	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
bit	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
timestamp	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
uniqueidentifier	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
image	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
ntext	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
text	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
sql_variant	binary	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
xml	binary	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
CLR UDT	binary	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
hierarchyid	binary	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	

■ Explicit conversion
● Implicit conversion
✖ Conversion not allowed
◆ Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion.
○ Implicit conversions between xml data types are supported only if the source or target is untyped xml. Otherwise, the conversion must be explicit.

Próxima Aula



- Capítulo 5 - Agrupamento de Dados e Funções de Agregação

A Linguagem SQL

Capítulo 5 - Agrupamento de Dados e Funções de Agregação

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 5. AULA 5.1. FUNÇÕES DE AGREGAÇÃO

PROF. GUSTAVO AGUILAR

Nesta Aula

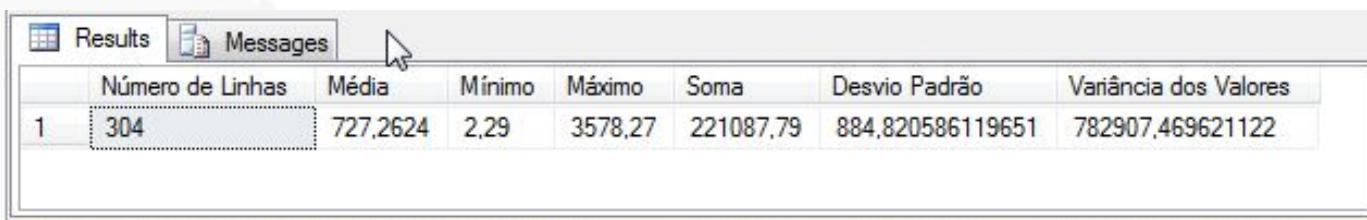


- ❑ Funções de Agregação
- ❑ MIN() e MAX()
- ❑ COUNT()
- ❑ SUM() e AVG()
- ❑ Funções de Agregação x Valores Nulos

Funções de Agregação

IGTI

- Para executar um **cálculo** (contagem, soma, média etc.) **em um conjunto de linhas**, ao invés de fazê-lo linha a linha.
 - Sumarização / totalização por grupos de dados.
- **Retornam** um valor único (**escalar**) e podem ser usadas nas instruções SELECT.



	Número de Linhas	Média	Mínimo	Máximo	Soma	Desvio Padrão	Variância dos Valores
1	304	727,2624	2,29	3578,27	221087,79	884,820586119651	782907,469621122

Funções de Agregação



- **Ignoram valores nulos (NULL)**
 - Com exceção da função COUNT(*).
- **Não geram alias de coluna**
 - Pode-se defini-lo explicitamente.
- **Operam em todas as linhas** retornadas pela instrução SELECT
 - Respeitando os filtros definidos na cláusula WHERE.

MIN() e MAX()

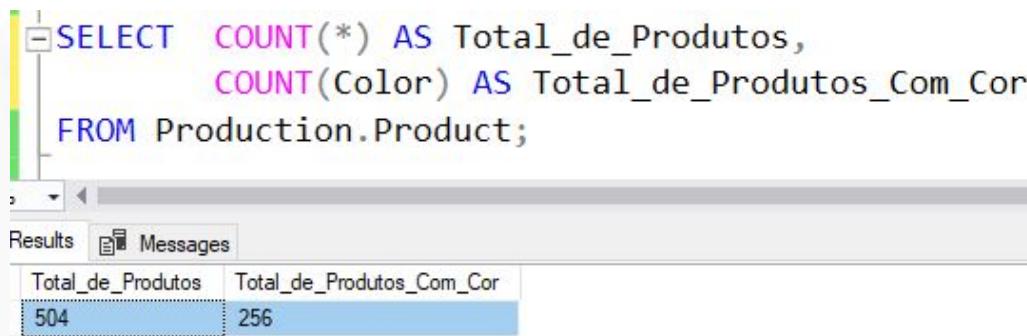
- **MIN (expressão)**: retorna o **menor** número, data / hora mais **antiga** ou **primeira** ocorrência de uma string.
- **MAX (expressão)**: retorna o **maior** número, data / hora mais **recente** ou **última** ocorrência de uma string.

```
SELECT MIN(UnitPrice) AS Preço_Mínimo, MAX(UnitPrice) AS Preço_Máximo  
FROM Sales.SalesOrderDetail;
```

Preço_Mínimo	Preço_Máximo
1,3282	3578,27

COUNT()

- **COUNT (*)**: conta todas as linhas, incluindo aquelas com valores nulos (**NULL**).
- **COUNT (expressão)**: retorna a **contagem** de linhas **não nulas**.



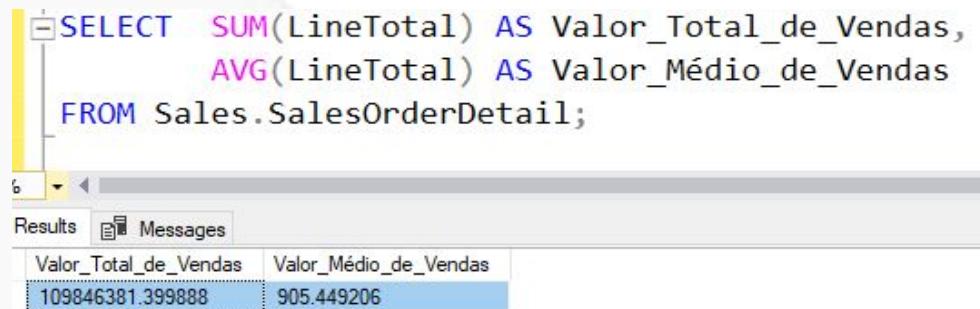
```
SELECT COUNT(*) AS Total_de_Produtos,
       COUNT(Color) AS Total_de_Produtos_Com_Cor
  FROM Production.Product;
```

The screenshot shows a SQL query being run against the Production.Product table. The query uses COUNT(*) to get the total number of products and COUNT(Color) to get the count of products that have a color assigned. The results are displayed in a table with two columns: 'Total_de_Produtos' and 'Total_de_Produtos_Com_Cor'. The value for 'Total_de_Produtos' is 504, and the value for 'Total_de_Produtos_Com_Cor' is 256.

Total_de_Produtos	Total_de_Produtos_Com_Cor
504	256

SUM() e AVG

- **SUM (expressão):** soma todos os valores numéricos **não nulos** de uma coluna.
- **AVG (expressão):** calcula a **média** de todos os **valores não nulos** retornados pela consulta
 - $\text{SUM}() / \text{COUNT}()$.



```
SELECT SUM(LineTotal) AS Valor_Total_de_Vendas,
       AVG(LineTotal) AS Valor_Médio_de_Vendas
  FROM Sales.SalesOrderDetail;
```

The screenshot shows a SQL query being run in SSMS. The query calculates the total and average value of sales from the SalesOrderDetail table. The results are displayed in a table with two columns: 'Valor_Total_de_Vendas' and 'Valor_Médio_de_Vendas'. The total value is 109846381.399888 and the average value is 905.449206.

Valor_Total_de_Vendas	Valor_Médio_de_Vendas
109846381.399888	905.449206

Funções de Agregação x



Valores Nulos

- Com exceção da função COUNT(*), todas as funções de agregação ignoram valores nulos.
 - Por exemplo, ao usar a função SUM, ela somará apenas valores não nulos, uma vez que valores nulos não são avaliados como zero.
- Para tratar valores nulos em tempo de execução ☐ COALESCE()

```
SELECT Weight AS "Sem Peso", COALESCE (Weight,0) AS "Peso Zerado"  
FROM Production.Product;
```

Results	
Sem Peso	Peso Zerado
NULL	0.00

Próxima Aula



- Demonstração: Funções de Agregação

A Linguagem SQL

CAPÍTULO 5. AULA 5.1.1. DEMONSTRAÇÃO: FUNÇÕES DE AGREGAÇÃO

PROF. GUSTAVO AGUILAR

Demonstração: Funções de Agregação



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the Adventureworks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right pane shows a Solution Explorer with various SQL files listed under a folder named 'SQL Query4.sql'.

```
***** Object: Database [Adventureworks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [Adventureworks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'Adventureworks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 64 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Agrupamentos de Dados

A Linguagem SQL

CAPÍTULO 5. AULA 5.2. AGRUPAMENTOS DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Agrupamentos de Dados
- A Cláusula GROUP BY
- Alias no GROUP BY
- Filtrando Agrupamento de Dados

Agrupamentos de Dados

- São subconjuntos de dados.

Marca	Estado	Vendas
A	SP	100
B	SP	200
A	SP	150
B	RJ	140
B	RS	120
C	RS	150
A	RS	200
B	RS	250
C	RJ	230
B	AM	210
A	GO	190
B	GO	180



Marca	Vendas
A	640
B	1100
C	380

- Usado com as funções de agregação para realizar cálculos numéricos por agrupamento.

A Cláusula GROUP BY



- Cláusula da Linguagem SQL responsável por fazer agrupamento de dados.

SELECT < lista de colunas / expressões >

FROM < tabela >

WHERE < condição de filtro – opcional >

GROUP BY < lista de colunas / expressões >

ORDER BY < lista de colunas / expressões – opcional >

A Cláusula GROUP BY



- Cria grupos e agrupa as linhas em cada grupo, conforme determinado pelas combinações exclusivas dos elementos na cláusula GROUP BY
 - Uma coluna / múltiplas colunas / expressões.

The screenshot shows a SQL query being run in a query editor and its results in a results grid.

```
SELECT SalesPersonID AS ID_do_Vendedor, COUNT(SalesOrderID) AS Total_de_Vendas
FROM Sales.SalesOrderHeader
GROUP BY SalesPersonID
ORDER BY 2 DESC
```

The results grid displays the following data:

ID_do_Vendedor	Total_de_Vendas
NULL	27659
277	473
275	450
279	429
276	418

A Cláusula GROUP BY

IGTI

- Colunas ou expressões, na cláusula SELECT, que não estiverem em uma função de agregação, precisam estar no GROUP BY
 - Se não estiver no GROUP BY ou não estiver em uma função de agregação, não fará parte do grupo de colunas passado para a cláusula SELECT.

<i>Ordem Lógica das Operações</i>	Cláusula
5	SELECT
1	FROM
2	WHERE
3	GROUP BY
4	HAVING
6	ORDER BY

Alias no GROUP BY

- Não é possível utilizar um *alias* definido na cláusula *SELECT*, na cláusula *GROUP BY*.

<i>Ordem Lógica das Operações</i>	<i>Cláusula</i>
5	SELECT
1	FROM
2	WHERE
3	GROUP BY
4	HAVING
6	ORDER BY

A callout box with a green border and white background points from the 'GROUP BY' row to a green rectangular callout box containing the text 'Alias não definido ainda!'. A purple circle is located on the right side of the slide.

Filtrando Agrupamento de Dados



- Cláusula **WHERE** não tem a capacidade de filtrar os grupos.

<i>Ordem Lógica das Operações</i>	Cláusula
5	SELECT
1	FROM
2	WHERE
3	GROUP BY
4	HAVING
6	ORDER BY

- Filtra as linhas que serão passadas para a etapa seguinte, onde serão agrupadas com a cláusula GROUP BY
 - Pode ou não reduzir a quantidade de grupos.

Filtrando Agrupamento de Dados

IGTI

```
SELECT OrderDate,  
       SUM(TotalDue)  
  FROM Sales.SalesOrderHeader
```

OrderDate	TotalDue
2013-01-01	2184,41
2013-01-01	1289,21
2013-01-01	3490,41
...	...
2013-12-01	4480,41
2013-12-01	3970,41
2014-01-01	5498,55
2014-01-01	6255,642
...	...
2014-06-30	2921,19



WHERE
YEAR(OrderDate)='2013'

OrderDate	TotalDue
2013-01-01	2184,41
2013-01-01	1289,21
2013-01-01	3490,41
...	...
2013-12-01	4480,41
2013-12-01	3970,41
...	...
2013-12-31	2921,19



GROUP BY
OrderDate

OrderDate	SUM(TotalDue)
2013-01-01	21684,41
2013-01-02	12489,21
2013-01-03	34090,41
...	...
2013-12-31	3108771,97

Filtrando Agrupamento de Dados



- **Cláusula HAVING:** possibilita o filtro de agrupamentos de dados.

- Processada após a cláusula GROUP BY

<i>Ordem Lógica das Operações</i>	Cláusula
5	SELECT
1	FROM
2	WHERE
3	GROUP BY
4	HAVING
6	ORDER BY

Alias não definido ainda!

- Só atua nos grupos e colunas formados pela cláusula GROUP BY

- As colunas a serem usadas como filtro na cláusula HAVING, precisam estar definidas na cláusula GROUP BY ou estarem sendo usadas em uma função de agregação.

Filtrando Agrupamento de Dados



- **Cláusula HAVING:** sintaxe semelhante à da cláusula *WHERE*
 - Possível utilizar também todos os operadores de comparação

SELECT < lista de colunas / expressões >

FROM < tabela >

WHERE < condição de filtro – opcional >

GROUP BY < lista de colunas / expressões >

HAVING < condição de filtro de grupos – opcional >

ORDER BY < lista de colunas / expressões – opcional >

Filtrando Agrupamento de Dados

IGTI

```
SELECT OrderDate,  
       SUM(TotalDue)  
FROM Sales.SalesOrderHeader
```

OrderDate	TotalDue
2013-01-01	2184,41
2013-01-01	1289,21
2013-01-01	3490,41
...	...
2013-12-01	4480,41
2013-12-01	3970,41
...	...
2014-01-01	5498,55
2014-01-01	6255,642
...	...
2014-06-30	2921,19

WHERE
YEAR(OrderDate)=2013'

OrderDate	TotalDue
2013-01-01	2184,41
2013-01-01	1289,21
2013-01-01	3490,41
...	...
2013-12-01	4480,41
2013-12-01	3970,41
...	...
2013-12-31	2921,19

OrderDate	SUM(TotalDue)
2013-01-01	21684,41
2013-01-02	12489,21
2013-01-03	34090,41
...	...
2013-12-31	3108771,97

OrderDate	SUM(TotalDue)
2013-06-30	4800610,79
2013-07-31	4610250,36
2013-10-30	4024573,83

HAVING
SUM(TotalDue)>4000000

Próxima Aula



- Demonstração: Agrupando Dados

A Linguagem SQL

CAPÍTULO 5. AULA 5.2.1. DEMONSTRAÇÃO: AGRUPANDO DADOS

PROF. GUSTAVO AGUILAR

Demonstração: Agrupando Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right-hand pane shows a Solution Explorer with various SQL files listed under a folder named 'SQL Query4.sql'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 6 - Relacionamento de Tabelas (JOIN)

A Linguagem SQL

Capítulo 6 - Relacionamento de Tabelas (JOIN)

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 6. AULA 6.1. INTRODUÇÃO À JUNÇÃO DE TABELAS

PROF. GUSTAVO AGUILAR

Nesta Aula



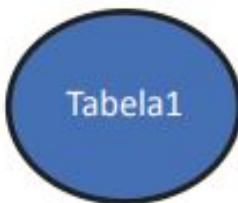
- Introdução à Join de Tabelas
- Join no Padrão ANSI SQL-89
- Produto Cartesiano

Introdução à Join de Tabelas

IGTI

- Query sem join □ query em apenas uma tabela

Conjuntos



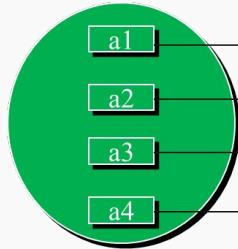
Consultas SQL equivalentes

```
SELECT * FROM Tabela1;  
SELECT * FROM Tabela2;
```

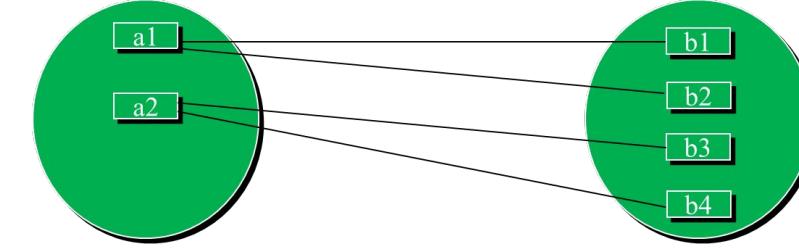
Introdução à Join de Tabelas

IGTI

- Juncão de tabelas está intrinsecamente inserida na teoria de banco de dados relacional
 - Mecanismo para juntar informações
 - Uma tabela pode estar relacionada à uma ou mais tabelas.



Relacionamento 1x1



Relacionamento 1xN

Introdução à Join de Tabelas

IGTI

- Joins são feitos usando os recursos de chave primária / secundária e chave estrangeira (*foreign key*) de 2 ou mais tabelas
 - Coluna de uma tabela é comparada com a coluna de outra tabela, construindo-se o elo desejado entre as mesmas
 - Relacionamento pode ser físico (com constraints) ou apenas lógico



Join no Padrão ANSI SQL-89



- Relacionamento entre duas tabelas: usando **operador de comparação “=”**

```
SELECT Production.ProductCategory.Name AS Nome_da_Categoria,
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria
  FROM Production.ProductCategory, Production.ProductSubCategory
 WHERE Production.ProductCategory.ProductCategoryID = Production.ProductSubCategory.ProductCategoryID
 ORDER BY Nome_da_Categoria ASC, Nome_da_SubCategoria ASC;
```

Results

Nome_da_Categoria	Nome_da_SubCategoria
Accessories	Bike Racks
Accessories	Bike Stands
Accessories	Bottles and Cages
Accessories	Cleaners
Accessories	Fenders
Accessories	Helmets
Accessories	Hydration Packs
Accessories	Lights
Accessories	Locks
Accessories	Panniers
Accessories	Pumps
Accessories	Tires and Tubes
Bikes	Mountain Bikes
Bikes	Road Bikes
Bikes	Touring Bikes
Clothing	Bib-Shorts

Join no Padrão ANSI SQL-89



- Relacionamento entre mais de duas tabelas ☐ operador “AND”

```
SELECT Production.Product.Name AS Nome_do_Produto,
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria,
       Production.ProductCategory.Name AS Nome_da_Categoria
  FROM Production.Product, Production.ProductCategory, Production.ProductSubCategory
 WHERE Production.Product.ProductSubCategoryID = Production.ProductSubCategory.ProductSubcategoryID
   AND Production.ProductCategory.ProductCategoryID = Production.ProductSubCategory.ProductCategoryID
 ORDER BY Nome_do_Produto ASC;
```

% ▶

Nome do Produto	Nome da SubCategoria	Nome da Categoria
All-Purpose Bike Stand	Bike Stands	Accessories
AWC Logo Cap	Caps	Clothing
Bike Wash - Dissolver	Cleaners	Accessories
Cable Lock	Locks	Accessories
Chain	Chains	Components
Classic Vest, L	Vests	Clothing
Classic Vest, M	Vests	Clothing
Classic Vest, S	Vests	Clothing
Fender Set - Mountain	Fenders	Accessories
Front Brakes	Brakes	Components
Front Derailleur	Derailleurs	Components

Join no Padrão ANSI SQL-89



- Possível usar alias de tabelas no join

```
SELECT P.Name AS Nome_do_Produto,  
       S.Name AS Nome_da_SubCategoria,  
       C.Name AS Nome_da_Categoria  
  FROM Production.Product P, Production.ProductCategory C,  
       Production.ProductSubCategory S  
 WHERE P.ProductSubCategoryID = S.ProductSubcategoryID  
   AND C.ProductCategoryID = S.ProductCategoryID  
 ORDER BY Nome_do_Produto ASC;
```

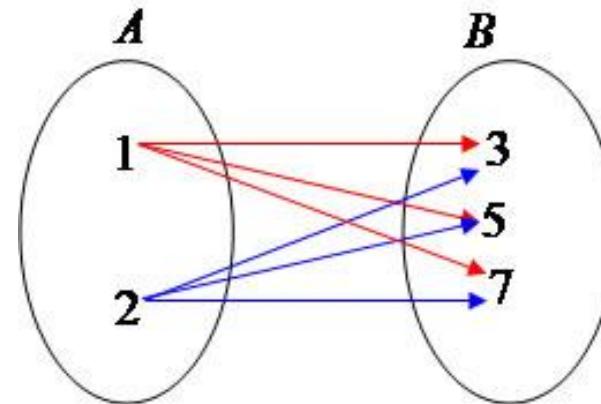
Produto Cartesiano no Padrão

ANSI SQL-89



- Conceito da Teoria de Conjuntos, consistindo no produto entre dois conjuntos.

- Se há um conjunto com 2 itens e outro conjunto com 3 itens, o produto cartesiano entre eles conterá 6 itens (2×3).



- Deve ser evitado oneroso para o servidor, em termos de consumo de recursos (CPU, memória RAM e I/O de disco).

Produto Cartesiano no Padrão

ANSI SQL-89



- Ocorre quando tabelas são relacionadas em uma query, sem se considerar qualquer relação lógica entre elas
 - Sem a expressão do join

```
SELECT Production.ProductCategory.Name AS Nome_da_Categoria,
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria
  FROM Production.ProductCategory, Production.ProductSubCategory
 ORDER BY Nome_da_Categoria ASC, Nome_da_SubCategoria ASC;
```

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The query window contains the provided SQL code. Below it, the 'Results' tab is selected, showing a grid of data. The columns are 'Nome_da_Categoria' and 'Nome_da_SubCategoria'. The data is as follows:

Nome_da_Categoria	Nome_da_SubCategoria
Accessories	Bib-Shorts
Accessories	Bike Racks
Accessories	Bike Stands
Accessories	Bottles and Cages
Accessories	Bottom Brackets

Próxima Aula



- ❑ INNER JOIN, CROSS JOIN e OUTER JOIN

A Linguagem SQL

CAPÍTULO 6. AULA 6.2. INNER JOIN, CROSS JOIN e OUTER JOIN

PROF. GUSTAVO AGUILAR

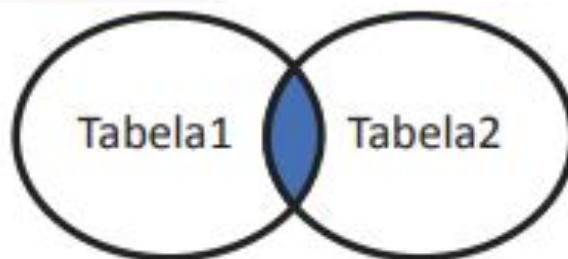
Nesta Aula



- INNER JOIN
- CROSS JOIN
- OUTER JOIN
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN

INNER JOIN

- Para as linhas serem retornadas, os valores das colunas das tabelas relacionadas devem ser iguais (ter correspondência).
- Na Teoria de Conjuntos, isso representa a operação de **interseção**.
- ANSI SQL-92 □ cláusula **INNER JOIN**
- Ordem das tabelas não altera o resultado



```
SELECT * FROM Tabela1 t1;  
INNER JOIN Tabela2 t2  
ON t1.id = t2.fk;
```

INNER JOIN



```
SELECT Production.ProductCategory.Name AS Nome_da_Categoria,  
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria  
  FROM Production.ProductCategory  
 INNER JOIN Production.ProductSubCategory  
    ON Production.ProductCategory.ProductCategoryID = Production.ProductSubCategory.ProductCategoryID  
 ORDER BY Nome_da_Categoria ASC, Nome_da_SubCategoria ASC;
```

```
SELECT Production.Product.Name AS Nome_do_Produto,  
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria,  
       Production.ProductCategory.Name AS Nome_da_Categoria  
  FROM Production.Product  
 INNER JOIN Production.ProductSubCategory  
    ON Production.Product.ProductSubCategoryID = Production.ProductSubCategory.ProductSubcategoryID  
 INNER JOIN Production.ProductCategory  
    ON Production.ProductCategory.ProductCategoryID = Production.ProductSubCategory.ProductCategoryID  
 ORDER BY Nome_do_Produto ASC;
```

Chave primária composta:
ON TAB1.ID1 = TAB2.ID1 AND TAB1.ID2 = TAB2.ID2

CROSS JOIN



- Cláusula para produto cartesiano no padrão ANSI SQL-92

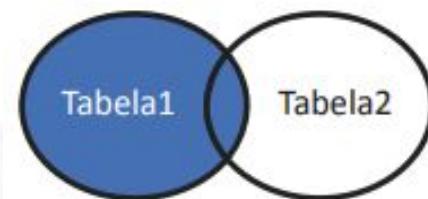
```
SELECT Production.ProductCategory.Name AS Nome_da_Categoria,  
       Production.ProductSubCategory.Name AS Nome_da_SubCategoria  
  FROM Production.ProductCategory  
CROSS JOIN Production.ProductSubCategory  
 ORDER BY Nome_da_Categoria ASC, Nome_da_SubCategoria ASC;
```

LEFT OUTER JOIN

- Retorna as linhas que atendem ao critério do join em ambas as tabelas (assim como acontece no INNER JOIN)

+

- Linhas da tabela à esquerda para as quais não foram encontradas linhas correspondentes na tabela à direita



LEFT OUTER JOIN

- Cláusula OUTER pode ser suprimida no SQL Server

```
SELECT T1.COL1, T2.COL2  
FROM TABELA1 T1  
LEFT JOIN TABELA2 T2  
ON T1.id = T2.id_fk;
```

- **Ordem** em que as tabelas são colocadas **impacta diretamente no resultado** retornado pela query.
- Para as linhas que não tiverem correspondência na outra tabela, a coluna retornada é representada com o valor *NULL*.

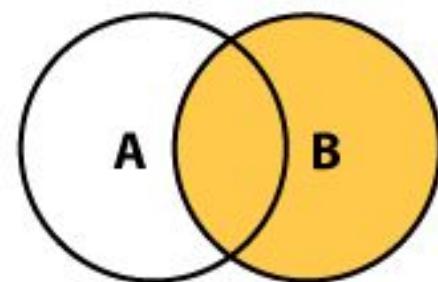
RIGHT OUTER JOIN

IGTI

- Retorna as linhas que atendem ao critério do join em ambas as tabelas (assim como acontece no INNER JOIN)

+

- Linhas da tabela à direita para as quais não foram encontradas linhas correspondentes na tabela à esquerda



RIGHT OUTER JOIN



- Cláusula OUTER pode ser suprimida no SQL Server

```
SELECT T1.COL1, T2.COL2  
FROM TABELA1 T1  
RIGHT JOIN TABELA2 T2  
ON T1.id = T2.id_fk;
```

RIGHT JOIN tem o efeito
inverso do LEFT JOIN

- **Ordem** em que as tabelas são colocadas **impacta diretamente no resultado** retornado pela query.
- Para as linhas que não tiverem correspondência na outra tabela, a coluna retornada é representada com o valor *NULL*.

FULL OUTER JOIN

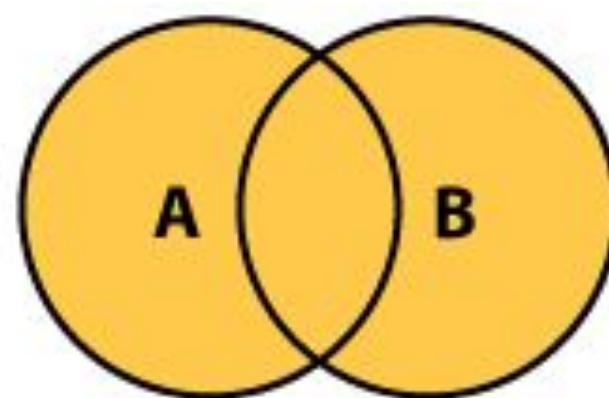
- Retorna as linhas que atendem ao critério do join em ambas as tabelas (assim como acontece no INNER JOIN)

+

- Linhas da tabela à esquerda para as quais não foram encontradas linhas correspondentes na tabela à direita

+

- Linhas da tabela à direita para as quais não foram encontradas linhas correspondentes na tabela à esquerda



FULL OUTER JOIN

- Cláusula OUTER pode ser suprimida no SQL Server

```
SELECT *  
FROM TABELA1 T1  
FULL JOIN TABELA2 T2  
ON T1.id = T2.id_fk;
```

- **Ordem** em que as tabelas são colocadas **não impacta** no resultado retornado pela query.
- Para as linhas que não tiverem correspondência na outra tabela, a coluna retornada é representada com o valor *NULL*.

Próxima Aula



- Demonstração: JOIN de Tabelas

A Linguagem SQL

CAPÍTULO 6. AULA 6.3. DEMONSTRAÇÃO: JOIN DE TABELAS

PROF. GUSTAVO AGUILAR

Demonstração: JOIN de Tabelas



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting database settings like ANSI_PADDING and ANSI_NULL_DEFAULT. The Object Explorer on the left shows the connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU-gusta (56))'. The Solution Explorer on the right lists various SQL scripts related to database creation and management.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\MDF\Adventureworks2017.mdf' )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017_log.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 7 - Subconsultas

A Linguagem SQL

Capítulo 7 - Subconsultas

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 7. AULA 7.1. SUBCONSULTA ESCALAR E MULTIVALORADA

PROF. GUSTAVO AGUILAR

Nesta Aula



- Introdução à Subconsultas
- Subconsulta Escalar
- Subconsulta Multivvalorada

Introdução à Subconsultas



- Uma **subconsulta** é uma **instrução SELECT** aninhada em outra **consulta** (em outra instrução SELECT).
- Consulta aninhada (subconsulta) ☐ **consulta interna** (*inner query*).
- Consulta com a subconsulta ☐ **consulta externa** (*outer query*).
- A forma dos resultados retornados pela subconsulta ou a forma como ela foi construída, determinará o **tipo da subconsulta**:
 - Escalar / Multivalorada
 - Correlacionada
 - Operação de Conjunto

Subconsulta Escalar

- Instrução SELECT aninhada internamente à uma consulta externa, escrita de forma a **retornar um único valor**.
- Pode ser usada em qualquer lugar de uma instrução SQL externa em que uma expressão de valor único é permitida:
 - Instrução SELECT
 - WHERE / HAVING
 - Cláusula FROM

Subconsulta Escalar

- Regras gerais para escrever uma subconsulta escalar:
 - A subconsulta deve ser colocada **entre parênteses**.
 - Podem existir **vários níveis** de subconsultas □ limite do SGBD.
 - Se tivermos uma consulta de 3 níveis, temos 1 consulta interna dentro de uma subconsulta externa, que por sua vez está dentro de uma consulta externa.
 - Se a subconsulta retornar um **conjunto vazio**, o resultado da subconsulta será convertido e retornado como um **NULL**.

Subconsulta Escalar

- Dicas para trabalhar com subconsultas
 - Escrever primeiramente a consulta mais interna
 - Aumentar a complexidade da query gradativamente
- **Exemplo:** retornar os itens inclusos na última venda realizada.
 - Inner Query: última venda □

```
SELECT MAX(SalesOrderID) AS Última_Venda  
FROM Sales.SalesOrderHeader;
```
 - Outer Query: itens da venda

```
SELECT SalesOrderID, SalesOrderDetailID, OrderQty, ProductID, LineTotal  
FROM Sales.SalesOrderDetail WHERE.... (Qual venda?)
```

Subconsulta Escalar

```
SELECT SalesOrderID, SalesOrderDetailID, OrderQty, ProductID, LineTotal  
FROM Sales.SalesOrderDetail  
WHERE SalesOrderID = ( SELECT MAX(SalesOrderID) AS Última_Venda  
                      FROM Sales.SalesOrderHeader  
                    )
```

SalesOrderID	SalesOrderDetailID	OrderQty	ProductID	LineTotal
75123	121315	1	878	21.980000
75123	121316	1	879	159.000000
75123	121317	1	712	8.990000

- **Precedendo** a subconsulta escalar, devem ser usados **operadores** de comparação que **exigem a comparação com um valor único**

= != <
<= > >=

Subconsulta Multivalorada



- Instrução SELECT aninhada internamente à uma consulta externa, escrita de forma a **retornar mais de um valor**.
- Também precisa estar entre parênteses.
- Precedida dos operadores **IN** ou **EXISTS**.
 - O operador IN verifica se a coluna ou expressão informada na cláusula WHERE possui valores que correspondam aos valores retornados pela subconsulta multivalorada.

Subconsulta Multivalorada



- **Exemplo:** retornar a identificação e o número da conta dos clientes da Austrália e França.

```
SELECT CustomerID, AccountNumber  
FROM Sales.Customer  
WHERE TerritoryID IN (  
    SELECT TerritoryID  
    FROM Sales.SalesTerritory  
    WHERE Name = 'Australia' OR Name = 'France'  
);
```

Subconsulta Multivalorada



- Pode-se usar o operador lógico **NOT** em conjunto com o operador IN
 - Inverte o resultado esperado da comparação feita com o operador IN, que na prática pode-se traduzir para “não está contido”.
- Com **NOT IN**, a verificação feita é a de que a coluna ou expressão informada na cláusula WHERE possua valores que não correspondam à nenhum dos valores retornados pela subconsulta multivalorada
 - Exige a leitura da tabela inteira.
 - Deve ser evitado, por questões de performance.

Subconsulta Multivalorada



- **Exemplo:** retornar a identificação e o número da conta de todos os clientes, **exceto** os da Austrália e da França.

```
SELECT CustomerID, AccountNumber  
FROM Sales.Customer  
WHERE TerritoryID NOT IN (  
    SELECT TerritoryID  
    FROM Sales.SalesTerritory  
    WHERE Name = 'Australia' OR Name = 'France'  
);
```

Próxima Aula



- Demonstração: Subconsulta Escalar e Multivalorada

A Linguagem SQL

CAPÍTULO 7. AULA 7.1.1. DEMONSTRAÇÃO: SUBCONSULTA ESCALAR E MULTIVALORADA

PROF. GUSTAVO AGUILAR

Demonstração: Subconsulta Escalar e Multivalorada



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU)'. The Solution Explorer on the right lists various SQL scripts: 01-Criar Banco.sql, 02-Criar Tabelas.sql, 03-Criar Síndrome.sql, 04-Alter.sql, 05-Drop.sql, 06-Select.sql, 07-Alias.sql, 08-CASE.sql, 09-Operadores Aritméticos e Concatenate.sql, 10-ORDER BY.sql, 11-Filtrando Dados.sql, 12-Funções de Caracteres.sql, 13-Funções de Data e Hora.sql, 14-Convenções de Dados.sql, 15-Funções Agregadas.sql, 16-GROUP BY.sql, 17-Filtrando Agrupamentos.sql, 18-Introdução a JOINs.sql, 19-INNER,SELF e CROSS JOIN.sql, 20-OUTER JOIN.sql, 21-SubConsulta Escalar.sql, Miscellaneous, and Scripts. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 512 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\LOG\Adventureworks2017.ldf' , SIZE = 10 , MAXSIZE = 50 , FILEGROWTH = 5 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- ❑ Subconsultas Correlacionadas e com Operadores de Conjuntos

A Linguagem SQL

CAPÍTULO 7. AULA 7.2. SUBCONSULTAS CORRELACIONADAS E COM OPERADORES DE CONJUNTOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Subconsulta Correlacionada
- Subconsulta Correlacionada Escalar
- Subconsulta Correlacionada Multivalorada
- Subconsultas com Operadores de Conjuntos

Subconsulta Correlacionada



- Subconsulta que não possui dependência da consulta externa e pode ser executada isoladamente ☐ **subconsulta independente**.
- Subconsultas correlacionadas não podem ser executadas separadamente da consulta externa ☐ **não são independentes**.
 - A consulta externa passa um valor para a consulta interna (subconsulta), para ser usado como um parâmetro na execução.
- Subconsulta correlacionada pode ser escalar ou multivalorada.
- Executam várias vezes, de forma que a *outer query* é executada primeiro e, para cada linha retornada, a *inner query* é processada.

Subconsulta Correlacionada Escalar



- **Exemplo de subconsulta correlacionada escalar:** retornar as vendas mais recentes feitas por cada empregado.

```
SELECT Q1.SalesOrderID, Q1.SalesPersonID, Q1.OrderDate  
FROM Sales.SalesOrderHeader AS Q1  
WHERE Q1.OrderDate = (  
    SELECT MAX(Q2.OrderDate)  
    FROM Sales.SalesOrderHeader AS Q2  
    WHERE Q2.SalesPersonID = Q1.SalesPersonID  
)  
ORDER BY Q1.SalesPersonID, Q1.OrderDate;
```

Subconsulta Correlacionada



Multivalorada

- Uso do operador **EXISTS**
 - Ao contrário do operador IN, interrompe a execução da subconsulta, assim que uma correspondência for encontrada.
 - Ao invés de recuperar um valor escalar ou uma lista com vários valores em uma subconsulta, ele simplesmente verifica se existe qualquer linha, no resultado, que satisfaça a condição de comparação com a consulta mais externa.
 - Bem mais performático que o operador IN.
 - Também pode ser usado em conjunto com o operador lógico **NOT**.

Subconsulta Correlacionada



Multivalorada

- **Exemplo:** retornar a identificação e o número da conta dos clientes da Austrália e França.

```
|SELECT CustomerID, AccountNumber  
FROM Sales.Customer  
WHERE TerritoryID IN (  
    SELECT TerritoryID  
    FROM Sales.SalesTerritory  
    WHERE Name = 'Australia' OR Name = 'France'  
);
```

```
SELECT CustomerID, AccountNumber  
FROM Sales.Customer C  
WHERE EXISTS (  
    SELECT *  
    FROM Sales.SalesTerritory T  
    WHERE T.TerritoryID = C.TerritoryID  
    AND (Name = 'Australia' OR Name = 'France')  
);
```

Subconsultas com Operadores de Conjuntos



- **Operadores de conjuntos** usados para realizar operações entre duas ou mais **subconsultas independentes entre si**.
 - Todas as **subconsultas (*inner query / outer query*)** são **independentes** e podem ser executadas separadamente.
 - Operadores da Linguagem T-SQL: **UNION, INTERSECT e EXCEPT**.
- **Cada conjunto de entrada** é o resultado de uma consulta (subconsulta), que pode incluir qualquer instrução SELECT, mas **não pode possuir uma cláusula ORDER BY**
 - ORDER BY pode ser colocado na consulta geral (final da query).

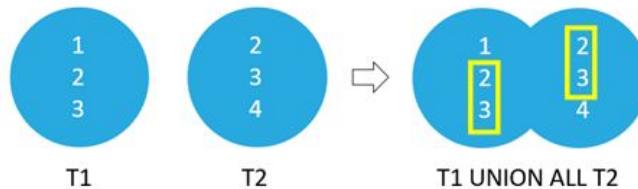
Subconsultas com Operadores de Conjuntos



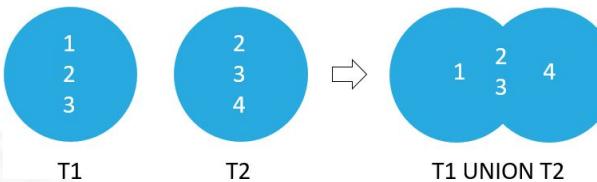
- Os conjuntos de entrada devem ter o **mesmo número de colunas** e as colunas devem ter **tipos de dados compatíveis**
 - Conversões explícitas são executadas quando permitido.
 - Pode-se forçar uma conversão explícita usando CAST ou CONVERT.
- Um valor nulo (NULL) em um conjunto é tratado como igual a outro valor nulo em outro conjunto.

União

- Unir dois ou mais conjuntos de dados.
- Operador **UNION ALL**: não elimina as linhas duplicadas.



- Operador **UNION**: exclui as linhas duplicadas.



União



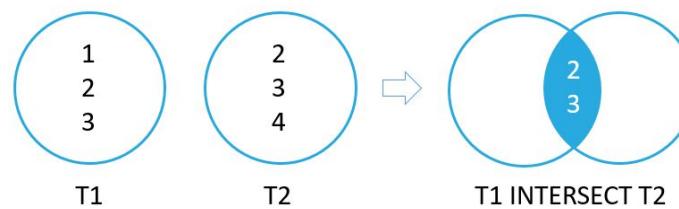
- Sintaxe

```
SELECT nome, cpf, telefone  
FROM Tab_Medicos  
UNION ALL  
SELECT nome, cpf, telefone  
FROM Tab_Pacientes;
```

```
SELECT nome, cpf, telefone  
FROM Tab_Medicos  
UNION  
SELECT nome, cpf, telefone  
FROM Tab_Pacientes;
```

Interseção

- Encontrar os valores comuns a dois ou mais conjuntos.
- Operador ***INTERSECT***.

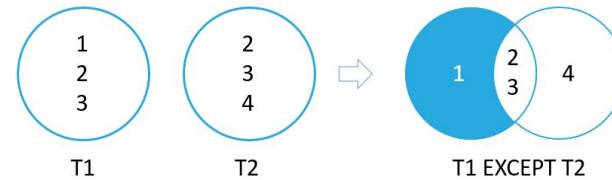


- **Exemplo:** lista com nomes distintos das pessoas que são médicos e pacientes.

```
SELECT nome, cpf, telefone  
FROM Tab_Medicos  
INTERSECT  
SELECT nome, cpf, telefone  
FROM Tab_Pacientes;
```

Exceção

- Operação de diferença entre dois ou mais conjuntos.
- Operador **EXCEPT**.



- **Exemplo:** lista com nomes distintos das pessoas que são médicos e não são pacientes.

```
SELECT nome, cpf, telefone  
FROM Tab_Medicos  
EXCEPT  
SELECT nome, cpf, telefone  
FROM Tab_Pacientes;
```

- Retornar as linhas distintas da subconsulta de entrada acima que não são retornadas pela subconsulta de entrada abaixo.

Próxima Aula



- Demonstração: Subconsulta Correlacionadas e de Conjuntos

A Linguagem SQL

CAPÍTULO 7. AULA 7.2.1. DEMONSTRAÇÃO: SUBCONSULTAS CORRELACIONADAS E DE CONJUNTOS

PROF. GUSTAVO AGUILAR

Demonstração: Subconsultas Correlacionadas e de Conjuntos



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU.gusta (56))'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 512 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\LOG\Adventureworks2017.ldf' , SIZE = 10 , MAXSIZE = 50 , FILEGROWTH = 5 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 8 - Inserção, Atualização e Exclusão de Dados

A Linguagem SQL

Capítulo 8 - Inserção, Atualização e Exclusão de Dados

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 8. AULA 8.1. INSERINDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- INSERT INTO
- SELECT INTO
- Demonstração: Inserindo Dados

INSERT INTO



- Comando da classe Data Manipulation Language (DML) para **persistir dados** em uma tabela previamente criada.
 - Existem outros comandos para inserir dados: mais usado é o INSERT.
- **Sintaxe básica** *INSERT [INTO] <Tabela ou visão> [lista_de_colunas]*
VALUES (Valor | Expressão | NULL | DEFAULT)
 - Lista de colunas e cláusula *INTO* são opcionais
 - Ao não informar a lista de colunas, é preciso inserir os valores para todas as colunas, na ordem em que as colunas estão criadas na tabela.

```
INSERT Person.AddressType (AddressTypeID, Name, rowguid, ModifiedDate)
VALUES (7, 'Comercial', DEFAULT, GETDATE());
```

INSERT INTO

- Possível inserir múltiplas linhas com apenas um INSERT INTO

```
INSERT INTO TAB1 (COL1, COL2)
```

```
VALUES (100, 'Valor não numérico'),
```



```
(200, 'Valor não numérico 2'),
```



```
(300, 'Valor não numérico 3');
```

- Possível utilizar uma consulta SELECT em outra tabela ou conjunto de tabelas para retornar os valores a serem inseridos na tabela em questão.

```
INSERT INTO TAB1 (COL1, COL2)
```

```
SELECT COL4, COL5 FROM TAB2;
```

SELECT INTO



- Outra possibilidade para inserir dados em uma tabela.
- **Sintaxe**
*SELECT coluna1, coluna2, ...
INTO Nova_Tabela FROM Tabela_Origem;*
- **Tabela destino** dos dados **não pode estar criada**
 - A cláusula SELECT INTO criará a tabela em tempo de execução.
 - Estrutura física definida pela lista de colunas na cláusula SELECT.
 - Cada coluna na nova tabela terá o **mesmo nome, tipo de dados e anulabilidade da coluna correspondente (ou expressão)** da lista de colunas na cláusula SELECT
 - **Índices e constraints não são criados na nova tabela.**

Demonstração: Inserindo Dados



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right pane shows a Solution Explorer with various SQL files listed, and the bottom pane shows the status bar indicating a connection to 'ULTRA-GU (14.0 RTM)'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Atualizando Dados

A Linguagem SQL

CAPÍTULO 8. AULA 8.2. ATUALIZANDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- A Cláusula UPDATE
- A Cláusula MERGE
- Demonstração: Atualizando Dados

A Cláusula UPDATE



- Cláusula da Linguagem SQL usada para **alterar os dados de uma ou mais colunas existentes em uma tabela.**
- Pode operar em um conjunto de linhas definido por uma condição em uma cláusula WHERE ou de JOIN.
- É usada a **cláusula SET** para atribuir o novo valor à coluna
 - Permite atualização dos valores de uma ou mais colunas no mesmo comando UPDATE.
 - Sem o filtro (WHERE / JOIN), serão atualizadas todas as colunas existentes na cláusula SET, **para todas as linhas da tabela.**

A Cláusula UPDATE



▪ Sintaxe

```
UPDATE <Nome_Tabela>
SET <Coluna1> = { valor | expressão | DEFAULT | NULL },
    <Coluna2> = { valor | expressão | DEFAULT | NULL },
    <ColunaN> {...n}
WHERE <Condição>;
```

```
UPDATE TAB2
SET COL3 = 500
WHERE COL1 = 10;
```

A Cláusula MERGE



- Usada quando é preciso **atualizar, inserir ou excluir** linhas de uma tabela, no mesmo comando.
- MERGE atua com base em uma ou mais condições de decisão, com as quais será decidida qual operação (INSERT/UPDATE/DELETE) executar:
 - Quando os dados de origem correspondem aos dados no destino, ocorre a atualização dos dados.
 - Quando não há correspondência no destino, ocorre a inserção dos dados.
 - Quando os dados de destino não têm correspondência no conjunto de dados de origem, ocorre a exclusão dos dados na tabela de destino.

A Cláusula MERGE



- **Exemplo:** atualizar os campos Data_Venda e Valor quando a venda já existir na tabela, e inserir a linha quando a venda ainda não existir.

```
MERGE Tab_Venda_Destino AS Destino
USING Tab_Venda_Origem AS Origem ON (Origem.Cod_Venda = Destino.Cod_Venda)
WHEN MATCHED THEN -- Registro existe nas 2 tabelas
    UPDATE SET
        Destino.Data_Venda = Origem.Data_Venda,
        Destino.Valor = Origem.Valor
-- Registro não existe na origem, apenas no destino. Vamos apagar o dado da origem.
WHEN NOT MATCHED BY SOURCE THEN
    DELETE
-- Registro não existe no destino. Vamos inserir.
WHEN NOT MATCHED BY TARGET THEN
    INSERT
        VALUES(Origem.Cod_Venda, Origem.Data_Venda, Origem.Cod_Produto, Origem.Qtde, Origem.Valor);
```

Demonstração: Atualizando Dados



Play icon

Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting its properties (CONTAINMENT = NONE, ON PRIMARY, LOG ON, etc.), enabling full-text search, and adjusting database settings like ANSI_NULL_DEFAULT and ANSI_PADDING. The right-hand pane shows a Solution Explorer with various SQL scripts listed under a folder named 'SQLQuery4.sql'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'Adventureworks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Excluindo Dados

A Linguagem SQL

CAPÍTULO 8. AULA 8.3. EXCLUINDO DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- A Cláusula DELETE
- Demonstração: Excluindo Dados

A Cláusula DELETE



- Cláusula da Linguagem SQL usada para **excluir (“deletar”)** **dados(linhas) de uma tabela.**
- Pode operar em um conjunto de linhas definido por uma condição em uma cláusula WHERE ou de JOIN.
- **Sintaxe:** *DELETE FROM <Nome_Tabela>
WHERE <Condição>;*

```
DELETE FROM TAB2  
WHERE COL1 = 10;
```

A Cláusula DELETE



- Sem o filtro (WHERE / JOIN), são **excluídas todas as linhas** da tabela.

DELETE FROM TAB2;

- Nas situações onde deseja-se excluir todos os dados de uma tabela, por questões de performance é preferível usar o comando **TRUNCATE** da classe **DDL** da Linguagem SQL.

TRUNCATE TABLE TAB2;

Demonstração: Excluindo Dados

IGTI



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the Adventureworks2017 database. The script includes commands for creating the database, setting its properties (CONTAINMENT = NONE, ON PRIMARY, LOG ON, etc.), enabling full-text search, and adjusting ANSI settings. The right pane shows the Solution Explorer with various SQL files listed, such as 01-Criar Banco.sql, 02-Criar Tabelas.sql, and others related to database creation and management.

```
***** Object: Database [Adventureworks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [Adventureworks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'Adventureworks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Demonstração: Notebook no Azure Data Studio

A Linguagem SQL

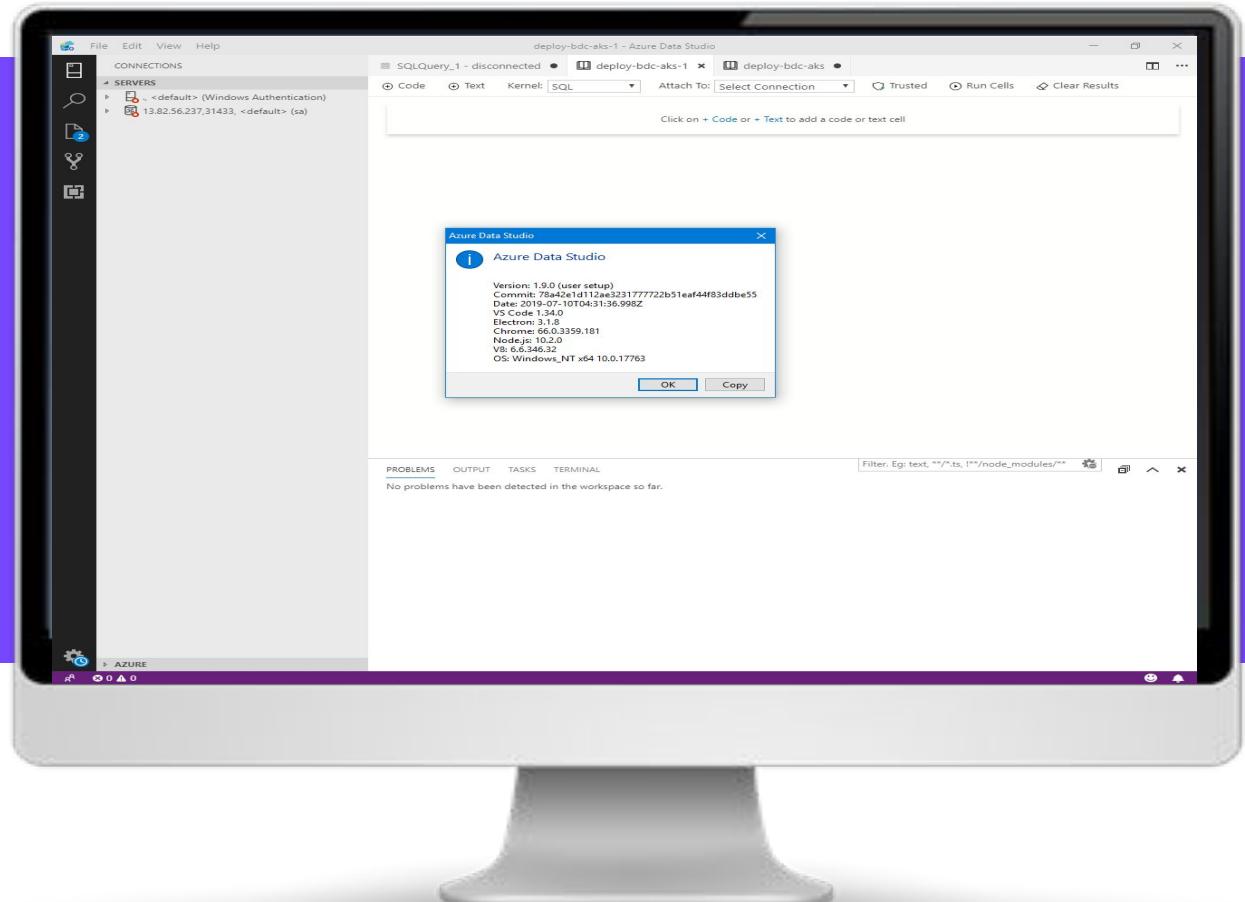
CAPÍTULO 8. AULA 8.4. DEMONSTRAÇÃO: NOTEBOOK NO AZURE DATA STUDIO

PROF. GUSTAVO AGUILAR

Notebook no Azure Data Studio



Demo



Próxima Aula



- Capítulo 9 - Linguagem de Controle de Transação (TCL)

A Linguagem SQL

Capítulo 9 - Linguagem de Controle de Transação (TCL)

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 9. AULA 9.1. CONCEITOS BÁSICOS DE TRANSAÇÃO

PROF. GUSTAVO AGUILAR

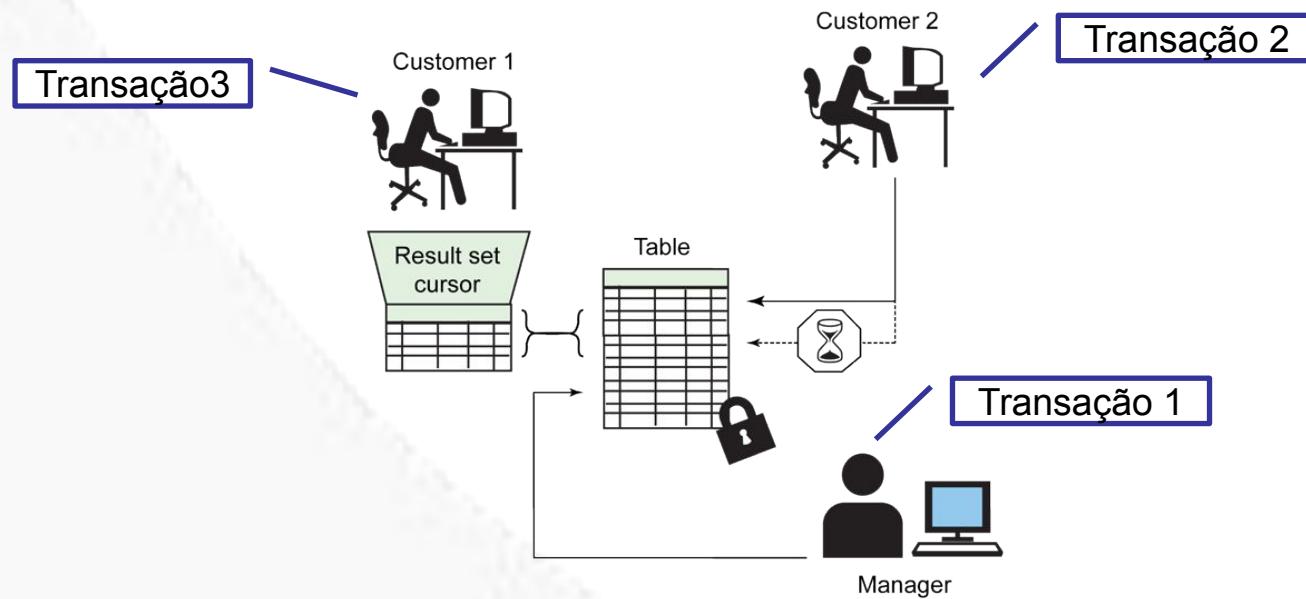
Nesta Aula



- Transação
- Propriedades ACID
- Conceitos Básicos de Transação

Transação

- Uma sequência de instruções SQL executadas no SGBD.



Propriedades ACID

- Propriedades das transações em bancos de dados relacionais.
- Garantem a integridade e consistência dos dados.

A

Atomic

C

Consistent

I

Isolated

D

Durable

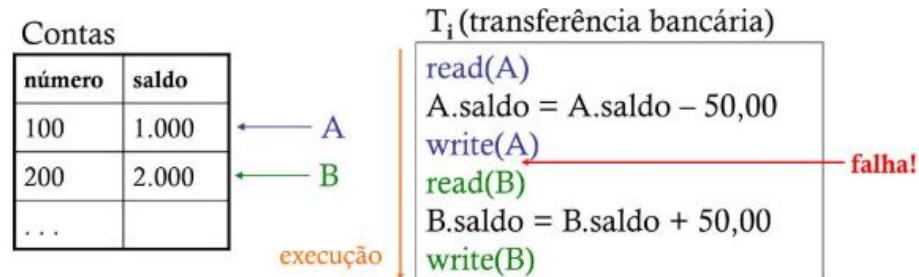


Propriedades ACID

IGTI

▪ ATOMICIDADE

- A transação será executada totalmente ou não será executada
 - Transação atômica.
- Requer que as transações sejam “tudo ou nada” (“all or nothing”)
 - Se uma parte da transação falhar, a transação inteira deve falhar.
 - O estado do banco de dados permanecer inalterado.

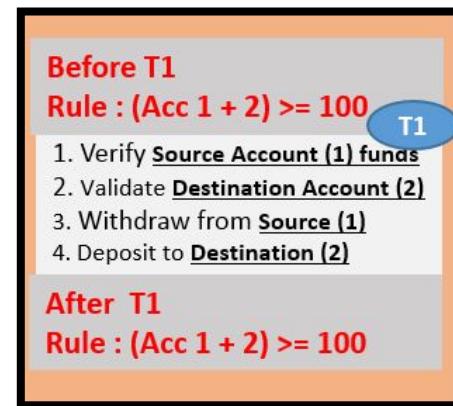


Propriedades ACID

IGTI

▪ CONSISTÊNCIA

- Qualquer transação levará o banco de dados de um estado válido a outro.
- Em caso de sucesso, a transação cria um novo estado válido dos dados ou, em caso de falha, retorna todos os dados ao estado imediatamente anterior ao início da transação.

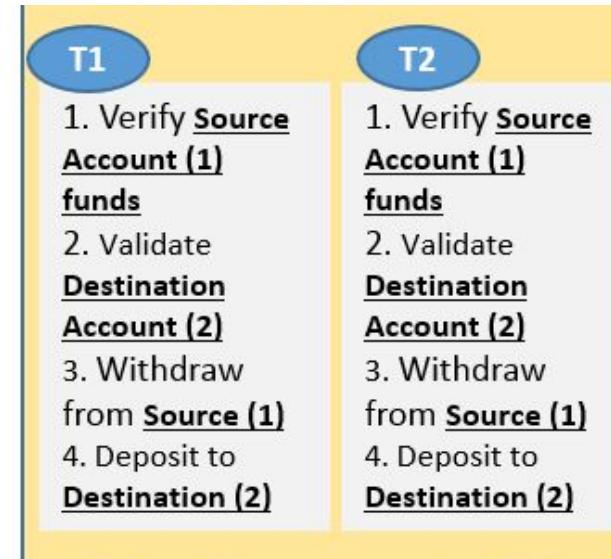


Propriedades ACID

IGTI

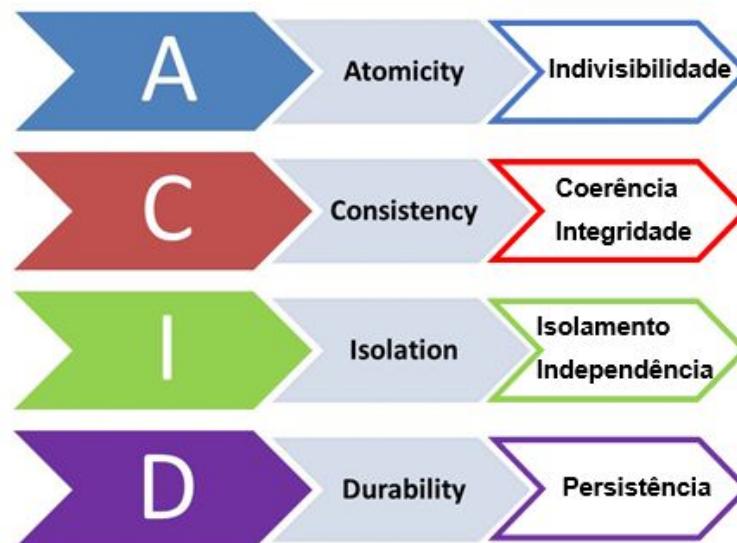
▪ ISOLAMENTO

- Execuções concorrentes (em paralelo) de transações resultem em um estado do banco de dados que seria obtido caso as transações fossem executadas serialmente.
- Uma transação em andamento, mas ainda não validada, permanece isolada de qualquer outra operação, garantindo que a transação não sofra interferência de nenhuma outra transação concorrente.



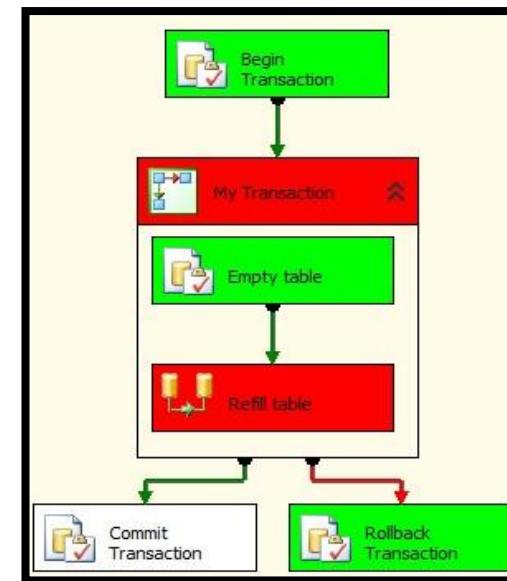
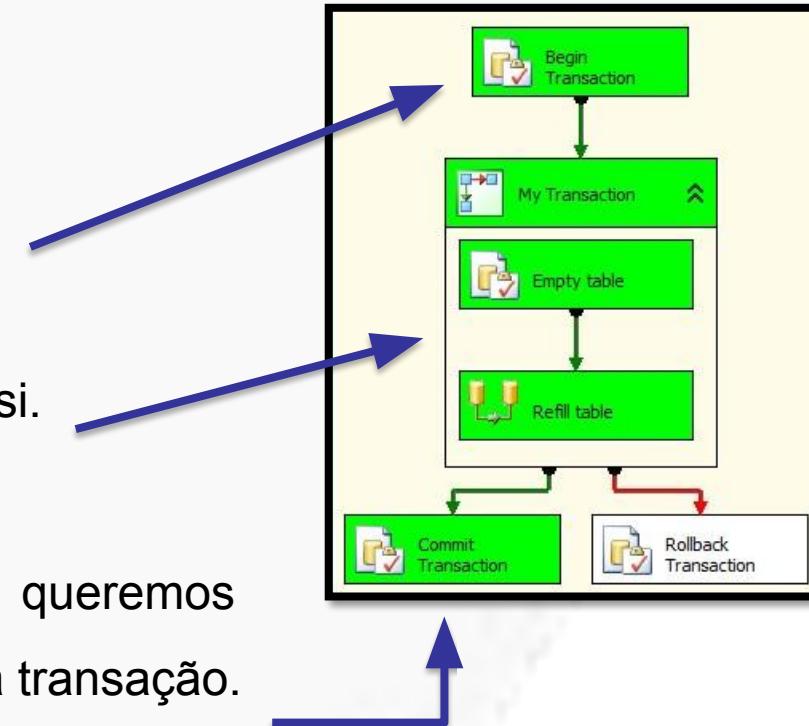
Propriedades ACID

- **DURABILIDADE:** uma vez que uma transação tenha sido confirmada, ela permanecerá assim, mesmo no caso de um reinício do sistema, crash, falta de energia ou erros posteriores.



Transação

- Início da transação
- Meio: transação em si.
- COMMIT: quando queremos confirmar (persistir) a transação.
- ROLLBACK: desfazer a transação (não persistir).



Transação

▪ Forma de Criação da Transação

- **EXPLÍCITA:** iniciadas através de comandos da Linguagem de Controle de Transação (TCL).
- **IMPLÍCITA:** instruções enviadas separadamente para execução são automaticamente inseridas em uma transação pelo SGBD.
 - Também chamadas de transações com autocommit.
 - São confirmadas (“commitadas”) automaticamente quando a instrução é bem-sucedida, ou revertidas automaticamente quando a instrução encontra um erro em tempo de execução.
 - É como se o SGBD abrisse a transação explicitamente e acompanhasse o status dela para saber se a confirma no banco de dados ou se a desfaz.

Próxima Aula



- Controle de Transações

A Linguagem SQL

CAPÍTULO 9. AULA 9.2. CONTROLE DE TRANSAÇÕES

PROF. GUSTAVO AGUILAR

Nesta Aula



- ❑ Iniciando Explicitamente uma Transação
- ❑ Efetivando uma Transação
- ❑ Desfazendo uma Transação
- ❑ Transações Aninhadas e Nomeadas
- ❑ TRY / CATCH

Iniciando Explicitamente uma Transação



- Transações explícitas são controladas (criadas / confirmadas / desfeitas) usando os comandos da classe TCL.
- Cada SGBD pode implementar variações ou novos comandos para controlar as transações.
- Para iniciar uma transação □ comando **BEGIN TRANSACTION**

BEGIN TRANSACTION;

```
DELETE FROM HumanResources.JobCandidate  
WHERE JobCandidateID = 13;
```

Efetivando uma Transação



- Transação aberta de forma explícita, deve ser concluída (persistida) usando o comando **COMMIT**.

BEGIN TRANSACTION;

*DELETE FROM HumanResources.JobCandidate
WHERE JobCandidateID = 13;*

COMMIT;

Desfazendo uma Transação



- Transação aberta de forma explícita, pode ser desfeita (cancelada) usando o comando **ROLLBACK**.

--Iniciando e desfazendo uma transação

```
BEGIN TRANSACTION;
    INSERT INTO ValueTable VALUES(1);
    INSERT INTO ValueTable VALUES(2);
ROLLBACK;
```

Transações Aninhadas

- Uma transação contida dentro de outra transação.
- Podem possuir N níveis, dependendo do SGBD em questão.

BEGIN TRANSACTION;

 UPDATE table1 ...;

BEGIN TRANSACTION;

 UPDATE table2 ...;

 SELECT * from table1;

COMMIT;

 UPDATE table3 ...;

COMMIT;

Transações Nomeadas



- Transações que tiveram um nome ("label") associado explicitamente.
- Nomes podem ser usados para “commitá-las” a qualquer momento.

```
BEGIN TRANSACTION T1;  
    UPDATE table1 ...;  
    BEGIN TRANSACTION T2;  
        UPDATE table2 ...;  
        SELECT * from table1;  
    COMMIT TRANSACTION T2;  
    UPDATE table3 ...;  
COMMIT TRANSACTION T1;
```

TRY / CATCH



- Comandos T-SQL para implementar tratamento de erros na execução das queries;
- Um grupo de instruções Transact-SQL pode ser incluído em um bloco TRY;
- Se ocorrer um erro no bloco TRY, o controle passará para outro grupo de instruções que está incluído em um bloco CATCH

TRY / CATCH



```
BEGIN TRANSACTION;

BEGIN TRY
    -- COMANDO SQL
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_SEVERITY() AS ErrorSeverity,
           ERROR_STATE() AS ErrorState,
           ERROR_PROCEDURE() AS ErrorProcedure,
           ERROR_LINE() AS ErrorLine,
           ERROR_MESSAGE() AS ErrorMessage;

    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
    COMMIT TRANSACTION;
```

TRY / CATCH



- Tipos de erros não afetados por uma construção TRY ... CATCH:
 - Avisos ou mensagens informativas com *severity* igual ou inferior a 10;
 - Erros com *severity* 20 ou superior que interrompem o processamento da tarefa para a sessão. Se ocorrer um erro com gravidade igual ou superior a 20 e a conexão com o banco de dados não for interrompida, TRY ... CATCH tratará do erro;
 - Atenções, como solicitações de interrupção do cliente ou conexões de clientes interrompidas;
 - Quando a sessão é encerrada usando a instrução KILL.
- Tipos de erros não tratados por um bloco CATCH quando ocorrem no mesmo nível de execução que a construção TRY ... CATCH:
 - Erros de compilação, como erros de sintaxe;
 - Erros de resolução de nome de objeto.

Próxima Aula



- Demonstração: Controle de Transação

A Linguagem SQL

CAPÍTULO 9. AULA 9.3. DEMONSTRAÇÃO: CONTROLE DE TRANSAÇÃO

PROF. GUSTAVO AGUILAR

Demonstração: Controle de Transação



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the Adventureworks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right pane shows a Solution Explorer with various SQL files listed, and the bottom pane shows the status bar indicating a connection to 'ULTRA-GU (14.0 RTM)'.

```
***** Object: Database [Adventureworks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [Adventureworks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 512 , FILEGROWTH = 64 )
LOG ON
( NAME = N'Adventureworks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 512 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [Adventureworks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 10 - Linguagem de Controle de Acesso a Dados (DCL)

A Linguagem SQL

Capítulo 10 - Linguagem de Controle de Acesso a Dados (DCL)

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 10. AULA 10.1. SEGURANÇA EM BANCOS DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- ❑ Segurança em Bancos Relacionais
- ❑ Principais Componentes de Segurança

Segurança em Bancos Relacionais



- Valor crescente, seja financeiro ou estratégico, dos dados armazenados
 - geração de **conhecimento, informação, vantagem competitiva.**
- **Lei Geral de Proteção de Dados (LGPD)**
- Aumento da preocupação com a proteção dos dados armazenados.
 - **Proteção externa ao SGBD:** firewall, antivírus, etc.
 - **Proteção interna do SGBD:** autenticação, autorização, criptografia, etc.
- Autorização / permissão de acesso no SGBD □ configurada através da **Linguagem de Controle de Acesso a Dados (Data Control Language),** a **DCL.**

Segurança em Bancos Relacionais

IGTI

- Processo de **Concessão** das Permissões de Acesso.



- Processo de **Revogação** das Permissões de Acesso.

Principais Componentes de Segurança



- Cada SGBD disponibiliza sua gama de recursos e comandos para habilitar, configurar e gerenciar seu framework de segurança.
 - Objetos que podem ser protegidos, os mecanismos e a granularidade de proteção.
- **PERMISSION (Privilégio):** permissão que pode ser concedida para executar alguma ação, ou seja, executar algum comando SQL.
- **ROLE (Papel):** conjunto de privilégios que pode ser concedido a usuários ou a outras roles.
 - Criadas com o comando CREATE ROLE da classe DDL da Linguagem SQL.

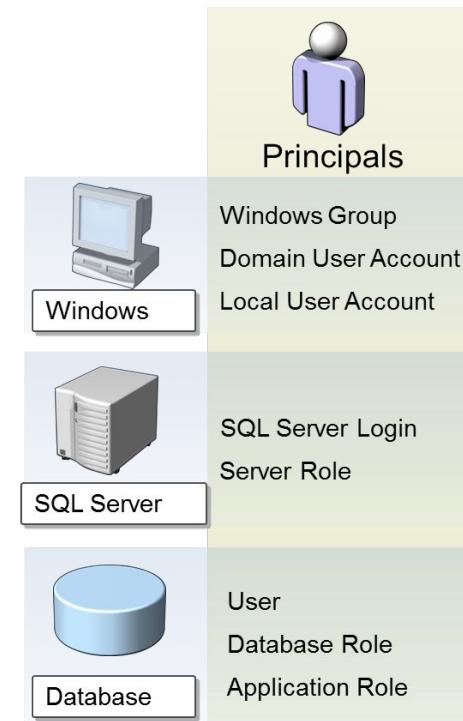
Principais Componentes de Segurança



- **PRINCIPALS:** são entidades com acesso concedido à uma instância do SQL Server, no nível do servidor, da instância ou do banco de dados, para os quais podem ser concedidos privilégios ou roles.

- Criados com o comando CREATE da classe DDL.
- **Exemplo:** criação de um PRINCIPAL do tipo login

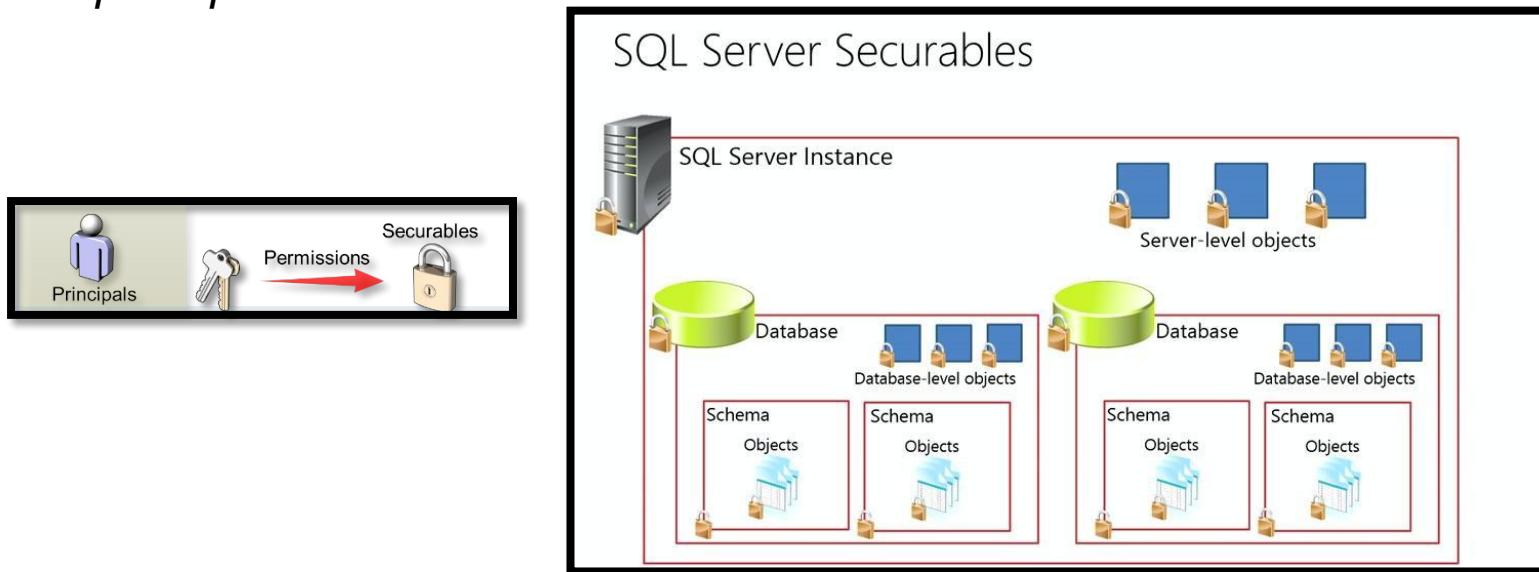
```
CREATE LOGIN UsrIGTI  
WITH PASSWORD = 'Pa55w.rd',  
DEFAULT_DATABASE = [BDIGTI];
```



Principais Componentes de Segurança

IGTI

- **SECURABLES:** objetos do SQL Server, no nível do servidor ou do banco de dados, nos quais são concedidas as permissões para os *principals*.



Próxima Aula



- Concessão e Revogação de Privilégios

A Linguagem SQL

CAPÍTULO 10. AULA 10.2. CONCESSÃO E REVOGAÇÃO DE PRIVILÉGIOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Concessão de Privilégios
- Revogação de Privilégios

Concessão de Privilégios



- Feita através do comando **GRANT** da classe DCL.
- Privilégios para poder executar comandos **DML / DDL / DCL**.
- **Sintaxe DML:** *GRANT [Privilégio(s)] ON [Objeto] TO [Usuário];*
- **Exemplos de Concessão de Privilégio DML**

GRANT SELECT ON [Person].[Address] TO UsrAdventureSystem;

GRANT INSERT,UPDATE ON [Person].[Person] TO Role_App;

GRANT SELECT, INSERT, DELETE, UPDATE ON [Person].[Address] TO UsrApp;

Concessão de Privilégios



- Sintaxe DDL: *GRANT [Privilégio(s)] TO [Usuário];*
- Exemplos de Concessão de Privilégio DDL

GRANT CREATE TABLE TO Role_App;

GRANT CREATE USER TO UsrSystem;

- Opção **WITH GRANT OPTION**: permite que o usuário que recebeu uma permissão, possa concedê-la para outro usuário.

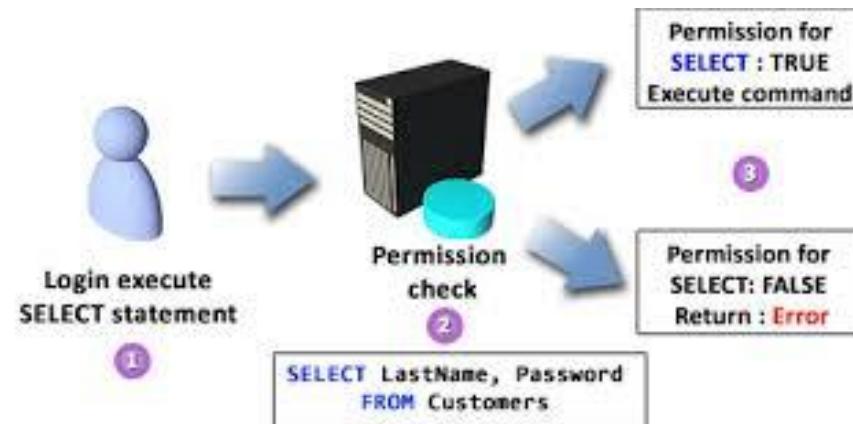
GRANT DELETE ON DatabaseLog

*TO UsrAdventureSystem **WITH GRANT OPTION**;*

Concessão de Privilégios

IGTI

- Permissões de acesso são verificadas em tempo de execução das queries.

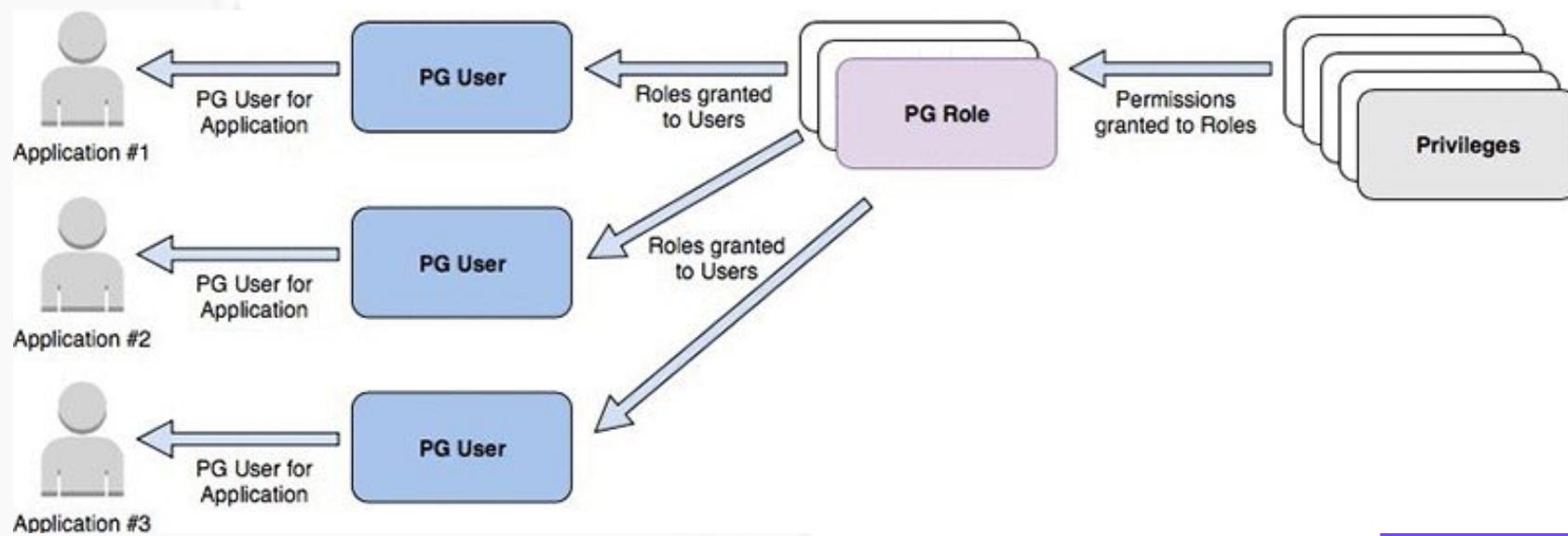


- Caso o usuário não tenha permissão em qualquer objeto contido da query, toda a execução da query falha.

Concessão de Privilégios

IGTI

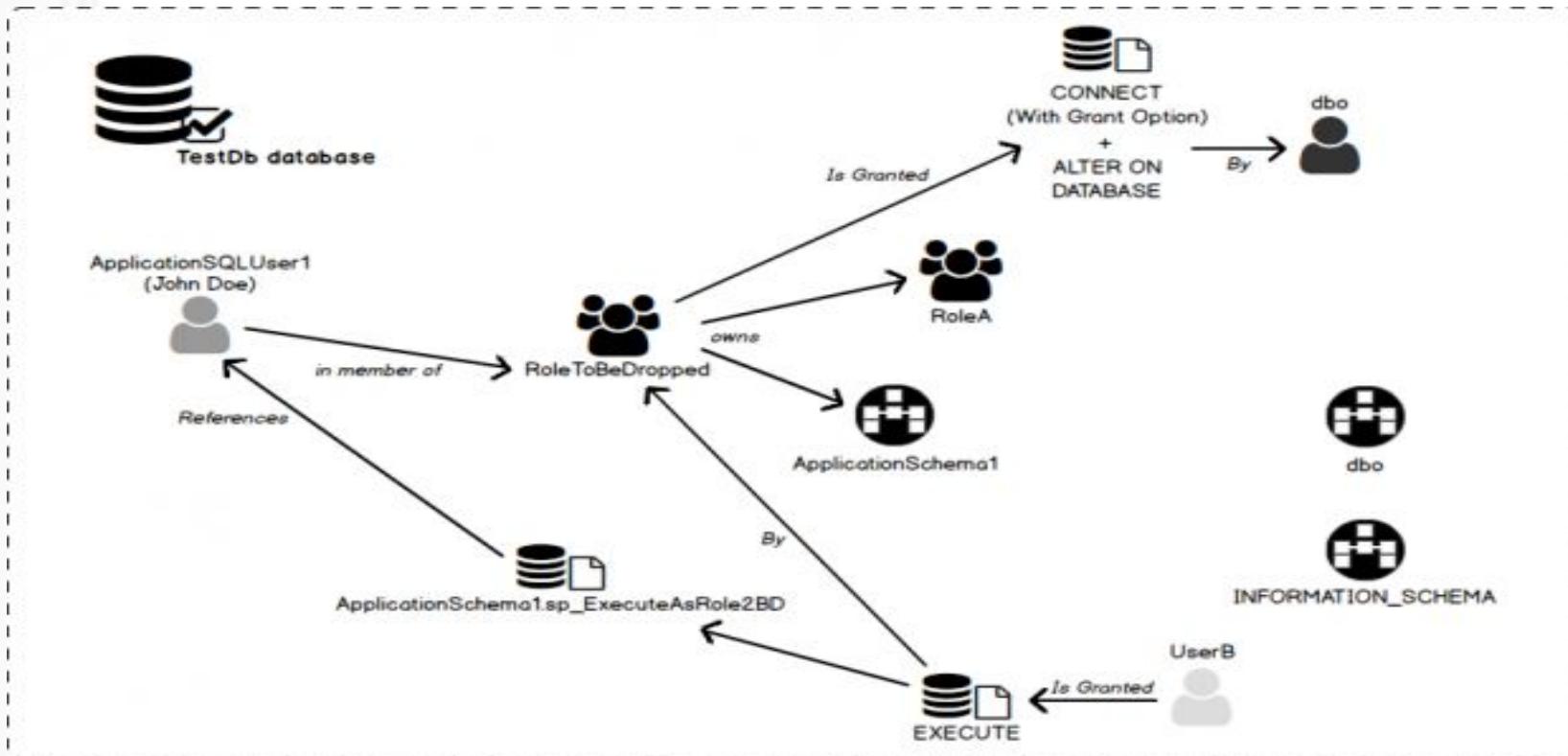
- **Roles:** papéis, funções, perfis.



Concessão de Privilégios

IGTI

- “Mix” de Opções: roles + roles com roles + privilégios diretamente.



Revogação de Privilégios



- Feita através do comando **REVOKE** da classe DCL.
- Revogação de privilégios para execução de comandos **DML/DDL/DCL**.
- **Sintaxe DML:** *REVOKE [Privilégio(s)] ON [Objeto] FROM [Usuário];*
- Opção **CASCADE**: remove permissões concedidas por um usuário com WITH GRANT OPTION.
- **Exemplos de Revogação de Privilégio DML**

REVOKE DELETE ON DatabaseLog FROM UsrAdventureSystem;

- Comando **DENY**: para negar privilégios em objetos (*alguns SGBDs).

Próxima Aula



- Capítulo 11 - Programação com a Linguagem T-SQL

A Linguagem SQL

Capítulo 11 - Programação com a Linguagem T-SQL

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 11. AULA 11.1. VISÕES (VIEWS) E CTE

PROF. GUSTAVO AGUILAR

Nesta Aula



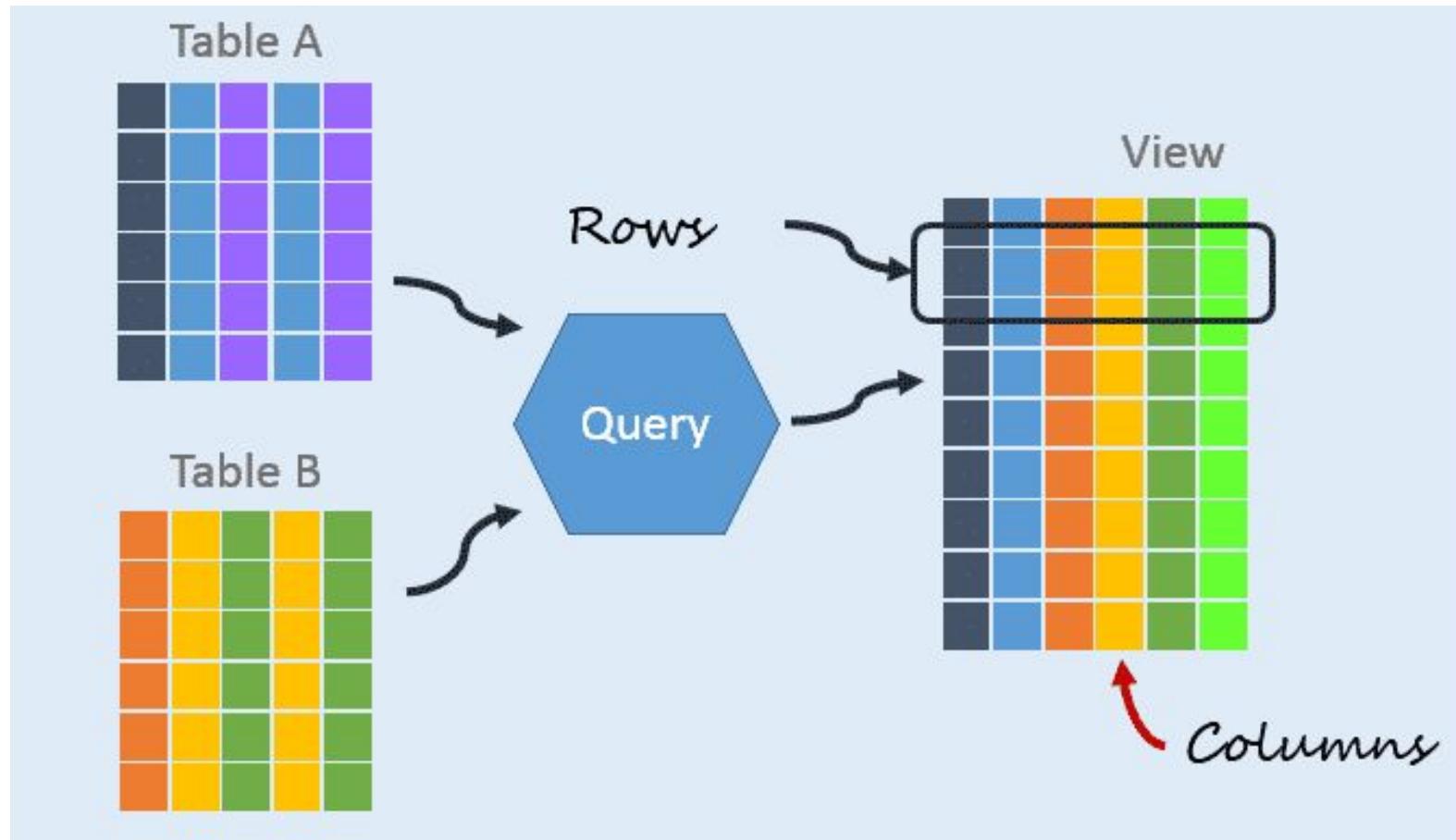
- Visões (Views)
- Demonstração com Views
- CTE (Common Table Expression)
- Demonstração com CTE

Visões (Views)

- É uma **query armazenada**, criada como um objeto no banco de dados;
- Não armazena dados e são vistas como tabelas virtuais dinâmicas para exibir dados específicos de uma ou mais tabelas / views;
- É **definida por uma query** no momento da sua criação, que especifica exatamente os dados que a view irá retornar;
- Criadas com o comando **CREATE VIEW** da classe DDL;
- Alteradas com o comando **ALTER VIEW**;
- Excluídas através do comando **DROP VIEW**.

Visões (Views)

iGTT



Visões (Views)



- Podem participar de joins, agrupamentos, como se fossem tabelas.
- Podem conter filtros na query de criação da view, mas podem ser filtradas em tempo de execução também.
- Query de criação da view **não pode contar a cláusula ORDER BY**.
 - Pode-se usar a cláusula ORDER BY na query que utilizará a view para retornar os dados □ externamente à view.

Visões (Views)



```
CREATE VIEW VW_Clientes_Australia AS
    SELECT CustomerID, AccountNumber
    FROM Sales.Customer
    WHERE TerritoryID IN (
        SELECT TerritoryID
        FROM Sales.SalesTerritory
        WHERE Name = 'Australia'
    );
```

Demonstração: Views



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU.gusta (56))'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 512 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 512 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Visões (Views)



▪ Vantagens do Uso de Views

- Simplificam os relacionamentos complexos entre tabelas, mostrando apenas os dados relevantes □ transparência e comodidade para usuários;
- Fornecem segurança, permitindo que os usuários vejam apenas as informações (colunas / linhas) que estão autorizados a ver □ não é necessário conceder permissões diretamente nas tabelas base da view;
- Fornecem uma camada de abstração, além de compatibilidade com versões anteriores do schema físico, caso as tabelas base mudem;
- Evitam problemas de performance causado por filtros indevidos ou ausentes nas queries que consultam diretamente as tabelas base da view.

- **CTE: Common Table Expression**
 - Função similar à de uma subquery ou de inline view (tabela derivada);
 - Não são persistidas no banco de dados como um objeto;
- **Vantagens:**
 - Conjunto de dados poder ser utilizado mais de uma vez na query, obtendo um ganho de performance (nesse cenário);
 - Código mais legível e podem ser recursivas.
- As seguintes cláusulas não podem ser usadas na CTE:
 - ORDER BY (exceto quando uma cláusula TOP for especificada);
 - INTO e a cláusula OPTION com hints.

CTE



```
WITH <Nome_da_CTE> (coluna1, coluna2, ...coluna n )  
AS ( Query_da_CTE )
```

```
SELECT .....
```

```
FROM Nome_da_CTE
```

Demonstração: CTE



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting ANSI settings. The right pane shows a Solution Explorer with various SQL files listed under a folder named 'SQLQuery4.sql'.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\M...LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\...GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Stored Procedures

A Linguagem SQL

CAPÍTULO 11. AULA 11.2. STORED PROCEDURES

PROF. GUSTAVO AGUILAR

Nesta Aula



- Stored Procedures
- Demonstração

Stored Procedures



- Stored Procedure = Procedimento Armazenado;
- Recurso de programação da Linguagem SQL criado (compilado) como um objeto no banco de dados;
- Diferentemente das views, que apenas encapsulam instruções SELECT, as procedures podem incluir uma gama mais ampla de instruções SQL, como UPDATE, INSERT e DELETE;
- Também podem ser usadas instruções de lógica de processamento, controle de fluxos, testes condicionais, variáveis e instruções para tratamento de erros na execução da procedure;

Stored Procedures



- Podem possuir parâmetros de entrada e saída;
- São criadas com o comando **CREATE PROCEDURE** da classe DDL;
- Alteradas pelo comando **ALTER PROCEDURE**;
- Excluídas através do comando **DROP**;
- Para executar uma procedure, é usado o comando **EXECUTE** (ou simplesmente **EXEC**) da classe DML.

Stored Procedures



```
CREATE PROCEDURE SP_Clientes_Australia AS
BEGIN
    SELECT CustomerID, AccountNumber
    FROM Sales.Customer
    WHERE TerritoryID IN (
            SELECT TerritoryID
            FROM Sales.SalesTerritory
            WHERE Name ='Australia'
    )
    ORDER BY CustomerID DESC
END;
```

Stored Procedures



- Com TRY / CATCH

```
CREATE PROCEDURE [dbo].[sp_open_ars_requests] @descText varchar(5000) .....variáveis de entrada e saída
AS
--VARIÁVEIS LOCAIS
BEGIN TRY
---COMANDOS...
    set @sql = 'exec master.dbo.xp_cmdshell "c:\wget\bin\wget -O "+@parameter+'\ARS_output.xml" -v "+@url+"""
    exec (@sql)
END TRY
BEGIN CATCH
    insert into Log_General (LogSource, LogText) values
    ('sp_open_ars_requests','Error: '+CAST(ERROR_NUMBER() as varchar(10))+'. Message: '+ERROR_MESSAGE())
    If not exists (select * from dbo.Alerts_Instance where ObjectName = 'sp_open_ars_requests'
    and AdComments = 'Procedure failed - Verify table Log_General'
    and FlagAlert in (0,1)
    )
    begin
        insert into dbo.Alerts_Instance (InstanceId, DatabaseName, ObjectName, AdComments, FlagAlert, ID_Cat)
        values ('Local','Not applicable','sp_open_ars_requests','Procedure failed - Verify table Log_General',0, @IDCAT)
    end
END CATCH
```

Demonstração: Stored Procedures



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting file properties, enabling full-text search, and adjusting database settings like ANSI_PADDING and ANSI_NULL_DEFAULT. The 'Object Explorer' pane on the left shows the connection details and database structure. The 'Solution Explorer' pane on the right lists various SQL scripts and files related to the demonstration.

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [AdventureWorks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Stored Procedures



- **Vantagens do Uso de Procedures**
 - **Mais segurança para os objetos do banco de dados:** os usuários tem permissão para executar uma determinada procedure, sem ter permissão para acessar diretamente os objetos que a procedure acessa.
 - **Reaproveitamento de Código:** a lógica é criada uma vez, no código da procedure, e depois **reutilizada** várias vezes, em diversos módulos da aplicação ou até mesmo em aplicações diferentes, bastando fazer uma chamada à procedure.
 - **Programação modular de fácil manutenção:** se houver necessidade de alteração de código, **só é preciso alterar a procedure** em questão.

Stored Procedures



- **Vantagens do Uso de Procedures**
 - **Redução do tráfego de rede:** *EXEC Nome_da_Procedure* versus *Execução Query AdHoc*.
 - **Performance:** reutilização do plano de execução **pré-compilado**, reduzindo o tempo necessário para executar a procedure, quando comparado com o mesmo código executado sem ser via procedure, pois não gasta tempo para gerar um novo plano de execução.

Próxima Aula



- Functions

A Linguagem SQL

CAPÍTULO 11. AULA 11.3. FUNCTIONS

PROF. GUSTAVO AGUILAR

Nesta Aula



- ❑ Functions
 - ❑ Escalar
 - ❑ TVF (Table-Value Function)
- ❑ Demonstração

Functions



- Assim como as procedures, é armazenada como um objeto persistente (compilado) no banco de dados;
- Principais diferenças com relação à procedure:

Função (User Defined Function)	Stored Procedure
A função deve retornar um valor	Procedure pode ou não retornar valores.
Permitirá apenas instruções SELECT, não permitirá usar outras instruções DML (INSERT / UPDATE / DELETE)	Pode ter instruções de seleção e instruções DML, como inserir, atualizar, excluir
Transações não são permitidas dentro de funções	Transações são permitidas
Stored Procedures não podem ser chamadas de uma função	Procedures podem chamar funções
As funções podem ser chamadas a partir de uma instrução SELECT.	Procedures não podem ser chamadas nas instruções Select / Where / Having e assim por diante

Functions

- **Escalar** ☐ retornar um valor.

CREATE FUNCTION <nome> (@<nome_parâmetro> AS <data_type>, ...)

RETURNS tipo_de_dados_retornado AS

BEGIN

Corpo da função

RETURN (<expressão_escalar>)

END

Functions

- **TVF (Table-Value Function)** □ multivalorada.
 - Retornar uma tabela;
 - Aceita parâmetros de entrada, podendo consultá-los na instrução SELECT do seu código;
 - Em linhas gerais a TVF encapsula uma única instrução SELECT, retornando uma tabela virtual para a query:

```
CREATE FUNCTION <nome> (@<nome_parâmetro> AS <data_type>, ...)  
RETURNS TABLE AS  
BEGIN  
    Corpo da função  
    RETURN (<instrução_SELECT>)  
END
```

Demonstração: Functions



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'ULTRA-GU (SQL Server 14.0.2027.2 - ULTRA-GU.gusta (56))'. The Solution Explorer on the right lists various SQL scripts from '01-Criar Banco.sql' to 'SQLQuery4.sql'. The central pane displays a T-SQL script for creating the 'AdventureWorks2017' database:

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' , SIZE = 128 , MAXSIZE = 1024 , FILEGROWTH = 64 )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' , SIZE = 32 , MAXSIZE = 1024 , FILEGROWTH = 1 )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Triggers

A Linguagem SQL

CAPÍTULO 11. AULA 11.4. TRIGGERS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Triggers
- Demonstração

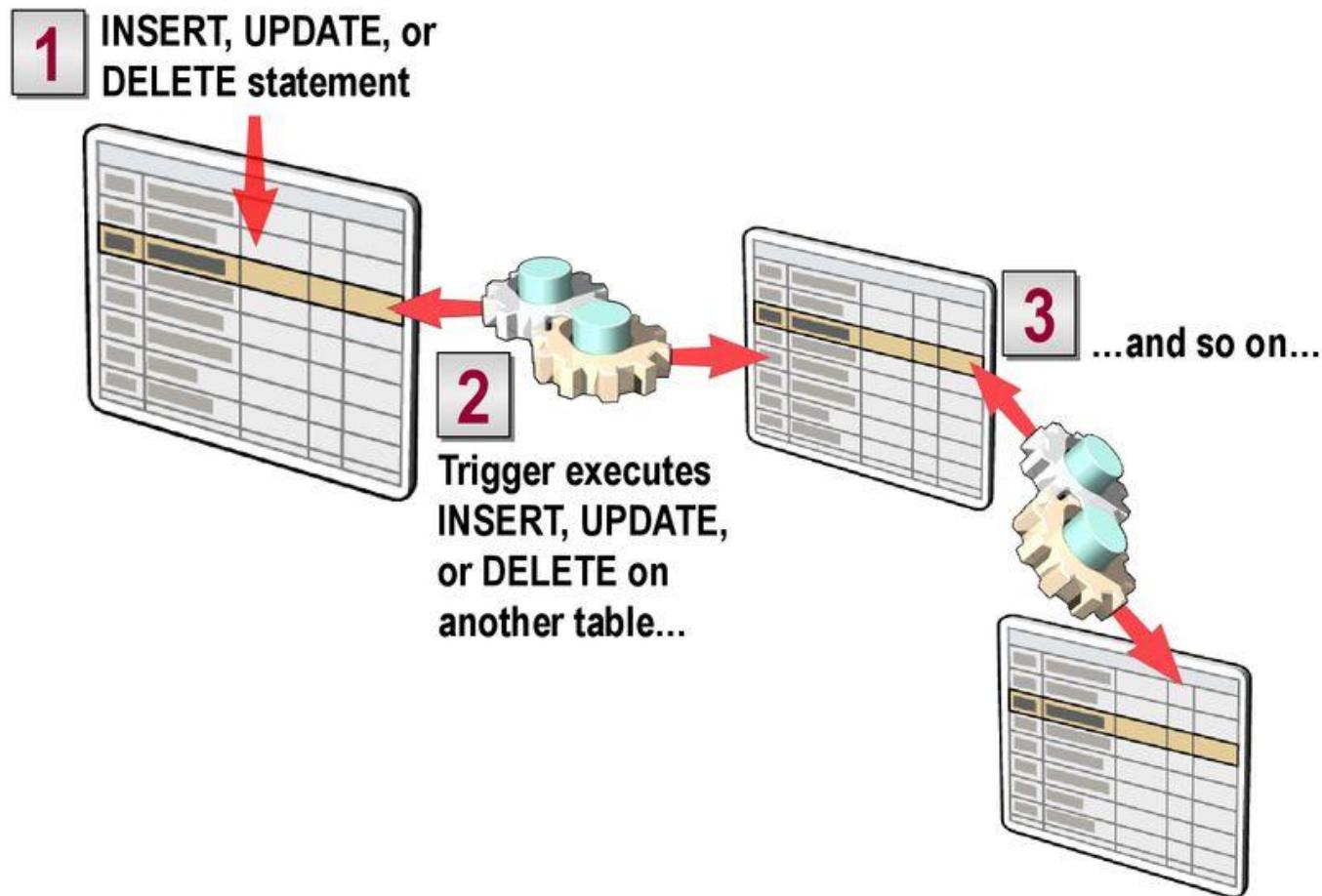
Triggers



- Triggers (gatilhos) são objetos programáticos, também armazenados de forma compilada no banco de dados;
- São disparados automaticamente, mediante uma ação no **banco de dados, instância ou objeto**;
- **DML TRIGGER:** disparadas quando ocorre INSERT, UPDATE ou DELETE em uma tabela ou view;
- **DDL TRIGGERS:** são disparadas quando ocorrem eventos DDL (CREATE, ALTER, DROP, etc.);
- **LOGON TRIGGERS:** dispara quando uma nova sessão é estabelecida.

Triggers

iGTTI



Triggers



- Sintaxe DML Trigger

```
CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name  
ON { table | view }  
[ WITH <dml_trigger_option> [ ,...n ] ]  
{ FOR | AFTER | INSTEAD OF }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
[ WITH APPEND ]  
[ NOT FOR REPLICATION ]  
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ]> }
```

Triggers



- Sintaxe DDL Trigger

```
CREATE [ OR ALTER ] TRIGGER trigger_name  
ON { ALL SERVER | DATABASE }  
[ WITH <ddl_trigger_option> [ ,...n ] ]  
{ FOR | AFTER } { event_type | event_group } [ ,...n ]  
AS { sql_statement [ ; ] [ ,...n ] } | EXTERNAL NAME <method specifier> [ ; ]  
}
```

Triggers



- Sintaxe LOGON Trigger

```
CREATE [ OR ALTER ] TRIGGER trigger_name  
ON ALL SERVER  
[ WITH <logon_trigger_option> [ ,...n ] ]  
{ FOR | AFTER } LOGON  
AS { sql_statement [ ; ] [ ,...n ] } | EXTERNAL NAME <method specifier> [ ; ]  
}
```

Demonstração: Triggers



Demo

A screenshot of Microsoft SQL Server Management Studio (SSMS) running on a Windows operating system. The window title is "Disciplina Aplicações com Linguagem SQL - Microsoft SQL Server Management Studio". The central pane displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting its properties (CONTAINMENT = NONE, ON PRIMARY, LOG ON, etc.), enabling full-text search, and adjusting ANSI settings. The right-hand pane shows the Solution Explorer with various SQL files listed, such as 01-Criar Banco.sql, 02-Criar Tabelas.sql, and others related to database creation and management. The bottom status bar indicates the connection is "Connected (1/1)" and the current database is "master".

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON [PRIMARY]
( NAME = 'Adventureworks2017', FILENAME = 'N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = 'AdventureWorks2017_log', FILENAME = 'N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Linked Server e Query Distribuída

A Linguagem SQL

CAPÍTULO 11. AULA 11.5. LINKED SERVER E QUERY DISTRIBUÍDA

PROF. GUSTAVO AGUILAR

Nesta Aula

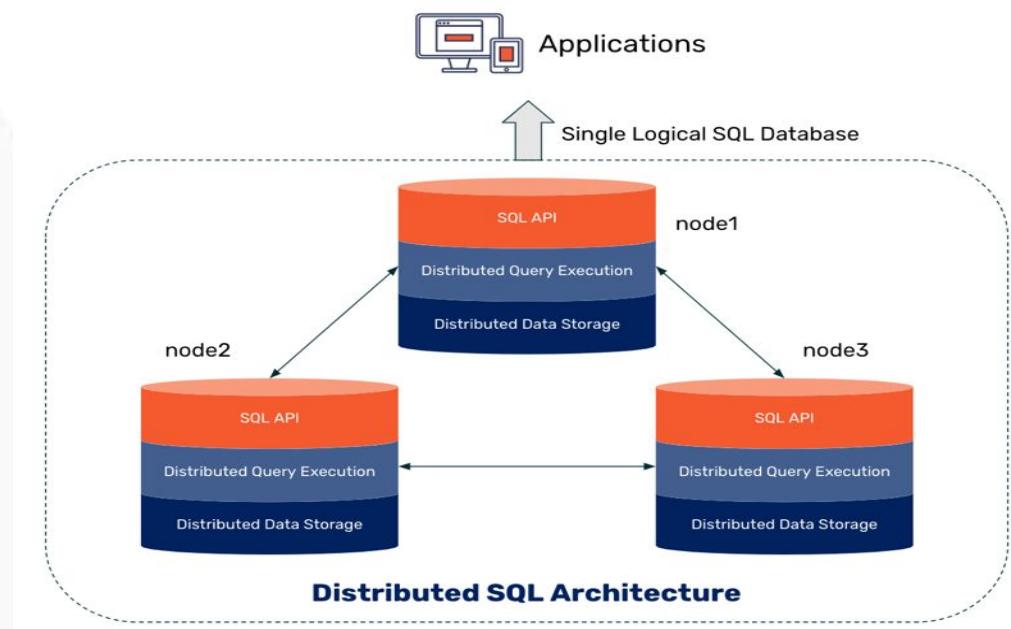


- Query Distribuída
- Linked Server
- Demonstração

Query Distribuída

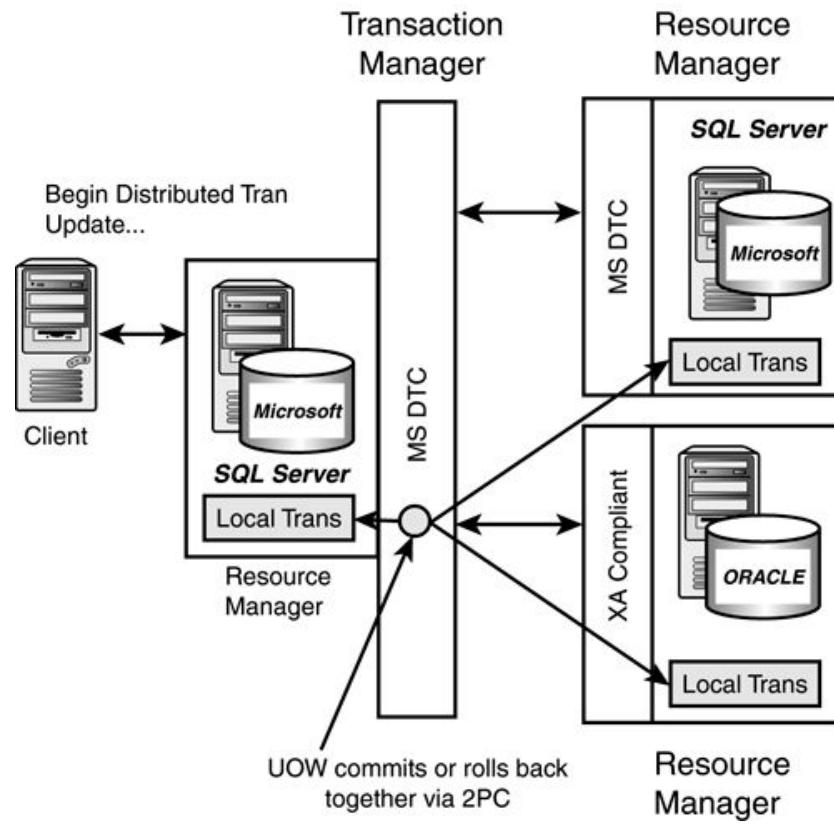
IGTI

- Consulta que executa em mais de uma instância de banco de dados;
- Comunicação entre as instâncias é feita usando protocolos de rede;
- Transparência da localização física dos dados



Query Distribuída

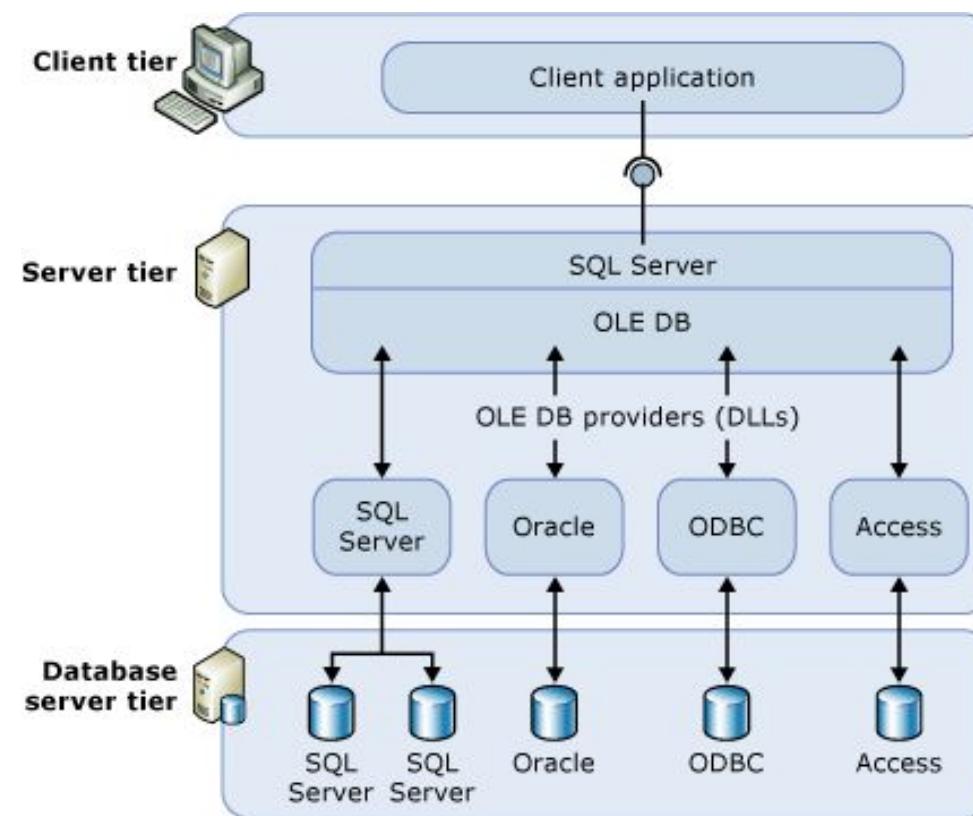
- Possível também realizar transações distribuídas
 - **Protocolo TWO PHASE COMMIT**



Linked Server

Recurso tradicional para se **executar queries distribuídas**.

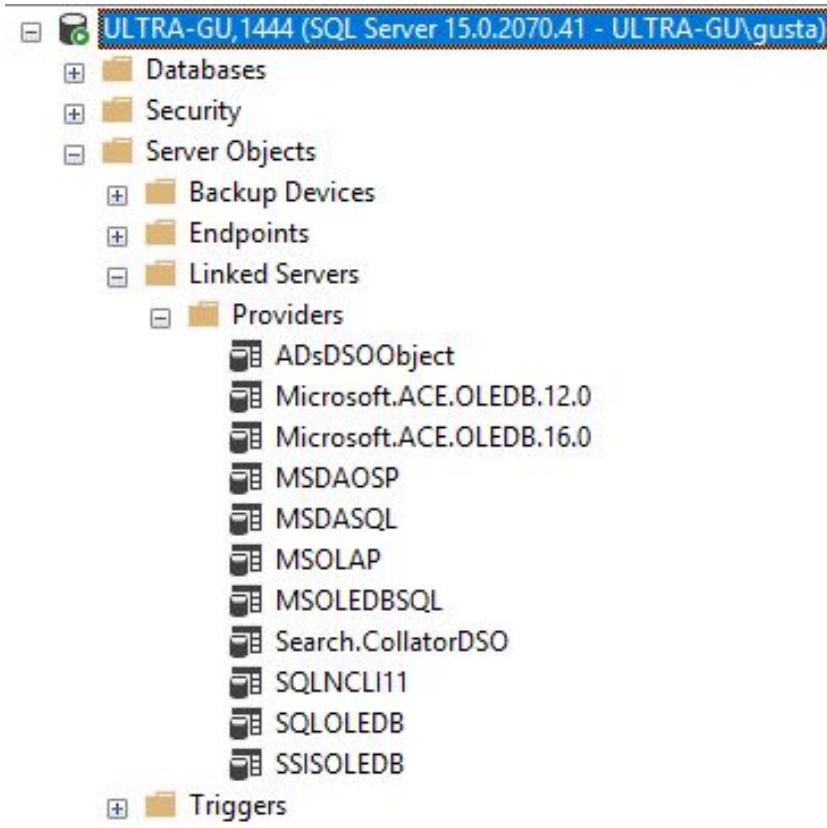
O **Linked Server** (no Oracle, chamado de **dblink**) permite que uma instância SQL Server **leia dados de fontes** de dados remotas e **executem comandos** nos servidores de banco de dados remotos.



Linked Server



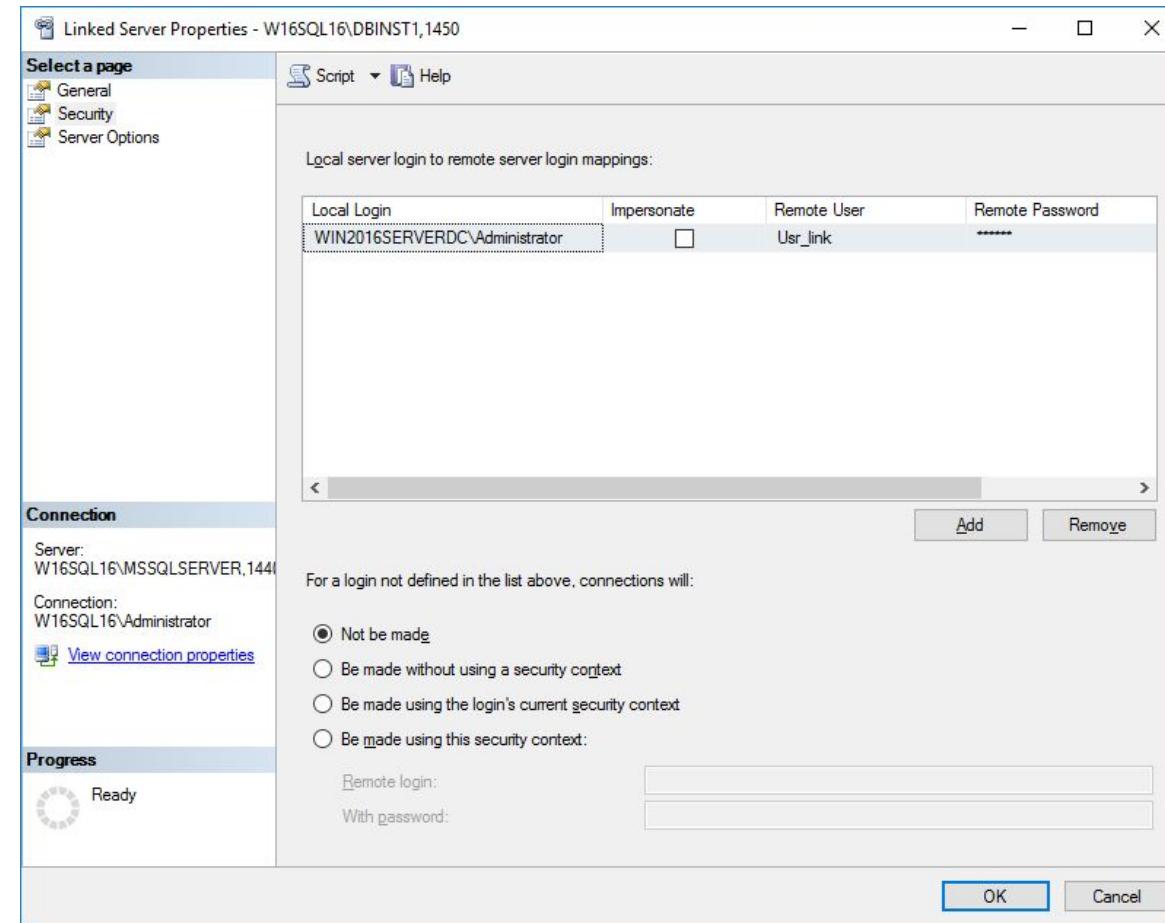
- Pode ser configurado para acessar dados que estejam no Microsoft **Access**, no **Excel** e até mesmo na nuvem, como o **Azure CosmosDB**.
- Funciona com qualquer **provider OLE DB** que implemente as interfaces de OLE DB exigidas;
- Provider OLE DB / ODBC de terceiro deve estar instalado previamente.



Linked Server



- Requer uma configuração de segurança para mapeamento de logins locais e remotos;
- Não aceita grupos no mapeamento: usar* opção “Be made using the login's current security context”;
- Linked server público*: opção “Be made using this security context”.



Linked Server



Consultas Distribuídas

□ Uso com **Four-Part-Name**:

- Cláusula FROM: **LinkedServer.Banco.Schema.Objeto**;
- Permite joins entre objetos de locais remotos distintos;
- Menos performáticos na maioria dos casos;
 - Filtros são feitos localmente: maior tráfego de dados na rede;
- Menos flexíveis.



Linked Server



Consultas Distribuídas

□ Uso com **OpenQuery / Execute AT**

- Cláusula FROM:
 - OPENQUERY (LinkedServerName, 'Query');
 - EXECUTE ('Query') AT [LinkedName]
- Não permite joins entre objetos de locais remotos distintos no mesmo Openquery / Execute AT;
- Mais performáticos: retorna somente o resultado já filtrado.
- Mais flexíveis: envia o comando deixando o parse a cargo do remoto;
- Execute AT requer RPC no Linked Server.



Linked Server



Transações Distribuídas

□ Uso com **Four-Part-Name**:

- Cláusula **INTO**: LinkedServerName.Banco.Schema.Objeto;
- Requer provider com Nested Transaction (transações aninhadas);
- Menos performático.



Linked Server



Transações Distribuídas

- Uso com **OpenQuery / Execute AT**
 - Cláusula: EXECUTE ('Query') AT [LinkedName]
 - Mais performático;
 - Mais flexíveis: envia o comando deixando o parse a cargo do servidor remoto.



Demonstração: Linked Server



Demo

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays a T-SQL script for creating the AdventureWorks2017 database. The script includes commands for creating the database, setting containment to none, defining primary and log files, enabling full-text search, and adjusting ANSI settings. The right-hand pane shows a Solution Explorer with various SQL scripts listed, such as 01-Criar Banco.sql, 02-Criar Tabelas.sql, etc. The bottom status bar indicates a connection to ULTRA-GU (14.0 RTM) and ULTRA-GU.gusta (56).

```
***** Object: Database [AdventureWorks2017] Script Date: 07/09/2019 12:04
CREATE DATABASE [AdventureWorks2017]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Adventureworks2017', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\DATA\Adventureworks2017.mdf' )
LOG ON
( NAME = N'AdventureWorks2017_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\140\LOG\Adventureworks2017.ldf' )
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Adventureworks2017].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_NULLS ON
GO

ALTER DATABASE [AdventureWorks2017] SET ANSI_PADDING ON
GO
```

Próxima Aula



- Capítulo 12 - Mapeamento Objeto Relacional

A Linguagem SQL

Capítulo 12 - Mapeamento Objeto Relacional

PROF. GUSTAVO AGUILAR

A Linguagem SQL

CAPÍTULO 12. AULA 12.1. INTRODUÇÃO AO MAPEAMENTO OBJETO RELACIONAL

PROF. GUSTAVO AGUILAR

Nesta Aula

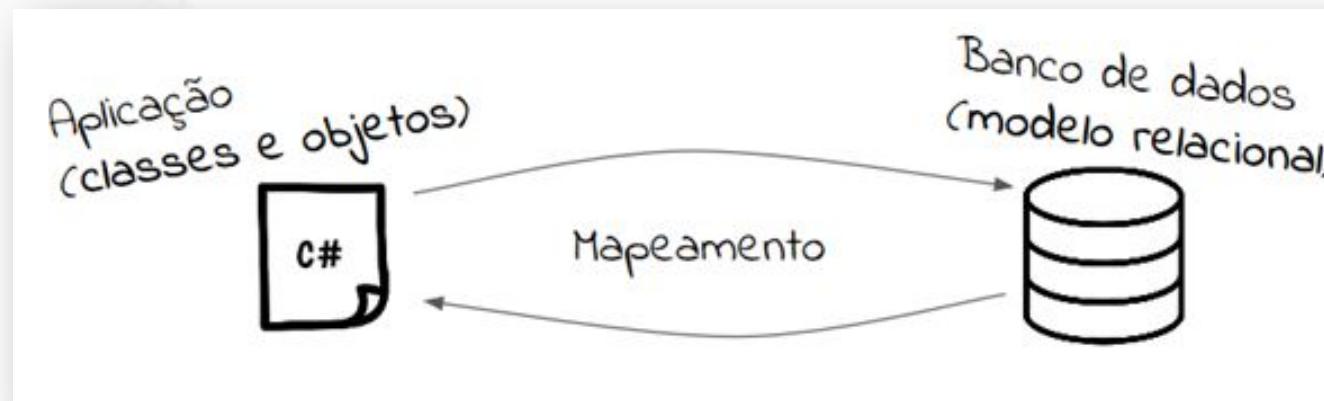


- Mapeamento Objeto Relacional
- As Regras de Mapeamento Objeto Relacional
- Ferramentas ORM de Mercado

Mapeamento Objeto Relacional

IGTI

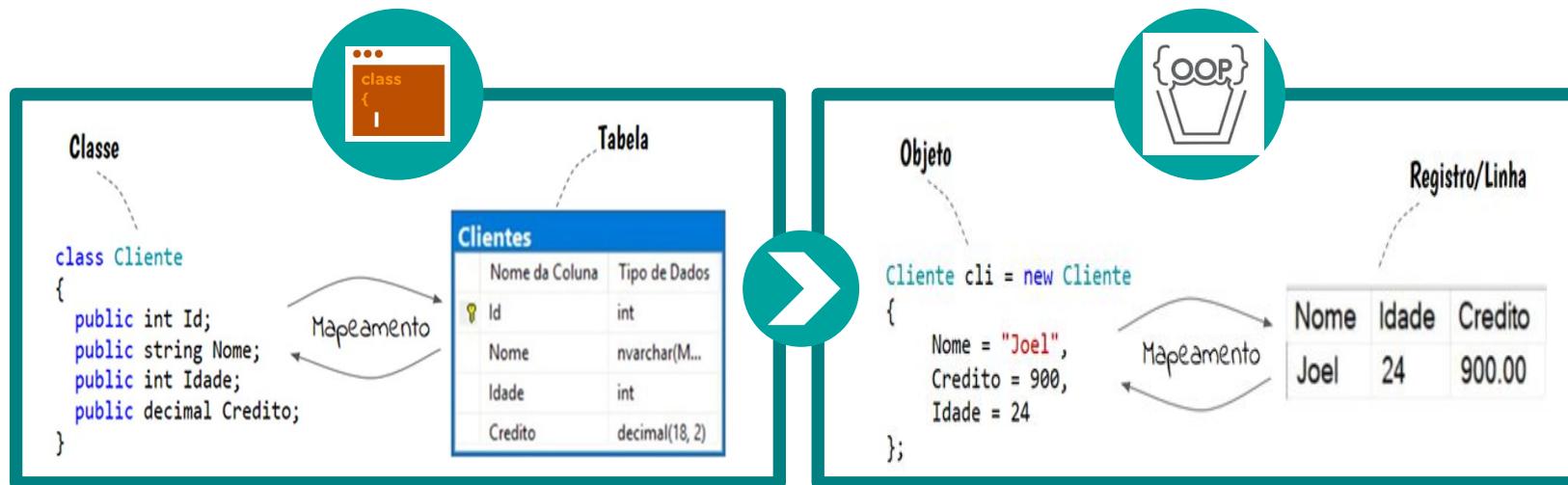
- Do inglês *Object Relational Mapping* □ **ORM**;
- Técnica usada para conversão de dados entre linguagens de programação orientadas a objetos e bancos de dados relacionais;



Mapeamento Objeto Relacional

IGTI

- Na prática: associação entre classes e objetos com tabelas e linhas.
- Nível de código da aplicação □ trabalha-se com classes e objetos.
- Nível do banco de dados □ traduzidos para tabelas e linhas.



Mapeamento Objeto Relacional

IGTI

- Mapeamento □ **Camada de Persistência de Objetos;**

- Uma biblioteca ou trecho de código;
 - Transparência do processo de persistência de dados;
 - Armazenamento em meio não volátil:
 - ORM □ em banco de dados relacional;
 - Para o programador:
 - Abstração dos comandos SQL do SGBD;
 - Como se estivesse em um ambiente 100% OO.



Regras de Mapeamento



Objeto Relacional

- Criação de uma **tabela** para cada **classe**;
- Criação de uma **coluna** para cada **atributo simples** do objeto;
- Criação de uma **coluna** para o **OID** (*Object Identifier*);
- Definição de uma **chave primária** para cada tabela: **cada objeto é único** e garantido pelo OID.
 - OID como chave primária ☐ garante a unicidade dos registros na tabela.

Regras de Mapeamento

Objeto Relacional



- **Atributos compostos:**

- mapeados em **várias colunas**, ou
- consolidados em apenas uma.

- **Atributos multivalorados:**

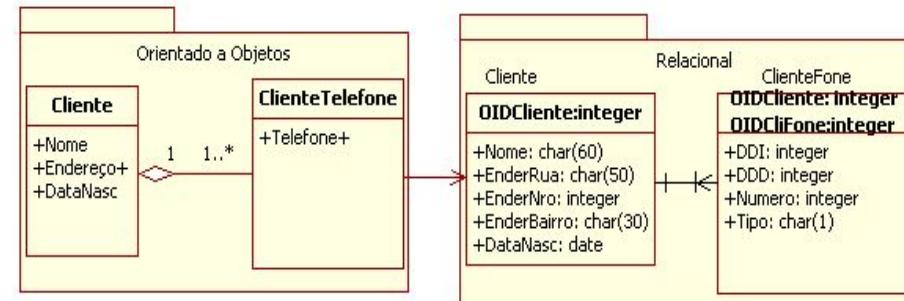
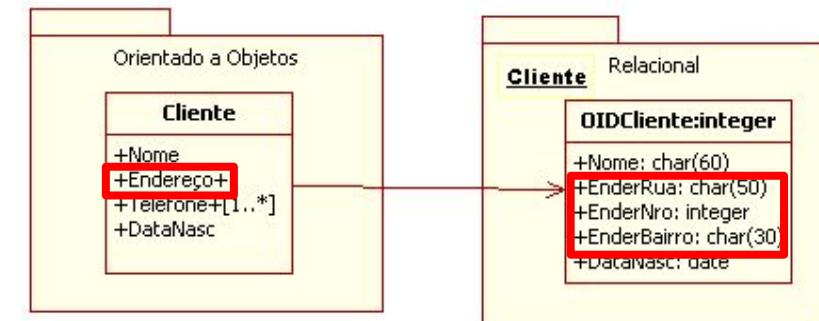
- **Nova tabela** (entidade fraca)

- **Chave primária:**

- **Chave estrangeira** (PK da tabela criada

para a classe com o atributo multivalorado

- + **PK da nova tabela.**



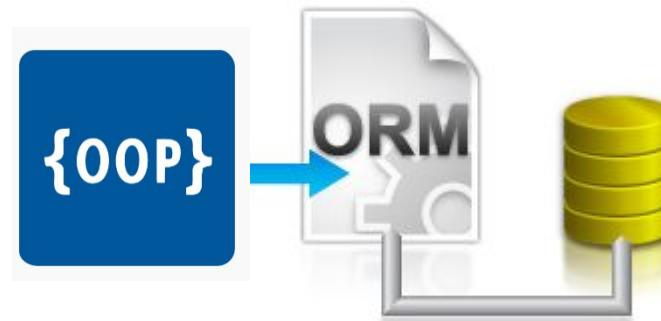
Mapeamento Objeto Relacional

IGTI

- Mapeamento concluído (nível de modelo de dados) □ definir como será feita a persistência dos dados:
 - De forma manual pelo programador □ haja código !!!

OU

- **Com uma Ferramenta ORM:**
 - Mapeamento configurado na ferramenta;
 - Persistência feita implicitamente pelo framework;
 - Sem o ônus de construção de código para isso;
 - Mais segurança e confiança em todo o processo de persistência de dados.



Ferramentas ORM de Mercado



- **Hibernate** HIBERNATE

- ✓ Grátis;
- ✓ Para linguagem **Java**;
- ✓ <http://hibernate.org/orm/>.

- **NHibernate** NHibernate

- ✓ Grátis;
- ✓ Para **.NET**;
- ✓ <https://nhibernate.info/>.

- **Django** django

- ✓ Grátis;
- ✓ Para linguagem **Python**;
- ✓ www.djangoproject.com/.

- **Ruby on Rails** RAILS

- ✓ Grátis;
- ✓ Para linguagem **Ruby**;
- ✓ <https://rubyonrails.org/>.

- **Laravel** Laravel

- ✓ Grátis;
- ✓ Para linguagem **PHP**;
- ✓ <https://laravel.com/>.

- **Entity Framework** Microsoft

- ✓ Grátis / Paga (extensões);
- ✓ Para **.NET**;
- ✓ <https://docs.microsoft.com/e>.

- **LLBLGen Pro** | LLBLGen Pro

- ✓ Paga;
- ✓ Para **.NET**;
- ✓ <http://www.llblgen.com/>.

- **Sequelize**



- ✓ Grátis;
- ✓ Para **Node.js**;
- ✓ <http://docs.sequelizejs.com/>.

Próxima Aula



- O ORM Sequelize

A Linguagem SQL

CAPÍTULO 12. AULA 12.2. O ORM SEQUELIZE

PROF. GUSTAVO AGUILAR

Nesta Aula

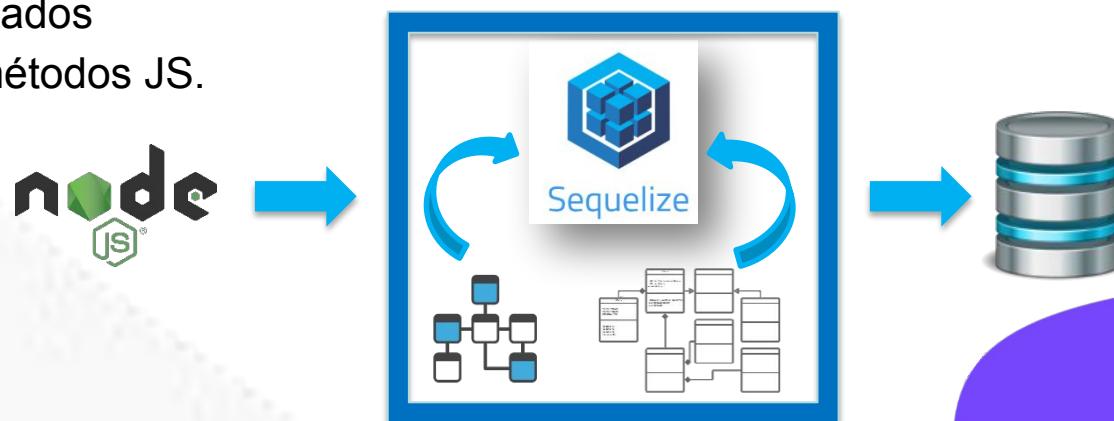


- ❑ Introdução ao Sequelize
- ❑ Dialetos e Opções da Engine
- ❑ Utilização Básica do Sequelize

Introdução ao Sequelize

IGTI

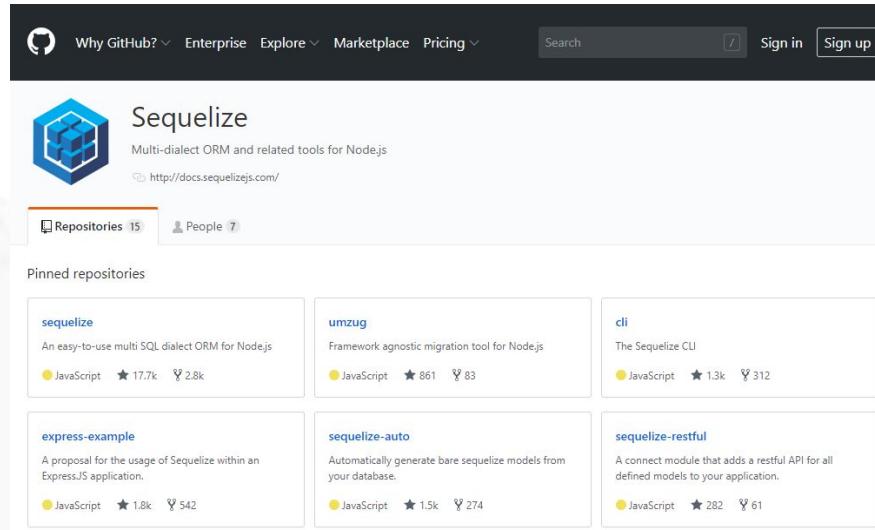
- Ferramenta ORM **open source** para **Node.js** (versão 4 / superior);
- Multidialeto (comandos SQL):
 - *MySQL / MariaDB, PostgreSQL, SQLite e SQL Server.*
- Objetos **JavaScript** mapeados em tabelas, colunas e linhas:
 - Manipulação de dados relacionais com métodos JS.



Introdução ao Sequelize



- Apesar de **gratuita** ☐ suporte a **transações, relacionamentos, leitura em cenários de replicação** e outros recursos avançados;
- Boa comunidade no **GitHub**: <https://github.com/sequelize>.
- Ótima **documentação**: <http://docs.sequelizejs.com>.



API Sequelize



- Codificada a partir da classe principal **sequelize**;
- Para usá-la, basta importa-la no código Java:

```
const Sequelize = require ('sequelize');
```

- Inúmeros métodos para:
 - Definir modelos;
 - Mapear objetos;
 - Persistir e manipular dados.
- Documentação completa:
 - <http://docs.sequelizejs.com/class/lib/sequelize.js~Sequelize.html>.

The screenshot shows a detailed view of the Sequelize API documentation for the `fn` method. On the left, there's a sidebar with navigation links for Home, Reference, Join us on Slack, and a search bar. The main content area has a header for the `fn` method, which is described as a public static function. It provides a brief description, usage examples, parameters, and return values. The code example shows how to convert a user's username to uppercase using the `fn('upper')` function.

Name	Type	Attribute	Description
fn	String		The function you want to call
args	any		All further arguments will be passed as arguments to the function

Return:
`fn`

Dialeto

- **Dialeto** ☐ SGBD que será utilizado;
- Biblioteca de conexão para o dialeto deve ser instalada;
 - Não é necessário importa-la ☐ o Sequelize se incube disso automaticamente.
 - *npm install --save pg pg-hstore* ☐ para PostgreSQL
 - *npm install --save mysql2* ☐ para MySQL
 - *npm install --save sqlite3* ☐ para SQLite
 - *npm install --save mariasql* ☐ para MariaDB
 - *npm install --save tedious* ☐ para SQL Server

Dialeto



- Cada dialeto conterá as respectivas configurações para o SGBD em questão.

```
var sequelize = new Sequelize('database', null, null,
{dialect: 'mysql',
port: 3306
replication:
{ read:
[ { host: '8.8.8.8', username: 'read-username', password: 'some-password' },
{ host: '9.9.9.9', username: 'another-username', password: null }
],
write: { host: '1.1.1.1', username: 'write-username', password: 'any-password' }
},
pool: { max: 20,
idle: 30000
}
})
```

Utilização Básica do Sequelize



- Conexão com o banco

```
var Sequelize = require('sequelize')
, sequelize = new Sequelize('BDEquipamentos', 'usrsequelize', 'sequelize',
    {
        host: 'localhost',
        dialect: "mysql",
        port:    3306,
        operatorsAliases: false
    });

sequelize
.authenticate()
.then(function(err) { console.log('SUCESSO!!! Conexao estabelecida.');?>
, function (err)
{ console.log('ERRO!!! Nao foi possivel conectar.', err);});
```

Utilização Básica do Sequelize



▪ Execução de Query SQL (RAW)

- Flexibilidade para execução de query SQL nativa do SGBD de dentro do código Java;
- Executadas através do método query().

```
sequelize.query("SELECT * FROM TipoEquipamento")
  .then(RegistrosTabela => {console.log(RegistrosTabela) })
```

Utilização Básica do Sequelize



▪ Definição do Modelo

- Feita usando o método define();
- O Sequelize tem suporte a vários tipos de dados:
- <http://docs.sequelizejs.com/manual/tutorial/models-definition.html#data-types>

```
var TipoEquipamento = sequelize.define('TipoEquipamento',
  {
    cod_tipo_equipamento: Sequelize.INTEGER,
    dsc_tipo_equipamento: Sequelize.STRING
  });

```

The diagram illustrates the Sequelize model definition code with three callout boxes and arrows pointing to specific parts of the code:

- A box labeled "Nome do objeto" (Object name) has an arrow pointing to the first argument of the `sequelize.define` method, "TipoEquipamento".
- A box labeled "Nome da coluna" (Column name) has arrows pointing to both column definitions: "cod_tipo_equipamento" and "dsc_tipo_equipamento".
- A box labeled "Tipo de dados" (Data type) has arrows pointing to the data types specified in the column definitions: "Sequelize.INTEGER" and "Sequelize.STRING".

Utilização Básica do Sequelize



▪ Sincronização do Schema Físico

- Criação do schema físico;
- Para os dados poderem ser persistidos;
- Feito com o método sync ()

```
sequelize.sync({ force: true })  
  .then(function(err) { console.log('Tabela criado com sucesso!'); },  
        function (err) { console.log('Erro ao criar a tabela.', err); });
```

Se a tabela existir ➔ dropa e a recria

Utilização Básica do Sequelize



```
var Sequelize = require('sequelize')
, sequelize = new Sequelize('BDEquipamentos', 'usrsequelize', 'sequelize',
{
    host: 'localhost',
    dialect: "mysql",
    port: 3306,
    operatorsAliases: false
});
sequelize.authenticate()
.then ( function(err) { console.log('SUCESSO!!! Conexao estabelecida.'),;
    function (err) { console.log('ERRO!!! Nao foi possivel conectar.', err); }
);
var TipoEquipamento = sequelize.define('TipoEquipmento',
{
    cod_tipo_equipamento: Sequelize.INTEGER,
    dsc_tipo_equipamento: Sequelize.STRING
});

sequelize.sync({ force: true })
.then( function(err) { console.log('Tabela criada com sucesso!'),;
    function (err){ console.log('Erro ao criar a tabela.', err); }
);
```

Utilização Básica do Sequelize



▪ Persistência dos dados

- Etapa de inserção dos dados (persistência da instância de um objeto)
- Pode ser feita em dois passos:
 - Cria o objeto e depois o salva (métodos build e save);
 - Mais flexível, com opções antes de persistir o objeto.

```
var tpequipamento = TipoEquipamento.build(  
    {  
        cod_tipo_equipamento: 01,  
        dsc_tipo_equipamento: 'Notebook'  
    });  
  
tpequipamento.save().then( function(err) { console.log('Registro inserido com sucesso!'); },  
    function (err){ console.log('Erro ao inserir o registro', err);} );
```

Utilização Básica do Sequelize



▪ Persistência dos dados

- Etapa de inserção dos dados (persistência da instância de um objeto)
- Pode ser feita com um passo:
 - Cria o objeto e já persiste (método create);
 - Mais performática.

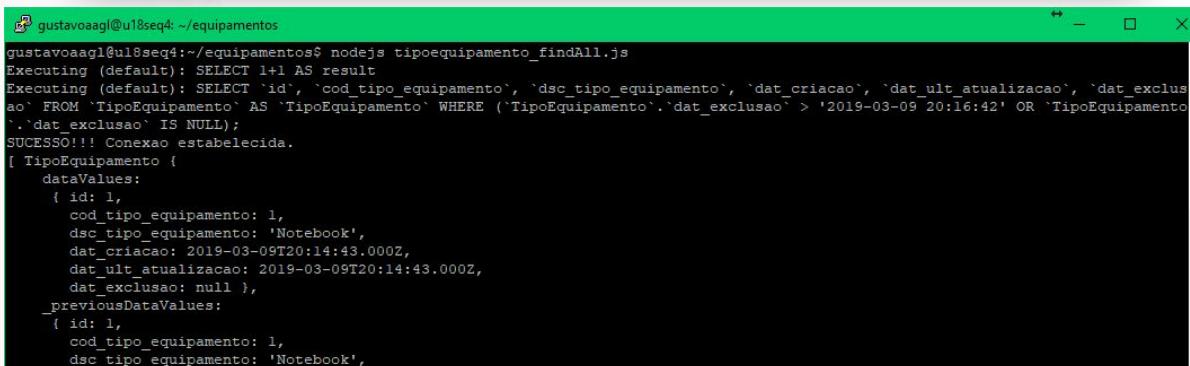
```
var tpequipamento = TipoEquipamento.create(  
  {  
    cod_tipo_equipamento: 01,  
    dsc_tipo_equipamento: 'Notebook'  
  })  
  .then(function(err) { console.log('Registro inserido com sucesso!'); },  
        function (err){ console.log('Erro ao inserir o registro', err);}  
  );
```

Utilização Básica do Sequelize

IGTI

- Leitura de dados

```
var tpequipamento = TipoEquipamento.findAll().  
then (tpequipamento => {  
    console.log(tpequipamento)  
})
```



A terminal window titled 'gustavoagl@ul8seq4: ~/equipamentos' showing the execution of a Sequelize query. The command 'nodejs tipoequipamento_findAll.js' is run, followed by two SELECT statements. The first statement is 'Executing (default): SELECT 1+1 AS result'. The second statement is 'Executing (default): SELECT `id`, `cod_tipo_equipamento`, `dsc_tipo_equipamento`, `dat_criacao`, `dat_ult_atualizacao`, `dat_exclusao` FROM `TipoEquipamento` AS `TipoEquipamento` WHERE (`TipoEquipamento`.`dat_exclusao` > '2019-03-09 20:16:42' OR `TipoEquipamento`.`dat_exclusao` IS NULL);'. The output shows the success message 'SUCESSO!!! Conexao estabelecida.' followed by the Sequelize object representation of the data.

```
gustavoagl@ul8seq4: ~/equipamentos  
gustavoagl@ul8seq4:~/equipamentos$ nodejs tipoequipamento_findAll.js  
Executing (default): SELECT 1+1 AS result  
Executing (default): SELECT `id`, `cod_tipo_equipamento`, `dsc_tipo_equipamento`, `dat_criacao`, `dat_ult_atualizacao`, `dat_exclusao` FROM `TipoEquipamento` AS `TipoEquipamento` WHERE (`TipoEquipamento`.`dat_exclusao` > '2019-03-09 20:16:42' OR `TipoEquipamento`.`dat_exclusao` IS NULL);  
SUCESSO!!! Conexao estabelecida.  
[ TipoEquipamento {  
    dataValues:  
      { id: 1,  
        cod_tipo_equipamento: 1,  
        dsc_tipo_equipamento: 'Notebook',  
        dat_criacao: 2019-03-09T20:14:43.000Z,  
        dat_ult_atualizacao: 2019-03-09T20:14:43.000Z,  
        dat_exclusao: null },  
    previousDataValues:  
      { id: 1,  
        cod_tipo_equipamento: 1,  
        dsc_tipo_equipamento: 'Notebook',  
        dat_criacao: 2019-03-09T20:14:43.000Z,  
        dat_ult_atualizacao: 2019-03-09T20:14:43.000Z,  
        dat_exclusao: null } } ]
```

Utilização Básica do Sequelize



- Leitura de dados

```
SELECT * FROM TipoEquipamento
```

```
WHERE cod_tipo_equipamento IN (12, 13);
```

```
TipoEquipamento.findAll({ where: {cod_tipo_equipamento:  
{ [Op.or]: [12, 13] }}});
```

```
[Op.and]: {a: 5}          // AND (a = 5)  
[Op.or]: [{a: 5}, {a: 6}] // (a = 5 OR a = 6)  
[Op.gt]: 6,                // > 6  
[Op.gte]: 6,               // >= 6  
[Op.lt]: 10,               // < 10  
[Op.lte]: 10,              // <= 10  
[Op.ne]: 20,               // != 20  
[Op.eq]: 3,                // = 3  
[Op.not]: true,            // IS NOT TRUE  
[Op.between]: [6, 10],       // BETWEEN 6 AND 10  
[Op.notBetween]: [11, 15]    // NOT BETWEEN 11 AND 15
```

Utilização Básica do Sequelize



```
var Sequelize = require('sequelize')
, sequelize = new Sequelize('BDEquipamentos', 'usrsequelize', 'sequelize',
  {
    host: 'localhost',
    dialect: "mysql",
    port: 3306,
    operatorsAliases: false  });
sequelize.authenticate()
.then(function(err) { console.log('SUCESSO!!! Conexao estabelecida.');?>,
  function (err) { console.log('ERRO!!! Nao foi possivel conectar.', err); } );

var TipoEquipamento = sequelize.define('TipoEquipamento',
  { cod_tipo_equipamento: Sequelize.INTEGER,
    dsc_tipo_equipamento: Sequelize.STRING
  },
  {
    tableName: 'TipoEquipamento',
    paranoid: true,
    createdAt: 'dat_criacao',
    updatedAt: 'dat_ult_atualizacao',
    deletedAt: 'dat_exclusao'
  });
}

var tpequipamento = TipoEquipamento.findAll().
then (tpequipamento => {
  console.log(tpequipamento)
})
```

Utilização Básica do Sequelize



▪ Associações

- Recurso para as operações de relacionamento (join) no modelo relacional;
- Suporte a todas as cardinalidades existentes.

```
const Equipamento = this.sequelize.define('equipamento', {/* atributos */})
const TipoEquipamento = this.sequelize.define('tipoequipamento', {/* atributos */});

TipoEquipamento.hasMany(Equipamento);

Equipamento.belongsTo(TipoEquipamento);
```

Próxima Aula



- Capítulo 13 - Controle de Versão de Banco de Dados

A Linguagem SQL

CAPÍTULO 13. AULA 13.1. INTRODUÇÃO AO CONTROLE DE VERSÃO DE BANCO DE DADOS

PROF. GUSTAVO AGUILAR

Nesta Aula



- Controle de Versão de Banco de Dados
- Migrations
- Ferramentas de Mercado

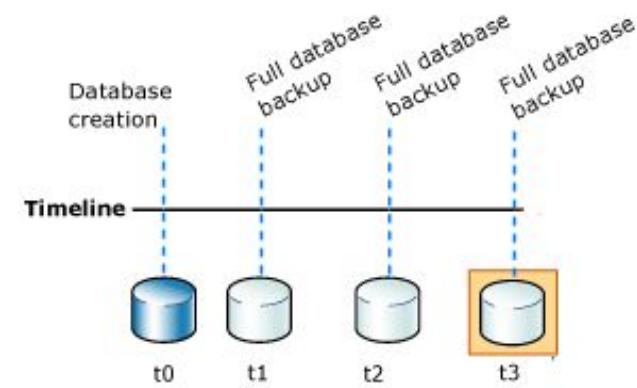
Controle de Versão de Banco de Dados

IGTI

- Antes do surgimento das ferramentas específicas para versionamento de banco de dados:

- **Versionamento full do banco**

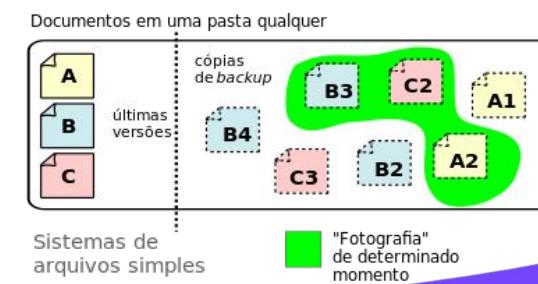
- Tamanho do banco (desperdício de espaço de armazenamento com as versões completas e não somente da alteração feita);
 - Dificuldade de replicar para outros ambientes somente a alteração feita;
 - Imprescindível intervenção humana: sujeito a erros operacionais (o que poderia invalidar o controle de versão).



Controle de Versão de Banco de Dados



- **Versionamento de cada alteração no banco de dados em um arquivo separado.**
 - Mais granular;
 - Otimizada em questões de espaço de armazenamento;
 - Mais fácil para replicação das alterações;
 - Necessidade de intervenção humana com perfil metódico;
 - Cada alteração, por menor que seja, deve ser versionada;
 - Complexidade para se gerir e coordenar muitas alterações;
 - Elevado número de arquivos / scripts para uma determinada versão.



Migrations

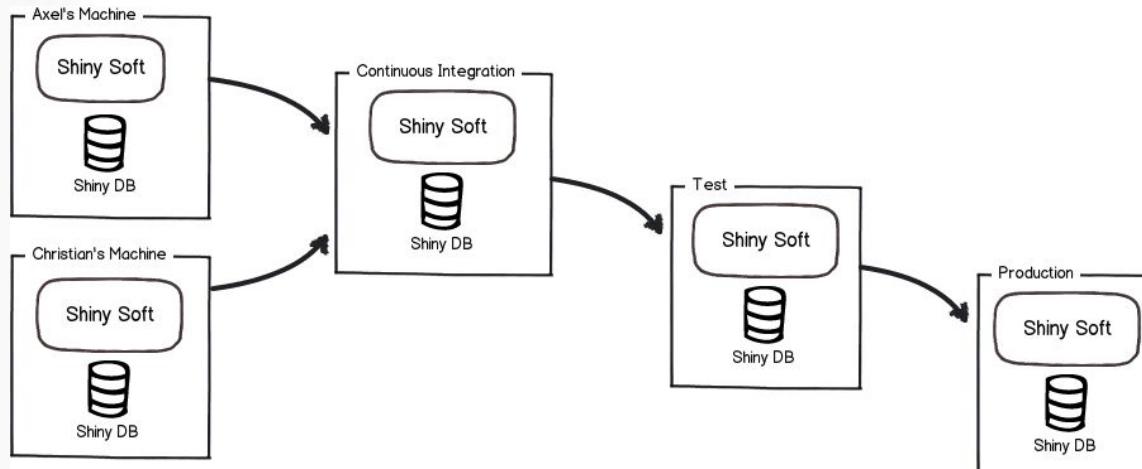


- Para eliminar os problemas e inconvenientes das formas de controle de versão de banco de dados que existiam;
- Atribuída por uma parte da comunidade ao framework Ruby on Rails e segundo outros, ao famoso Visual Studio da Microsoft e seu conhecido e poderoso mecanismo de controle de versão nos deploys;

Migrations

IGTI

- Pacote de alterações feitas no schema do banco de dados em um determinado momento:
 - Aplicar versão nova contida na migration (**subir versão**);
 - Desfazer as alterações (**voltar versão**).



Ferramentas de Mercado

IGTI



Visual Studio



redgate



Liquibase.org

versionSQL



Próxima Aula



- Version SQL

A Linguagem SQL

CAPÍTULO 13. AULA 13.2. VERSIONSQL

PROF. GUSTAVO AGUILAR

Nesta Aula

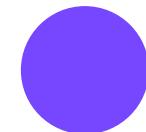


- ❑ Introdução ao VersionSQL
- ❑ Funcionamento Básico do VersionSQL

Introdução ao VersionSQL



- É um software de controle de versão de banco de dados **SQL Server**;
- Para plataforma **Windows**;
- Desenvolvido pela MV Webcraft;
- Repositório para armazenamento das versões dos bancos de dados:
 - Qualquer servidor Git ou Subversion hospedado em uma rede interna;
 - Nuvem □ GitHub, Atlassian Bitbucket, Visual Studio Team Services, etc.
- Após ser instalado, adiciona atalhos, no painel Object Explorer do SSMS, específicos para o fluxo de controle de versões
 - Check-in (Banco de Dados / Folder / Objeto)



Introdução ao VersionSQL



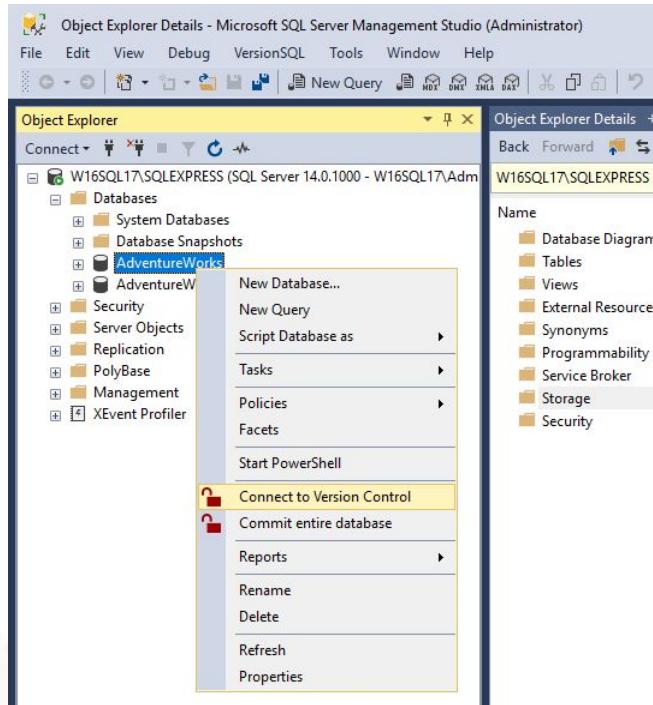
- Funcionamento:
 - Código T-SQL com as alterações do banco de dados é gravado em um arquivo, com a extensão **.sql** e organizado em pastas;
 - Enviado para o servidor de controle de versão.
- O VersionSQL Express é 100% gratuito □ SQL Server Express
- VersionSQL Professional
 - Suporte para outras edições do SQL;
 - Necessária a aquisição da licença, que é por usuário.

Introdução ao VersionSQL



- É preciso adicionar cada banco de dados ao VersionSQL;
- Botão direito sobre eles no Management Studio;

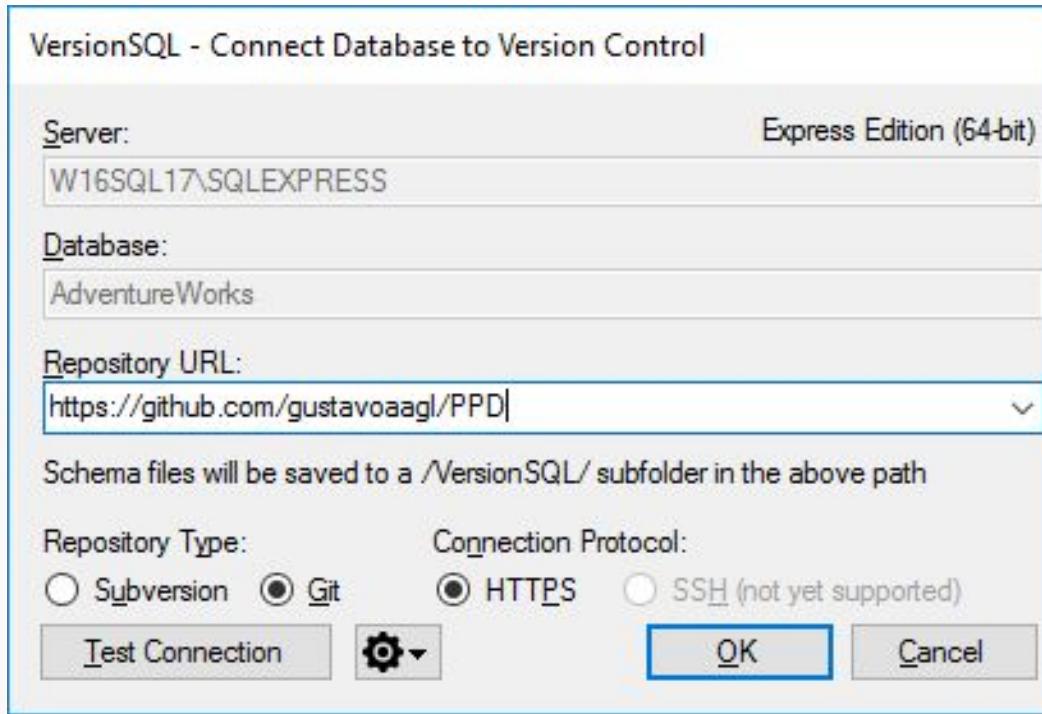
- ***Connect to Version Control***



Introdução ao VersionSQL

IGTI

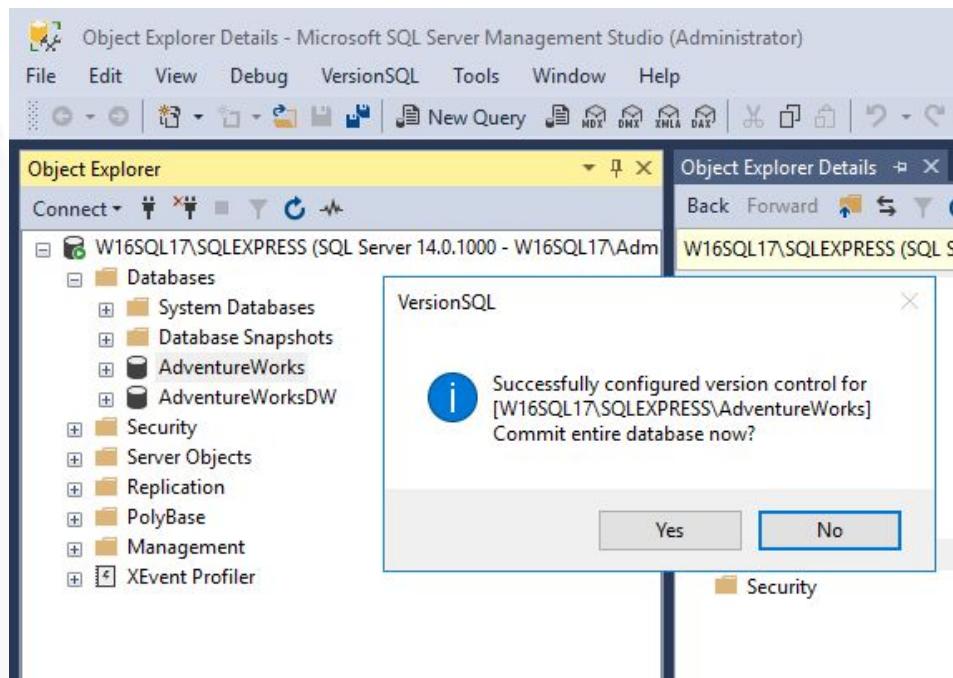
- Repositório



Introdução ao VersionSQL

IGTI

- Após configuração □ primeira sincronização full (**commit entire database**)
 - Criação da primeira versão do banco de dados.



Introdução ao VersionSQL



- Organização do repositório no GitHub:

A screenshot of a GitHub repository page for 'gustavoagl / PPD'. The repository structure is shown as 'PPD / VersionSQL / W16SQL17 / SQLEXPRESS / AdventureWorks / HumanResources / Table /'. The 'Code' tab is selected. The commit history shows the following files and their first versions:

File	Type	Message	Time
Department.sql	SQL	Primeira versão	17 minutes ago
Employee.sql	SQL	Primeira versão	17 minutes ago
EmployeeDepartmentHistory.sql	SQL	Primeira versão	17 minutes ago

Introdução ao VersionSQL



- GitHub Desktop;
- Navegar na estrutura, abrir scripts e copiar scripts.

The screenshot shows the GitHub Desktop application interface. On the left, there's a sidebar with a tree view of the repository structure:

- GitHub
- PPD
- VersionSQL
- W16SQL17
 - SQLEXPRESS
 - AdventureWorks
 - dbo
 - HumanResources
 - Procedure
 - Table
 - Department
 - View
 - Person
 - Production
 - Purchasing
 - Sales
 - AdventureWorksDW
 - jdbc
 - Lexicon
 - lib
 - MFGAssure

The main area displays a file list for the 'Table' folder:

Name	Date modified	Type	Size
Department.sql	20/04/2019 20:45	SQL File	2 KB
Employee.sql	20/04/2019 20:45	SQL File	6 KB
EmployeeDepartmentHistory.sql	20/04/2019 20:45	SQL File	3 KB
EmployeePayHistory.sql	20/04/2019 20:45	SQL File	2 KB

A specific file, 'Department.sql', is open in a Notepad window. The code content is as follows:

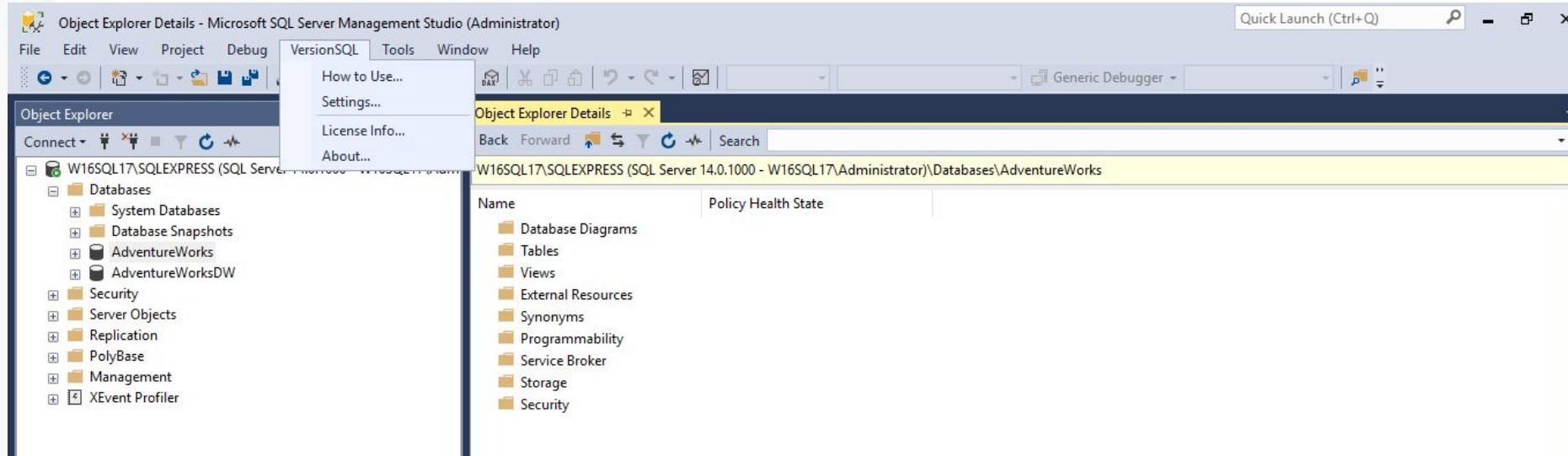
```
***** Object: Table [HumanResources].[Department] Committed by VersionSQL https://www.versionsql.com *****

SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
CREATE TABLE [HumanResources].[Department](
    [DepartmentID] [smallint] IDENTITY(1,1) NOT NULL,
    [Name] [dbo].[Name] NOT NULL,
    [GroupName] [dbo].[Name] NOT NULL,
    [ModifiedDate] [datetime] NOT NULL,
    CONSTRAINT [PK_Department_DepartmentID] PRIMARY KEY CLUSTERED
(
    [DepartmentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]

SET ANSI_PADDING ON
```

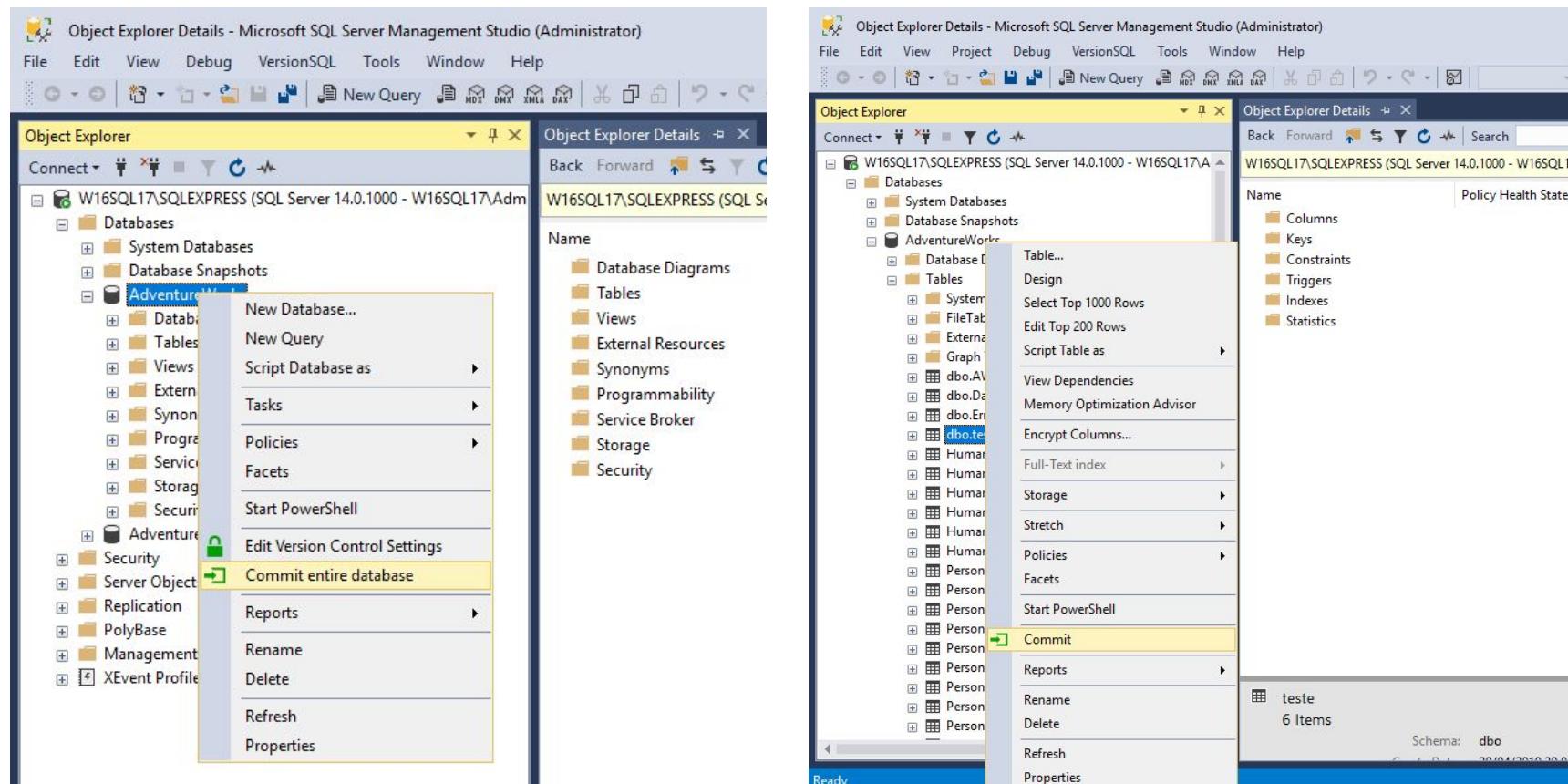
Funcionamento Básico do VersionSQL

IGTI



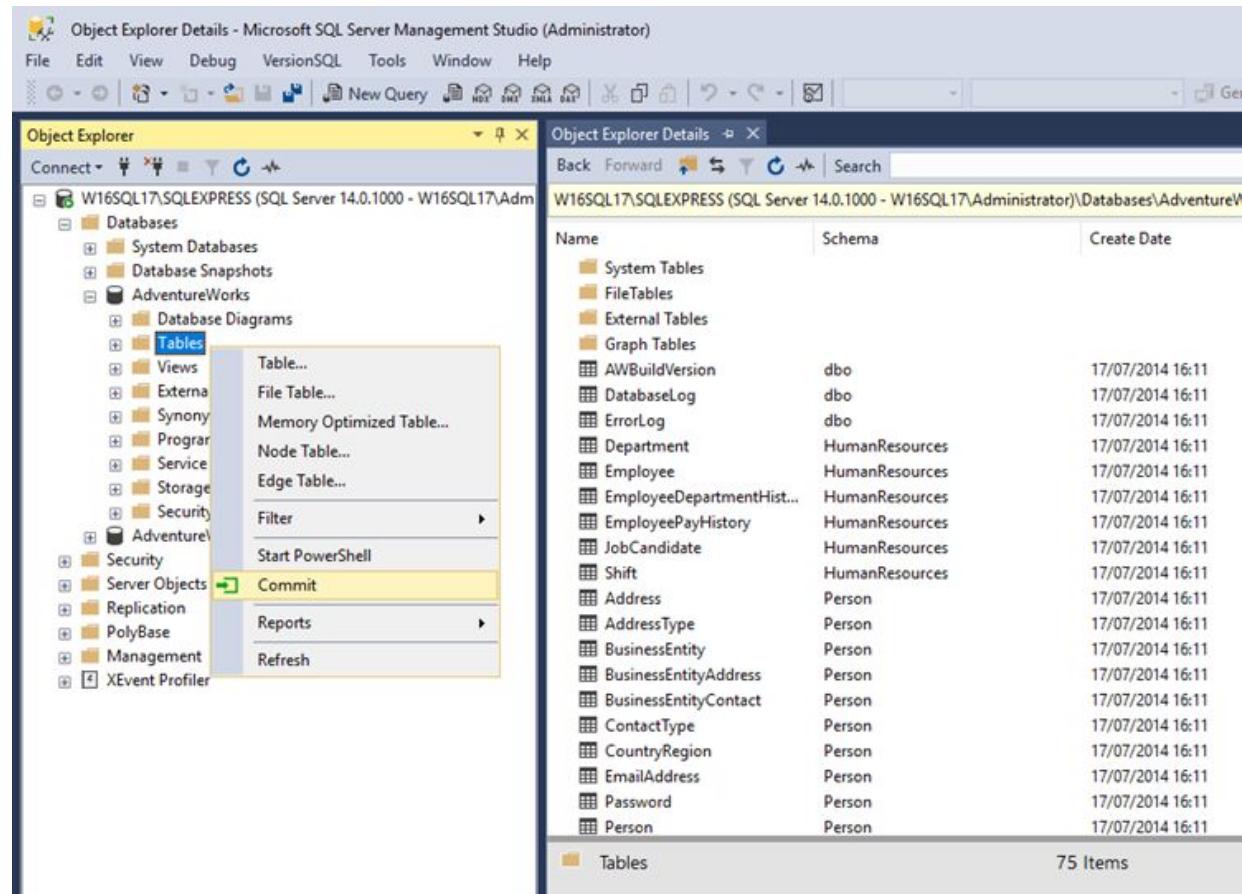
Funcionamento Básico do VersionSQL

IGTI



Funcionamento Básico do VersionSQL

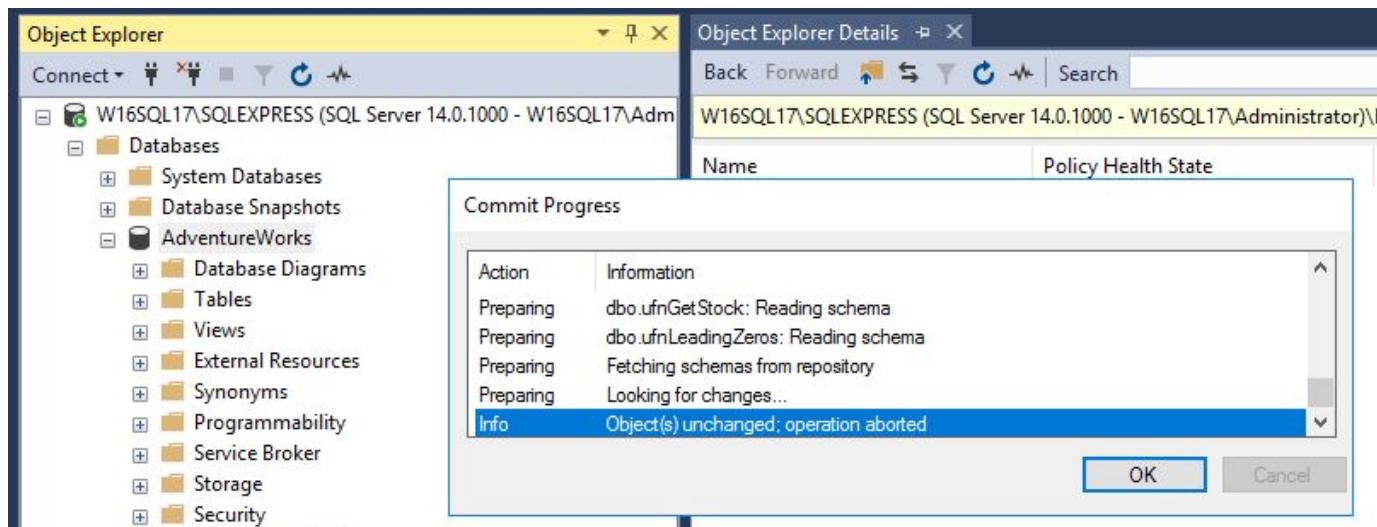
IGTI



Funcionamento Básico do VersionSQL

IGTI

- **MIGRATIONS:** usadas nas opções de commit do VersionSQL;
- Permite verificar que nenhuma alteração foi feita após o último check-in:
 - Aborta a operação □ economiza de espaço, menos poluição e mais controle.



Próxima Aula



- Flyway

A Linguagem SQL

CAPÍTULO 13. AULA 13.3. FLYWAY

PROF. GUSTAVO AGUILAR

Nesta Aula



- Introdução ao Flyway
- Funcionamento Básico do Flyway

Introdução ao Flyway

IGTI

- Software de controle de versão desenvolvido para as plataformas Windows, Linux, macOS e Docker;
- Suporte à diversos SGBDs:
 - Oracle, SQL Server, DB2, MySQL, MariaDB
 - CockroachDB, PostgreSQL, SQLite
 - Amazon Redshift
 - SAP HANA e Sybase ASE;
- Repositório interno para controle das versões, armazenado em *flat file*.



Introdução ao Flyway

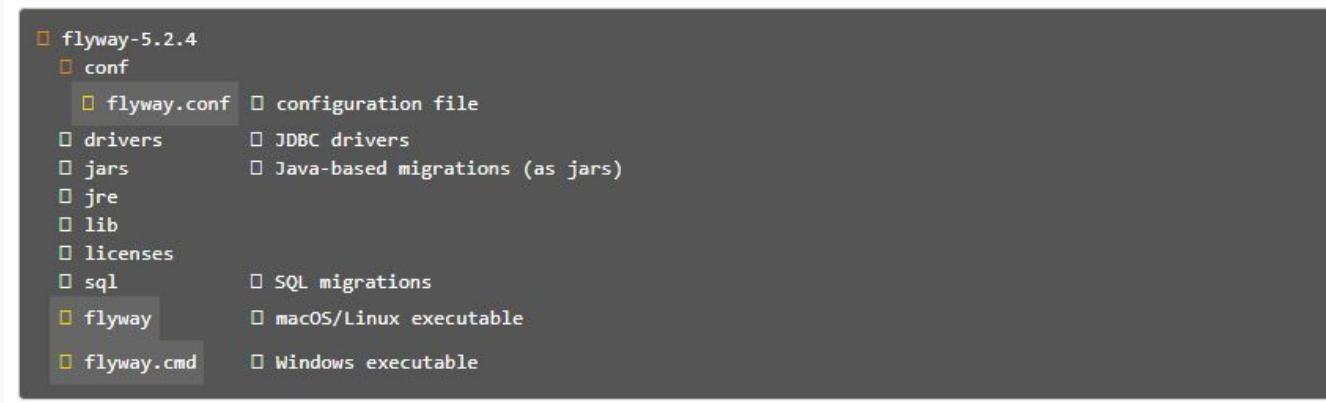


- Não possui atalhos nas interfaces gráficas como o VersionSQL;
- Controle de versão e migração de alterações em dados (scripts DML) e não somente DDL;
- Usa migrations para exportar a alteração (versão) para o banco de dados:
 - O próprio Flyway é quem executa e efetiva a alteração no schema físico do banco de dados.
- Edição gratuita □ apenas uma ferramenta de linha de comando;
- Edições pagas (Pro e Enterprise) □ recursos e APIs adicionais.

Funcionamento Básico do Flyway



- O Flyway funciona através de ferramenta de linha de comando;
- Baixada no endereço <https://flywaydb.org/download/>;
- Não possui um instalador □ copia para local desejado e descompacta.



Funcionamento Básico do Flyway



- URL do local do banco de dados + usuário e senha para acesso;
- Feita no arquivo **/conf/flyway.conf**.
- Exemplo com SQL Server:

```
C:\>cd "Program Files"
C:\Program Files>cd flyway-5.2.4
C:\Program Files\flyway-5.2.4>cd conf
C:\Program Files\flyway-5.2.4\conf>dir
Volume in drive C has no label.
Volume Serial Number is 6A05-C5D3

Directory of C:\Program Files\flyway-5.2.4\conf

21/04/2019  14:01    <DIR>      .
21/04/2019  14:01    <DIR>      ..
21/04/2019  14:00           16.542 flyway.conf
               1 File(s)     16.542 bytes
               2 Dir(s)   25.889.562.624 bytes free

C:\Program Files\flyway-5.2.4\conf>flyway.conf
```

flyway - Notepad

```
File Edit Format View Help
flyway.url=jdbc:sqlserver://W16SQL17:1440;databaseName=AdventureWorks2
flyway.user= user01
flyway.password= user01
```

Funcionamento Básico do Flyway



- **MIGRATIONS** □ arquivo (script) .sql;
- Criadas no diretório **/sql** do caminho de instalação do Flyway;

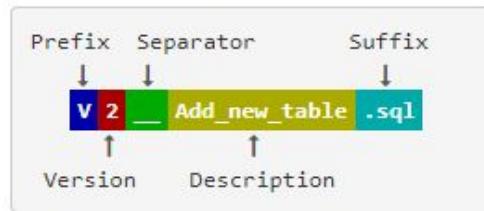
The screenshot shows a Windows File Explorer window with the following path: This PC > Local Disk (C:) > Program Files > flyway-5.2.4 > sql. The 'sql' folder contains two files: 'put-your-sql-migrations-here.txt' (Text Document, 0 KB) and 'V1_Create_person_table.sql' (Microsoft SQL Server script, 1 KB). A black arrow points from the bottom left towards the 'V1_Create_person_table.sql' file, which is highlighted with a light blue background. A callout box with a white background and a thin gray border is positioned over the arrow, displaying the SQL code for creating a table:

```
CREATE TABLE PESSOA
(
    CODIGO int not null,
    NOME varchar (200) not null
);
```

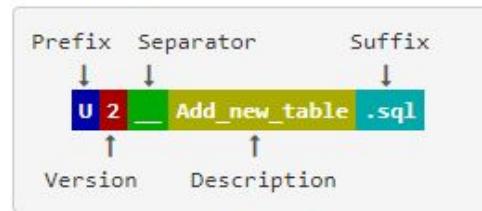
Funcionamento Básico do Flyway



Versioned Migrations



Undo Migrations



Repeatable Migrations



- **Prefixo:** V para migrations versionadas, U para migrations de undo e R para repeatable migrations;
- **Versão:** número sequencial para controlar a versão;
- **Separador:** dois underscores “__”;
- **Descrição:** underscore ou espaço separando as palavras;
- **Sufixo:** .sql.

Funcionamento Básico do Flyway



- Primeira migration □ Ex.: arquivo **V1__Cria_Tabela_Pessoa.sql**;
- Comando ***flyway migrate***;

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the output of a Flyway migration command. The text in the window is as follows:

```
C:\Program Files\flyway-5.2.4>flyway migrate
Flyway Community Edition 5.2.4 by Boxfuse
Database: jdbc:sqlserver://W16SQL17:1440;useBulkCopyForBatchInsert=false;cancelQueryTimeout=-1;sslProtocol=TLS;jaasConfigurationName=SQLJDBCDriver;statementPoolingCacheSize=0;serverPreparedStatementDiscardThreshold=10;enablePrepareOnFirstPreparedStatementCall=false;fips=false;socketTimeout=0;authentication=NotSpecified;authenticationScheme=nativeAuthentication;xopenStates=false;sendTimeAsDatetime=true;trustStoreType=JKS;trustServerCertificate=false;TransparentNetworkIPResolution=true;serverNameAsACE=false;sendStringParametersAsUnicode=true;selectMethod=direct;responseBuffering=adaptive;queryTimeout=-1;packetSize=8000;multiSubnetFailover=false;loginTimeout=15;lockTimeout=-1;lastUpdateCount=true;encrypt=false;disableStatementPooling=true;databaseName=AdventureWorks2;columnEncryptionSetting=Disabled;applicationName=Microsoft JDBC Driver for SQL Server;applicationIntent=readwrite; (Microsoft SQL Server 14.0)
Successfully validated 1 migration (execution time 00:00.071s)
Creating Schema History table: [AdventureWorks2].[dbo].[flyway_schema_history]
Current version of schema [dbo]: << Empty Schema >>
Migrating schema [dbo] to version 1 - Cria Tabela Pessoa
Successfully applied 1 migration to schema [dbo] (execution time 00:00.131s)

C:\Program Files\flyway-5.2.4>
```

Funcionamento Básico do Flyway



- Na primeira migration → criação também da tabela para controle do versionamento, chamada ***flyway_schema_history***;

A screenshot of the Microsoft SQL Server Management Studio (SSMS) Object Explorer. The title bar says "Object Explorer Details". The main pane shows a table named "flyway_schema_history" under the "Tables" node of the "AdventureWorks2" database. The table has three columns: "Name", "Schema", and "Create Date". There are two rows: one for the system table "flyway_schema_history" (schema dbo, created 2019-04-21 16:24) and one for the user table "PESSOA" (schema dbo, created 2019-04-21 16:24).

Name	Schema	Create Date
System Tables		
FileTables		
External Tables		
Graph Tables		
flyway_schema_history	dbo	21/04/2019 16:24
PESSOA	dbo	21/04/2019 16:24

Funcionamento Básico do Flyway



- Verificar informação das migrations já implantadas e versionada:
 - Comando ***flyway info***.

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the output of the "flyway info" command. The output includes the Flyway version (5.2.4), the database connection details (jdbc:sqlserver://W16SQL17:1440), and the schema version (1). A table is displayed showing one migration entry:

Category	Version	Description	Type	Installed On	State
Versioned	1	Cria Tabela Pessoa	SQL	2019-04-21 16:24:51	Success

```
C:\Program Files\flyway-5.2.4>flyway info
Flyway Community Edition 5.2.4 by Boxfuse
Database: jdbc:sqlserver://W16SQL17:1440;useBulkCopyForBatchInsert=false;cancelQueryTimeout=-1;sslProtocol=TLS;jaasConfigurationName=SQLJDBCDriver;statementPoolingCacheSize=0;serverPreparedStatementDiscardThreshold=10;enablePrepareOnFirstPreparedStatementCall=false;fips=false;socketTimeout=0;authentication=NotSpecified;authenticationScheme=nativeAuthentication;xopenStates=false;sendTimeAsDatetime=true;trustStoreType=JKS;trustServerCertificate=false;TransparentNetworkIPResolution=true;serverNameAsACE=false;sendStringParametersAsUnicode=true;selectMethod=direct;responseBuffering=adaptive;queryTimeout=-1;packetSize=8000;multiSubnetFailover=false;loginTimeout=15;lockTimeout=-1;lastUpdateCount=true;encrypt=false;disableStatementPooling=true;databaseName=AdventureWorks2;columnEncryptionSetting=Disabled;applicationName=Microsoft JDBC Driver for SQL Server;applicationIntent=readwrite; (Microsoft SQL Server 14.0)
Schema version: 1

+-----+-----+-----+-----+-----+
| Category | Version | Description      | Type | Installed On   | State  |
+-----+-----+-----+-----+-----+
| Versioned | 1       | Cria Tabela Pessoa | SQL  | 2019-04-21 16:24:51 | Success |
+-----+-----+-----+-----+-----+



C:\Program Files\flyway-5.2.4>
```

Funcionamento Básico do Flyway



- Flyway permite que também sejam versionados scripts DML;
- Ex.: criar uma segunda migration, de nome **V2__Insere_Pessoas.sql**:
 - INSERT INTO PESSOA (CODIGO, NOME) values (1, 'João');
 - INSERT INTO PESSOA (CODIGO, NOME) values (2, 'Maria');

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the output of a Flyway migration command. The text in the window reads:

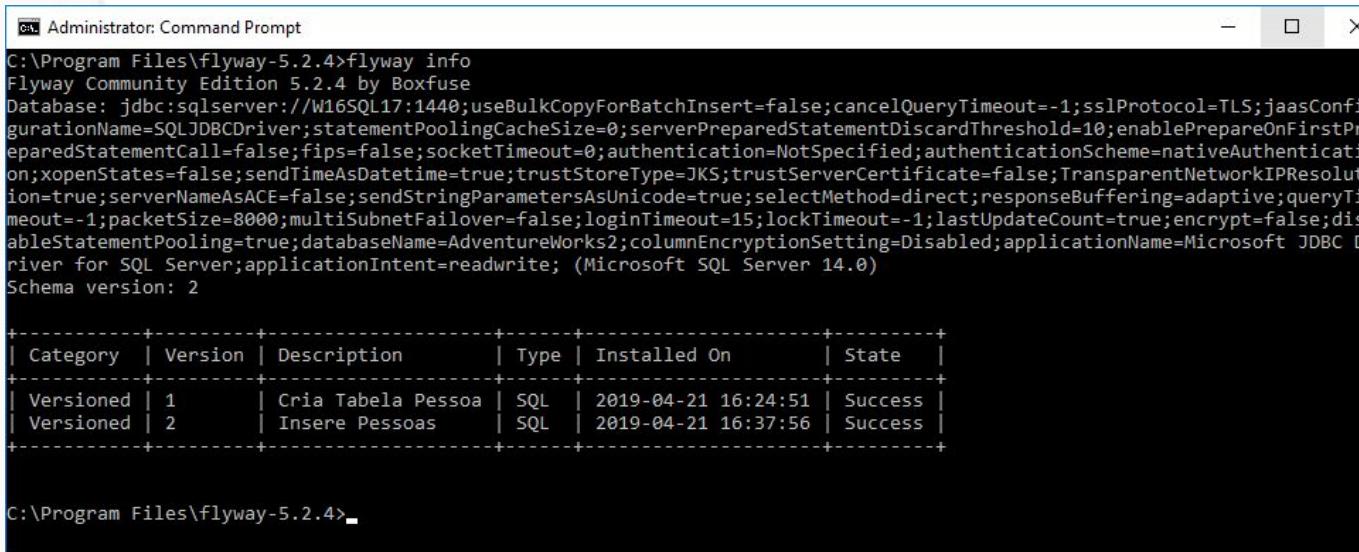
```
C:\Program Files\flyway-5.2.4>flyway migrate
Flyway Community Edition 5.2.4 by Boxfuse
Database: jdbc:sqlserver://W16SQL17:1440;useBulkCopyForBatchInsert=false;cancelQueryTimeout=-1;sslProtocol=TLS;jaasConfigurationName=SQLJDBCDriver;statementPoolingCacheSize=0;serverPreparedStatementDiscardThreshold=10;enablePrepareOnFirstPreparedStatementCall=false;fips=false;socketTimeout=0;authentication=NotSpecified;authenticationScheme=nativeAuthentication;xopenStates=false;sendTimeAsDatetime=true;trustStoreType=JKS;trustServerCertificate=false;TransparentNetworkIPResolution=true;serverNameAsACE=false;sendStringParametersAsUnicode=true;selectMethod=direct;responseBuffering=adaptive;queryTimeout=-1;packetSize=8000;multiSubnetFailover=false;loginTimeout=15;lockTimeout=-1;lastUpdateCount=true;encrypt=false;disallowStatementPooling=true;databaseName=AdventureWorks2;columnEncryptionSetting=Disabled;applicationName=Microsoft JDBC Driver for SQL Server;applicationIntent=readwrite; (Microsoft SQL Server 14.0)
Successfully validated 2 migrations (execution time 00:00.121s)
Current version of schema [dbo]: 1
Migrating schema [dbo] to version 2 - Insere_Pessoas
Successfully applied 1 migration to schema [dbo] (execution time 00:00.081s)

C:\Program Files\flyway-5.2.4>
```

Funcionamento Básico do Flyway



- Controle de versões implantadas é atualizado imediatamente:



```
C:\Program Files\flyway-5.2.4>flyway info
Flyway Community Edition 5.2.4 by Boxfuse
Database: jdbc:sqlserver://W16SQL17:1440;useBulkCopyForBatchInsert=false;cancelQueryTimeout=-1;sslProtocol=TLS;jaasConfigurationName=SQLJDBCDriver;statementPoolingCacheSize=0;serverPreparedStatementDiscardThreshold=10;enablePrepareOnFirstPreparedStatementCall=false;fips=false;socketTimeout=0;authentication=NotSpecified;authenticationScheme=nativeAuthentication;xopenStates=false;sendTimeAsDatetime=true;trustStoreType=JKS;trustServerCertificate=false;TransparentNetworkIPResolution=true;serverNameAsACE=false;sendStringParametersAsUnicode=true;selectMethod=direct;responseBuffering=adaptive;queryTimeout=-1;packetSize=8000;multiSubnetFailover=false;loginTimeout=15;lockTimeout=-1;lastUpdateCount=true;encrypt=false;disableStatementPooling=true;databaseName=AdventureWorks2;columnEncryptionSetting=Disabled;applicationName=Microsoft JDBC Driver for SQL Server;applicationIntent=readwrite; (Microsoft SQL Server 14.0)
Schema version: 2

+-----+-----+-----+-----+-----+
| Category | Version | Description      | Type | Installed On   | State  |
+-----+-----+-----+-----+-----+
| Versioned | 1       | Cria Tabela Pessoa | SQL  | 2019-04-21 16:24:51 | Success |
| Versioned | 2       | Insere Pessoas    | SQL  | 2019-04-21 16:37:56 | Success |
+-----+-----+-----+-----+-----+
C:\Program Files\flyway-5.2.4>
```



A Linguagem SQL

FIM

PROF. GUSTAVO AGUILAR