

# Apresentação de Modelo Relacional |

## Linguagem de Manipulação de Dados (DML)



DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

# Considerações Iniciais

- Nosso horário é das **19:00 às 22:00 (10 min tolerância de chegada)**
- Intervalo às **20:30 de 20 minutos**
- Interaja na aula!
- Se tiver dúvidas, levanta a mão no chat e eu explico novamente
- Me corrija!!!!

Erros são comuns no mundo do Software

Eu posso errar. Corrija-me quando acontecer

Aceito feedbacks :D

- Façam as atividades práticas, só se aprende praticando!
- Qualquer coisa me procurem no **slack**



# SQL

## Filtro HAVING

- Especifica uma condição de pesquisa para um grupo
- HAVING pode ser usado apenas com a instrução SELECT
- HAVING é normalmente usado com uma cláusula GROUP BY

### Exemplo:

```
USE Faculdade
GO

SELECT
    SUM(Nota.Nota) 'Soma das Notas'
    , NotaPeriodo.Periodo
FROM Aluno
INNER JOIN Nota ON Aluno.Id = Nota.IdAluno
INNER JOIN NotaPeriodo ON NotaPeriodo.Id = Nota.IdNotaPeriodo
GROUP BY NotaPeriodo.Periodo
HAVING SUM(Nota.Nota) > 40
```

# SQL

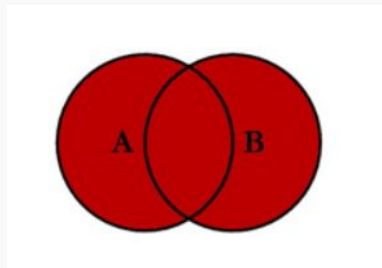
## Operador UNION

- O operador **UNION** combina os resultados de duas ou mais queries em um **único resultado**
- Para utilizar o **UNION**, o número e a ordem das colunas precisam ser idênticos em todas as queries e os tipos das colunas precisam ser compatíveis.

### Exemplo:

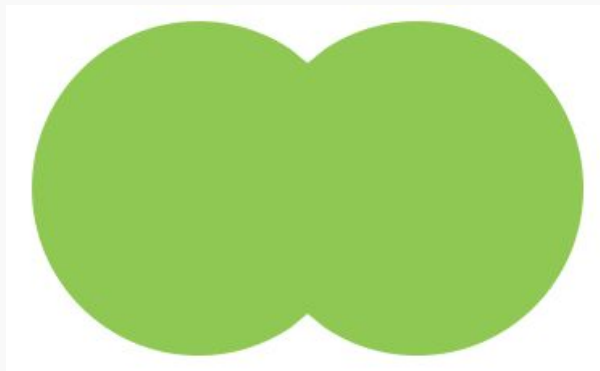
```
USE Faculdade
GO

SELECT
    Id
    , Nome
    , CPF
FROM Aluno
UNION
SELECT
    Id
    , Nome
    , Requisito
FROM Curso
```



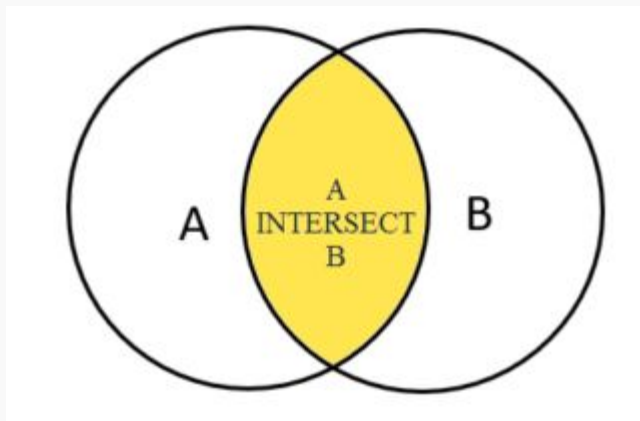
# União de conjuntos

- Conjunto A = (a, b, c, h, j)
- Conjunto B = (a, k, l, p, o)
- $A \cup B = (a, b, c, h, j, k, l, p, o)$



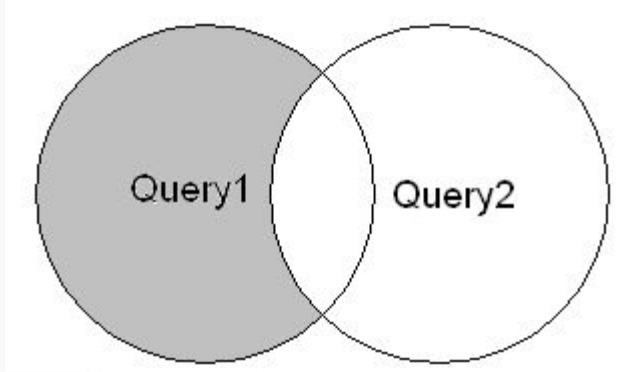
# Intersecção de conjuntos

- Conjunto A = (a, b, c, h, j)
- Conjunto B = (a, k, l, p, o)
- $A \cap B = (a)$



# Exceção/diferença de conjuntos

- Conjunto A = (a, b, c, h, j)
- Conjunto B = (a, k, l, p, o)
- $A - B = (b, c, h, j)$



# Conjuntos

- No SQL:
  - O número e a ordem das colunas na lista de seleção de ambas as consultas devem ser iguais.
  - Os tipos de dados devem ser compatíveis.
- ```
SELECT * FROM tabelaA
UNION
SELECT * FROM tabelaB
```
- ```
SELECT * FROM tabelaA
INTERSECT
SELECT * FROM tabelaB
```
- ```
SELECT * FROM tabelaA
EXCEPT
SELECT * FROM tabelaB
```



# SQL

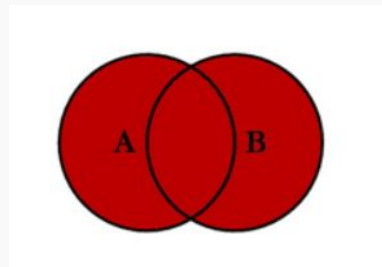
## Operador UNION ALL

- O operador **UNION ALL** tem a mesma funcionalidade do UNION, porém, não executa o SELECT DISTINCT no result set final e apresenta todas as linhas, inclusive as linhas duplicadas

**Exemplo:**

```
USE Faculdade  
GO
```

```
SELECT  
    Id  
    , Nome  
    , CPF  
FROM Aluno  
UNION ALL  
SELECT  
    Id  
    , Nome  
    , Requisito  
FROM Curso
```



# SQL

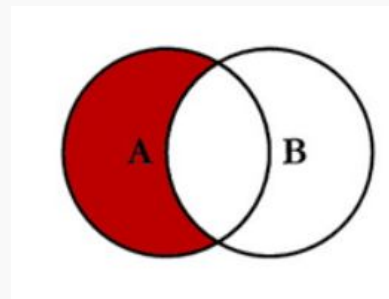
## Operador de Conjunto - EXCEPT

- O operador de conjunto **EXCEPT** retorna linhas distintas da consulta de entrada à esquerda que não são produzidas pela consulta de entrada à direita

**Exemplo:**

```
USE Livraria
GO

SELECT
    Id
FROM Livro
EXCEPT
SELECT
    IdLivro
FROM Livro_Autor
```



# SQL

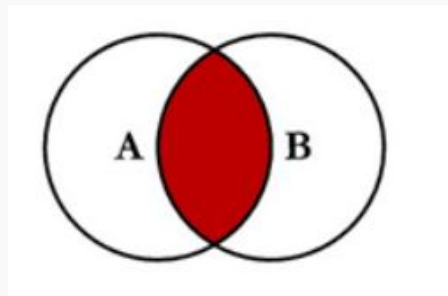
## Operador de Conjunto - INTERSECT

- O operador de conjunto **INTERSECT** retorna linhas distintas que são produzidas pelo operador das consultas de entrada à esquerda e à direita

**Exemplo:**

```
USE Livraria
GO

SELECT
    Id
FROM Livro
INTERSECT
SELECT
    IdLivro
FROM Livro_Autor
```



# SQL

- **DDL (Linguagem de definição de Dados)**
  - **TRUNCATE TABLE**
  - **COMMENT**



# SQL

## DDL - instrução TRUNCATE TABLE

- A instrução **TRUNCATE TABLE** é usada para remover todos os registros de uma tabela.
- Para demonstrar essa instrução, vamos criar uma tabela, inserir valores e depois realizar a instrução **TRUNCATE TABLE**



# SQL

## DDL - instrução COMMENT

- A instrução **COMMENT** é utilizada para explicar seções de instruções SQL ou para impedir a execução de instruções SQL

Vejam os exemplos a seguir:



A diagram showing a list of DDL (Data Definition Language) instructions. The title 'DDL' is in a yellow box at the top. Below it, the instructions are listed: CREATE, ALTER, DROP, TRUNCATE, COMMENT, and RENAME. Each instruction has a green checkmark next to it, except for COMMENT, which has a yellow checkmark.

| DDL      |   |
|----------|---|
| CREATE   | ✓ |
| ALTER    | ✓ |
| DROP     | ✓ |
| TRUNCATE | ✓ |
| COMMENT  | ✓ |
| RENAME   |   |

# SQL

## DDL - instrução COMMENT

- Instrução **COMMENT** => explicar seções de instruções SQL

```
-- Exibir dados distintos do autor
SELECT
    DISTINCT Autor.Id
    , Autor.Nome
FROM Autor
INNER JOIN Livro_Autor ON Autor.Id = Livro_Autor.IdAutor
```

# SQL

## DDL - instrução COMMENT

- Instrução **COMMENT** => impedir a execução de instruções SQL

```
SELECT
    Id
    ,Nome
    /*,Endereco
    ,Cidade
    ,Estado */
    ,EstadoCivil
FROM Pessoa;
```



# SQL

## DML - MERGE (no PostgreSQL v15)

- O comando **MERGE** executa operações de **inserção e atualização** em uma tabela de destino usando os resultados de uma união com uma tabela de origem

| DML          |   |
|--------------|---|
| SELECT       | ✓ |
| INSERT       | ✓ |
| UPDATE       | ✓ |
| DELETE       | ✓ |
| MERGE        | ✓ |
| CALL         |   |
| EXPLAIN PLAN |   |
| LOCK TABLE   |   |

# SQL

Vamos popular a tabela TargetTable usando a tabela SourceTable como fonte de dados

```
CREATE TABLE SourceTable
(
  Col1 INT NOT NULL PRIMARY KEY,
  Col2 VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE TargetTable
(
  Col1 INT NOT NULL PRIMARY KEY,
  Col2 VARCHAR(20) NOT NULL
);
```

```
INSERT INTO SourceTable (Col1, Col2)
VALUES
(2, 'Source2'),
(3, 'Source3'),
(4, 'Source4');
```

```
INSERT INTO TargetTable (Col1, Col2)
VALUES
(1, 'Target1'),
(2, 'Target2'),
(3, 'Target3');
```

# SQL

SELECT \* FROM SourceTable

Data Output

Messages

Notifications

≡+

📄

▼



📋

🗑️

🗄️

⬇️

📈

|   | col1<br>[PK] integer  | col2<br>character varying (20)  |
|---|--------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| 1 | 2                                                                                                      | Source2                                                                                                          |
| 2 | 3                                                                                                      | Source3                                                                                                          |
| 3 | 4                                                                                                      | Source4                                                                                                          |

SELECT \* FROM TargetTable

Data Output

Messages

Notifications

|   | col1<br>[PK] integer | col2<br>character varying (20) |
|---|----------------------|--------------------------------|
| 1 | 1                    | Target1                        |
| 2 | 2                    | Target2                        |
| 3 | 3                    | Target3                        |

# SQL

Exemplo do comando MERGE:

```
MERGE INTO TargetTable as TGT
USING SourceTable as SRC ON TGT.Col1 = SRC.Col1
WHEN MATCHED
    THEN UPDATE SET Col2 = SRC.Col2
WHEN NOT MATCHED
    THEN INSERT (Col1, Col2)
    VALUES (SRC.Col1, SRC.Col2);
```

# SQL

Após execução do MERGE ficou:

SELECT \* FROM SourceTable

| Data Output |                      |                                | Messages | Notifications |
|-------------|----------------------|--------------------------------|----------|---------------|
|             | col1<br>[PK] integer | col2<br>character varying (20) |          |               |
| 1           | 2                    | Source2                        |          |               |
| 2           | 3                    | Source3                        |          |               |
| 3           | 4                    | Source4                        |          |               |

SELECT \* FROM TargetTable

| Data Output |                      |                                | Messages | Notifications |
|-------------|----------------------|--------------------------------|----------|---------------|
|             | col1<br>[PK] integer | col2<br>character varying (20) |          |               |
| 1           | 1                    | Target1                        |          |               |
| 2           | 2                    | Source2                        |          |               |
| 3           | 3                    | Source3                        |          |               |
| 4           | 4                    | Source4                        |          |               |

# SQL

## Stored Procedures

- **Stored Procedure**, que traduzido significa Procedimento Armazenado, é um conjunto de comandos em **SQL** que podem ser executados de uma só vez, como em uma função. Ele armazena tarefas repetitivas e aceita parâmetros de entrada para que a tarefa seja efetuada de acordo com a necessidade individual

# SQL

## Stored Procedures

**Vejamos como criar uma procedure que retorna o nome do aluno e a sua nota por período:**

```
USE Faculdade  
GO
```

```
CREATE PROCEDURE Busca  
@NomeAluno VARCHAR(150)  
AS
```

```
    SELECT  
        Aluno.Nome [Nome Aluno]  
        , NotaPeriodo.Periodo  
        , Nota.Nota  
    FROM Aluno  
    INNER JOIN Nota ON Aluno.Id = Nota.IdAluno  
    INNER JOIN NotaPeriodo ON NotaPeriodo.Id =  
        Nota.IdNotaPeriodo  
    WHERE Aluno.Nome = @NomeAluno;
```

# SQL

## Stored Procedures

**Exemplo de como executar uma Stored Procedure:**

```
USE Faculdade  
GO
```

```
EXEC Busca 'Aluno 1';
```



# SQL

## Views

- A **view** pode ser definida como uma tabela virtual composta por linhas e colunas de dados vindos de tabelas relacionadas em uma query (um agrupamento de SELECT's, por exemplo). As linhas e colunas da view são geradas dinamicamente no momento em que é feita uma referência a ela

# SQL

## View

**Vejamos como criar uma View:**

```
USE Livraria  
GO
```

```
CREATE VIEW vwLivroAutor AS  
SELECT  
    Livro.Nome as 'Nome Livro'  
    , Livro.NumPagina  
    , Editora.Nome as 'Nome Editora'  
    , Autor.Nome  
FROM Livro  
INNER JOIN Editora ON Editora.Id = Livro.IdEditora  
INNER JOIN Livro_Autor ON Livro.Id = Livro_Autor.IdLivro  
INNER JOIN Autor ON Autor.Id = Livro_Autor.IdAutor
```

# SQL

## View

**Vejamos como utilizar uma View:**

```
USE Livraria  
GO
```

```
SELECT * from vwLivroAutor
```

# SQL

## View

**Vejamos como utilizar uma View:**

```
USE Livraria  
GO
```

```
SELECT * from vwLivroAutor
```



# DEVinHouse

Parcerias para desenvolver a sua carreira

**OBRIGADO!**



<LAB365>