



PROGRAMAÇÃO SQL

DDL | DML | DQL

Alexandre Fernandes

OBJECTIVOS

- SGBD
- SQL
- DDL
- DML
- DQL

OBJECTIVOS

- SGBD
- SQL
- DDL
- DML
- DQL

DQL

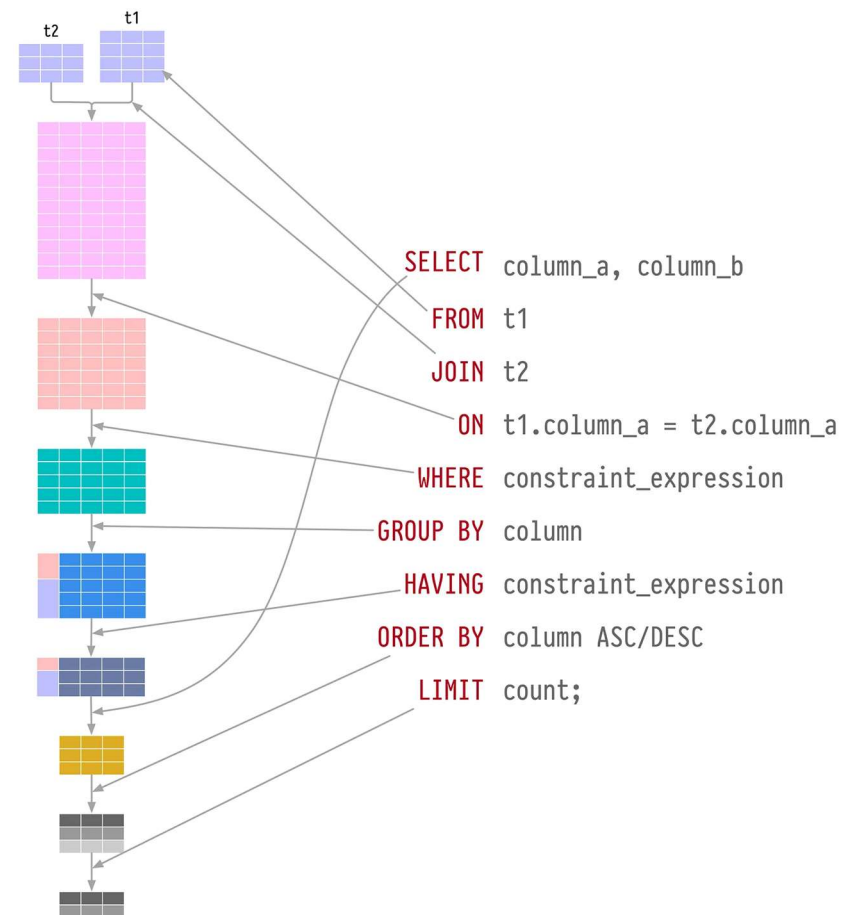
- DQL ou Data Query Language (Pesquisa de dados) são os comandos usados para **consultar** e **analisar** os dados armazenados:
 - **SELECT:** Comando principal para recuperar dados de uma ou mais tabelas
- Devem apenas selecionar as colunas necessárias para melhorar o desempenho da consulta.
- Os dados apresentados são filtrados através de condições.

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

A ordem das cláusulas deve ser sempre garantida

ESTRUTURA DQL



ESTRUTURA DQL

SELECT <lista_atributos>

FROM <tabela>

WHERE <condições>

GROUP BY <agrupamento_dados>

HAVING <condições_agrupamento>

ORDER BY <ordenação>

LIMIT <quantidade_registos>

Fundamental (Obrigatório)

Facultativo

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```


SELECT|FROM

- No SELECT são indicados os atributos a serem consultados na pesquisa.
- No FROM é indicada a tabela onde esses atributos estão presentes.

```
SELECT <lista_atributos>  
FROM <tabela>
```

EXEMPLO

- Consultar todos os atributos de uma tabela

```
SELECT *  
FROM Aluno
```

O asterisco (*) pode ser usado para indicar que pretendemos todos os atributos das tabelas que incluirmos na nossa consulta

EXEMPLO

- Consultar atributos específicos de uma tabela

```
SELECT Nome, Morada, DataInscricao  
FROM Aluno
```

ALIASES

- São **apelidos temporários** que podemos dar a tabelas ou colunas para facilitar a leitura, compreensão ou organização de uma consulta.
- **Não alteram o nome original das tabelas ou colunas** de uma base de dados, mas tornam a escrita de consultas mais concisa e legível.
- Definimos os aliases através da palavra *AS*, **apesar de ser opcional** em muitos dos SGBDs (incluído MySQL), mas frequentemente usada para maior clareza.

ALIASES

- Os ALIASES podem ser aplicados em tabelas e atributos

```
SELECT <atributo> AS <AliasAtributo>  
FROM <tabela> AS <AliasTabela>
```

EXEMPLO

- Apelidar atributos específicos de uma tabela

```
SELECT Nome AS 'Nome do Aluno', Morada, DataInscricao  
FROM Aluno
```

EXEMPLO

- Apelidar tabelas específicas

```
SELECT a.Nome, a.Morada, a.DataInscricao  
FROM Aluno AS a
```

Esta aplicação de apelidos torna-se substancialmente relevante quando trabalhamos com múltiplas tabelas

EXEMPLO

- Apelidar várias tabelas específicas

Sem Aliases

```
SELECT Aluno.Nome, Aluno.Morada, Professor.Email  
FROM Aluno, Professor
```

Com Aliases

```
SELECT a.Nome, a.Morada, a.Email  
FROM Aluno AS a, Professor AS p
```


DISTINCT

- É utilizada para **remover linhas duplicadas** dos resultados de uma consulta. **Garante que os valores** que aparecem nas colunas de um SELECT **são únicos no conjunto de resultados**.

```
SELECT DISTINCT <lista_atributos>  
FROM <tabela>
```

EXEMPLO

- Filtrar e apresentar apenas as linhas distintas

```
SELECT DISTINCT Nome, Localidade, Idade  
FROM Aluno
```

OPERADORES ARITMÉTICOS

- Os operadores aritméticos permitem **realizar cálculos em valores numéricos diretamente nas consultas**, seja para gerar novos resultados ou alterar a forma como os dados são exibidos.

OPERADOR	DESCRIÇÃO	EXEMPLO
+	Soma	Preço + 10
-	Subtração	Preço – Desconto
*	Multiplicação	Quantidade * Preço
/	Divisão	Total / Quantidade
%	Módulo (Resto da Divisão)	Valor % 2

EXEMPLO

- Realizar um calculo de um atributo com um valor

```
SELECT Descricao, Preco, Preco * 1.23  
FROM Produto
```

EXEMPLO

- Realizar um calculo de um atributo com um valor com ALIASES

```
SELECT Descricao, Preco, Preco * 1.23 AS 'Preço c/ IVA'  
FROM Produto
```

EXEMPLO

- Realizar multiplos calculo de um atributo com um valor

```
SELECT Descricao, Preco - (Preco * 0.10) AS 'Preco c/ Desc'  
FROM Produto
```

EXEMPLO

- Realizar calculos entre vários atributos

```
SELECT Descricao, Preco * Quantidade AS 'Valor Total'  
FROM Produto
```

FUNÇÕES

- São ferramentas poderosas que permitem **manipular, transformar e calcular dados diretamente nas consultas.**

```
SELECT CONCAT('O ', Nome, ' tem ', Idade, ' anos.')
```

```
FROM Aluno
```

```
SELECT YEAR(CURRENT_DATE()) - YEAR(DataInscricao)
```

```
FROM Aluno
```


FUNÇÕES PARA TEXTO

OPERADOR	DESCRIÇÃO	EXEMPLO
CONCAT	Combina dois ou mais valores numa única string.	CONCAT('Olá, ', Nome) → Olá, João
LENGTH	Retorna o tamanho (número de caracteres) de um texto.	LENGTH('MySQL') → 5
UPPER	Converte o texto para maiúsculas.	UPPER('mysql') → MYSQL
LOWER	Converte o texto para minúsculas.	LOWER('MySQL') → mysql
SUBSTRING	Extraí uma parte de um texto.	SUBSTRING('MySQL', 1, 2) → My
TRIM	Remove espaços extras no início e no fim do texto.	TRIM(' teste ') → teste

Para explorar mais funções do MySQL e obter suporte detalhado sobre como utilizá-las, consulte o link:

https://www.w3schools.com/mysql/mysql_ref_functions.asp

FUNÇÕES PARA NÚMEROS

OPERADOR	DESCRIÇÃO	EXEMPLO
ABS	Retorna o valor absoluto de um número.	ABS(-10) → 10
ROUND	Arredonda para o número específico de casas decimais.	ROUND(3.14159, 2) → 3.14
CEIL	Arredonda o número para cima (maior inteiro).	CEIL(3.2) → 4
FLOOR	Arredonda o número para baixo (menor inteiro).	FLOOR(3.8) → 3
MOD	Retorna o resto da divisão entre dois números.	MOD(10, 3) → 1
POWER	Eleva um número a uma potência especificada.	POWER(2, 3) → 8

Para explorar mais funções do MySQL e obter suporte detalhado sobre como utilizá-las, consulte o link:

https://www.w3schools.com/mysql/mysql_ref_functions.asp

FUNÇÕES PARA DATAS

OPERADOR	DESCRIÇÃO	EXEMPLO
CURRENT_DATE	Retorna a data atual do sistema.	CURRENT_DATE() → 2025-01-08
NOW	Retorna a data e hora atual do sistema.	NOW() → 2025-01-08 14:35:00
YEAR	Extraí o ano de uma data.	YEAR('2025-01-08') → 2025
MONTH	Extraí o mês de uma data.	MONTH('2025-01-08') → 1
DATEDIFF	Calcula a diferença entre duas datas.	DATEDIFF('2025-01-08', '2025-01-01') → 7
DATE_FORMAT	Formata uma data de maneira personalizada.	DATE_FORMAT('2025-01-08', '%d/%m/%Y') → 08/01/2025

Para explorar mais funções do MySQL e obter suporte detalhado sobre como utilizá-las, consulte o link:

https://www.w3schools.com/mysql/mysql_ref_functions.asp

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

WHERE

- A cláusula **WHERE** define as condições para filtrar os resultados retornados através do uso de vários operadores comuns como comparação, lógicos, intervalo ou inclusão.

SELECT <lista_atributos>

FROM <tabela>

WHERE <condições>

OPERADORES COMPARAÇÃO

- Os operadores de comparação **permitem comparar valores para filtrar ou selecionar dados com base em condições** e criar assim critérios de pesquisa para identificar registos que atendem às condições.

OPERADOR	DESCRIÇÃO	EXEMPLO
=	Igual (exatamente igual)	Preco = 100
<> ou !=	Diferente	Preco <> 100
>	Maior que	Preco > 100
<	Menor que	Preco < 100
>=	Maior ou igual que	Preco >= 100
<=	Menor ou igual que	Preco <= 100

EXEMPLO

- Filtrar os resultados com base numa condição

```
SELECT *  
FROM Aluno  
WHERE Nome = 'António'
```

EXEMPLO

- Filtrar os resultados com base numa condição

```
SELECT *  
FROM Aluno  
WHERE Idade <= 23
```


EXEMPLO

- Filtrar os resultados com base numa condição

```
SELECT Nome, Morada, CodPostal  
FROM Aluno  
WHERE DataInscricao > '2020-01-01'
```

OPERADORES LÓGICOS

- Os operadores lógicos **permitem combinar múltiplas condições**, e deste modo criar critérios de pesquisa mais complexos.

OPERADOR	DESCRIÇÃO	EXEMPLO
AND	E	Preço > 50 AND Quantidade < 100
OR	Ou	Preço > 100 OR Quantidade > 200
NOT	Não	NOT (Preço < 50)
IS [NOT] NULL	Valida se o valor está (ou não) vazio	Quantidade IS NULL

EXEMPLO

- Filtrar os resultados com base em múltiplas condições

```
SELECT *  
FROM Aluno  
WHERE Idade >= 18 AND Idade <= 23
```

EXEMPLO

- Filtrar os resultados com base em múltiplas condições

```
SELECT *  
FROM Aluno  
WHERE Nome = 'Alice' OR Localidade <> 'Porto'
```

EXEMPLO

- Filtrar os resultados com base em múltiplas condições

```
SELECT *  
FROM Aluno  
WHERE Nome = 'Alice' OR Nome = 'Alex' AND Idade >= 20
```

EXEMPLO

- Filtrar os resultados com base em múltiplas condições

```
SELECT *  
FROM Aluno  
WHERE Localidade IS NULL
```

EXEMPLO

- Filtrar os resultados com base em múltiplas condições

```
SELECT *  
FROM Aluno  
WHERE DataInscricao IS NOT NULL
```

OPERADORES DE INTERVALO

- Os operadores de intervalo **permitem verificar se um valor está dentro de um intervalo específico**, substituindo múltiplas condições de comparação.

OPERADOR	DESCRIÇÃO	EXEMPLO
BETWEEN ... AND	Verifica se o valor está dentro de um intervalo	Preco BETWEEN 50 AND 100
NOT BETWEEN ... AND	Verifica se o valor não está dentro do intervalo	Preco NOT BETWEEN 50 AND 100

EXEMPLO

- Filtrar os resultados com base num intervalo específico

```
SELECT Descricao, Stock  
FROM Produto  
WHERE Preco BETWEEN 5 AND 30
```

EXEMPLO

- Filtrar os resultados com base num intervalo específico

```
SELECT *  
FROM Aluno  
WHERE DataInscricao BETWEEN '2020-01-01' AND '2021-01-01'
```

OPERADORES DE INCLUSÃO

- Os operadores de inclusão **permitem verificar se um valor pertence a um conjunto específico de valores**, simplificando consultas que de outra forma exigiriam múltiplos *OR*.

OPERADOR	DESCRIÇÃO	EXEMPLO
IN	Verifica se o valor está numa lista específica	Categoria IN ('Jogos', 'Livros')
NOT IN	Verifica se o valor não está numa lista especificada	Categoria NOT IN ('Jogos', 'Livros')

EXEMPLO

- Filtrar os resultados com base num conjunto específico de valores

```
SELECT *  
FROM Aluno  
WHERE Localidade IN ('Porto', 'Coimbra', 'Braga')
```

OPERADORES DE CORRESPONDÊNCIA

Centro para o Desenvolvimento
de Competências Digitais

- Os operadores de correspondência de padrões **permitem realizar comparações de padrões em colunas do tipo texto**. Ambos podem ser usados para fazer pesquisas por padrões específicos dentro de strings.

OPERADOR	DESCRIÇÃO	EXEMPLO
LIKE	Para comparação de padrões simples em strings	Nome LIKE 'Jo%'
REGEXP	Para comparação de padrões mais complexas	Nome REGEXP '^Jo.*[0-9]'

EXEMPLO

- Filtrar os resultados com base em padrões específicos de texto

```
SELECT *  
FROM Aluno  
WHERE Nome LIKE 'A%'
```

```
SELECT *  
FROM Aluno  
WHERE Nome LIKE 'M_____'
```

LIKE

CARACTERE	DESCRIÇÃO	EXEMPLO
%	Corresponde a qualquer sequência de caracteres , inclusive uma string vazia.	'Jo%' encontra "Joana", "João", "José", etc.
_ (underscore)	Corresponde a um único carácter qualquer .	'J_n' encontra "Jan", "Jon", mas não "João".

EXEMPLO

- Filtrar os resultados com base em padrões específicos de texto

```
SELECT *  
FROM Aluno  
WHERE Nome REGEXP '^Jo'
```

```
SELECT *  
FROM Aluno  
WHERE Nome REGEXP '^Jo|ia$'
```

```
SELECT *  
FROM Aluno  
WHERE Nome REGEXP 'des$'
```

```
SELECT *  
FROM Aluno  
WHERE Nome REGEXP 'Mari[ao]'
```


REGEXP

CARACTERE	DESCRIÇÃO	EXEMPLO
^	Indica o início da string .	'^Jo' encontra "João", mas não "Antonio João".
\$	Indica o final da string .	'ão\$' encontra "João", mas não "Joãozinho".
.	Corresponde a qualquer carácter único .	'J..o' encontra "João" e "Julio", mas não "Jorge".
[]	Define um conjunto de caracteres .	'J[ao]ão' encontra "João" ou "Jaão".
[^]	Define os caracteres que não devem estar no conjunto .	'J[^a]ão' encontra "João", mas não "Jaão".
[-]	Define uma sequencia de caracteres ou números .	'Joã [a-p]' não encontra "Joãu". 'João[0-9]' encontra "João0" até "João9".
	Operador lógico <i>OR</i>	'João Alex Marta' encontra todos os "João", "Alex" ou "Marta"

EXEMPLO

```
SELECT *  
FROM Aluno  
WHERE  
    Nome LIKE 'Ana%' AND  
    Idade > 18 AND  
    DataInscricao BETWEEN '2023-01-01' AND '2023-12-31' AND  
    Localidade IN ('Lisboa', 'Porto', 'Coimbra') AND  
    (Idade+2) <= 30 OR  
    Nome LIKE '%Ana%';
```

EXEMPLO

SELECT *
FROM Aluno
WHERE

1ª Condição

```
Nome LIKE 'Ana%' AND  
Idade > 18 AND  
DataInscricao BETWEEN '2023-01-01' AND '2023-12-31' AND  
Localidade IN ('Lisboa', 'Porto', 'Coimbra') AND  
(Idade+2)<=30 OR
```

2ª Condição

```
Nome LIKE '%Ana%';
```

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

GROUP BY

- A cláusula GROUP BY é **usado para agrupar dados cujo valor seja igual num atributo específico**. Quando combinado com funções de agregação, é possível realizar cálculos nesses grupos.

```
SELECT <lista_atributos>  
FROM <tabela>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>
```

FUNÇÕES AGREGAÇÃO

FUNÇÃO	DESCRIÇÃO
COUNT(<atributo>)	Conta o numero de registos em cada grupo
SUM(<atributo>)	Soma o valor total dos registo de um atributo em cada grupo
AVG(<atributo>)	Média do valor total dos registo de um atributo em cada grupo
MIN(<atributo>)	Devolve o menor valor de menor de um atributo em cada grupo
MAX(<atributo>)	Devolve o maior valor de menor de um atributo em cada grupo

EXEMPLO

- Agregar os valores de um determinado atributo

```
SELECT COUNT (Nome)  
FROM Aluno
```

EXEMPLO

- Agregar os valores de um determinado atributo

```
SELECT MAX (Idade) , MIN (Idade)  
FROM Aluno
```


EXEMPLO

- Agregar os valores de um determinado atributo e agrupa por calculos por cada valor distinto

```
SELECT COUNT(Localidade)  
FROM Aluno  
GROUP BY Localidade
```

O atributo ou conjunto de atributos que estiverem na cláusula são aqueles que definem como os dados são agrupados

EXEMPLO

- Agregar os valores de um determinado atributo e agrupa por calculos por cada valor distinto

```
SELECT MAX(DataInscricao), MIN(DataInscricao)
FROM Aluno
GROUP BY Idade
```

EXEMPLO

- Agregar os valores de um determinado atributo e agrupa por calculos por cada valor distinto


```
SELECT COUNT (Nome)
FROM Aluno
GROUP BY Localidade
HAVING COUNT (Nome) > 3
```

EXEMPLO

- Agregar os valores de um determinado atributo e agrupa por calculos por cada valor distinto

```
SELECT COUNT (Nome)
FROM Aluno
GROUP BY Localidade
HAVING COUNT (Nome) > 3
```

```
SELECT COUNT (Nome)
FROM Aluno
WHERE COUNT (Nome) > 3
GROUP BY Localidade
```



Na necessidade de filtrar informação com base em funções de agregação as mesmas devem ser aplicadas na cláusula *HAVING*

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

ORDER BY

- A cláusula ORDER BY **permite ordenar registos**, de forma ascendente ou descendente. Sem o uso explícito desta cláusula não é garantida uma ordenação específica.

```
SELECT <lista_atributos>  
FROM <tabela>  
ORDER BY <ordenação> ASC | DESC
```

EXEMPLO

- Ordenar o resultado de uma determinada ordem;

```
SELECT Nome, Morada  
FROM Aluno  
ORDER BY Idade ASC
```

EXEMPLO

- Ordenar o resultado de uma determinada ordem;

```
SELECT Nome, Morada  
FROM Aluno  
ORDER BY DataInscricao DESC
```


ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

LIMIT

- A cláusula **LIMIT** **permite limitar o numero de registos que são apresentados.**

```
SELECT <lista_atributos>  
FROM <tabela>  
LIMIT <quantidade_registos>
```

```
SELECT <lista_atributos>  
FROM <tabela>  
LIMIT <quantidade_registos>, <offset>
```

EXEMPLO

- Limitar a quantidade de registos de um resultado.

```
SELECT Nome, Morada, Idade  
FROM Aluno  
LIMIT 3
```

EXEMPLO

- Limitar a quantidade de registos de um resultado.

```
SELECT Nome, Morada, Idade  
FROM Aluno  
LIMIT 3, 4
```

ESTRUTURA DQL

```
SELECT <lista_atributos>  
FROM <tabela a>  
JOIN <tabela b>  
ON <chave tabela a> = <chave tabela b>  
WHERE <condições>  
GROUP BY <agrupamento_dados>  
HAVING <condições_agrupamento>  
ORDER BY <ordenação>  
LIMIT <quantidade_registos>
```

JOIN VS WHERE

- Com WHERE

```
SELECT <select_list>
FROM <tableA>,<tableB>
WHERE <tableA.FK>=<tableB.PK>
      AND (<other_conditions>)
```

- Com JOIN

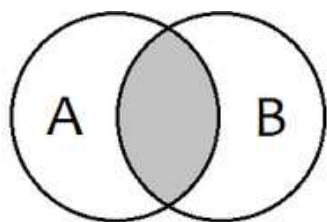
```
SELECT <select_list>
FROM <tableA>
JOIN <tableB>
ON <tableA.FK>=<tableB.PK>
WHERE <other_conditions>
```

JOINS

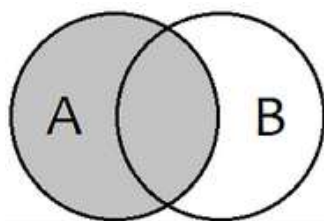
- As cláusulas JOIN são **usados para combinar colunas de duas ou mais tabelas** com base numa condição comum, geralmente uma chave estrangeira.
- Eles permitem consultar dados distribuídos em várias tabelas, unindo-os de forma eficiente.

JOINS

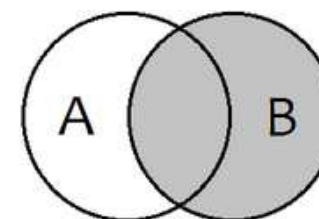
JOIN/INNER JOIN



LEFT JOIN



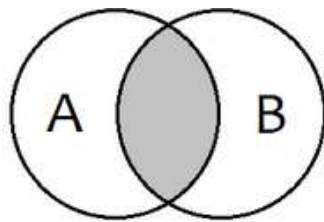
RIGHT JOIN



A tabela que estiver na cláusula JOIN corresponde à tabela B

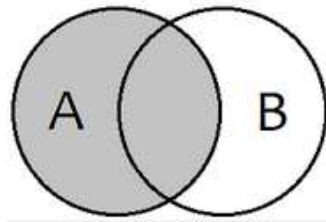
EXEMPLOS

JOIN/INNER JOIN



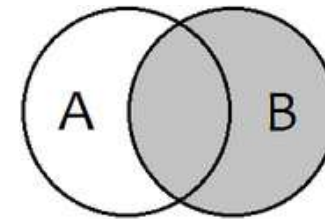
```
SELECT <lista atributos>  
FROM <Tabela A>  
JOIN <Tabela B>  
ON TabelaA.chave*=TabelaB.chave*
```

LEFT JOIN



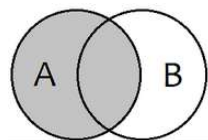
```
SELECT <lista atributos>  
FROM <Tabela A>  
LEFT JOIN <Tabela B>  
ON TabelaA.chave*=TabelaB.chave*
```

RIGHT JOIN



```
SELECT <lista atributos>  
FROM <Tabela A>  
RIGHT JOIN <Tabela B>  
ON TabelaA.chave*=TabelaB.chave*
```

JOIN/ INNER JOIN

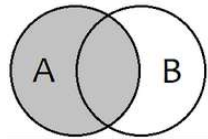


- O INNER JOIN retorna apenas os registros que possuem correspondência nas duas tabelas. Se não houver correspondência, esses registros são descartados

```
SELECT Cliente.Nome, Venda.Produto, Venda.DataVenda  
FROM Cliente  
INNER JOIN Venda ON Cliente.ID = Venda.ClienteID
```

Apenas os clientes que têm vendas serão retornados.
Clientes que não realizaram vendas serão excluídos.

LEFT JOIN

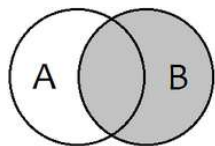


- O LEFT JOIN retorna todos os registros da tabela à esquerda, e os registros correspondentes da tabela à direita. Se não houver correspondência, os valores da tabela à direita serão *NULL*.

```
SELECT Cliente.Nome, Venda.Produto, Venda.DataVenda  
FROM Cliente  
LEFT JOIN Venda ON Cliente.ID = Venda.ClienteID
```

Todos os clientes serão retornados. Para clientes que não fizeram vendas, as colunas relacionadas à venda aparecerão como NULL

RIGHT JOIN



- O RIGHT JOIN **retorna todos os registros da tabela à direita, e os registros correspondentes da tabela à esquerda.** Se não houver correspondência, os valores da tabela à esquerda serão *NULL*.

```
SELECT Cliente.Nome, Venda.Produto, Venda.DataVenda  
FROM Cliente  
RIGHT JOIN Venda ON Cliente.ID = Venda.ClienteID
```

Todas as vendas serão retornadas. Se alguma venda não tiver um cliente correspondente, o nome do cliente será NULL

JOINS ATRAVÉS DO WHERE

- O conceito é o mesmo: combinar registros de várias tabelas com base em uma condição de correspondência.

```
SELECT Cliente.Nome, Venda.Produto, Venda.DataVenda  
FROM Cliente, Venda  
WHERE Cliente.ID = Venda.ClienteID
```

EXEMPLO

- Com WHERE

```
SELECT <lista atributos>  
FROM <Tabela A>,<Tabela B>  
WHERE TabelaA.chave*=TabelaB.chave*  
      AND (<Restantes condições>)
```

- Com JOIN

```
SELECT <lista atributos>  
FROM <Tabela A>  
JOIN <Tabela B>  
ON TabelaA.chave*=TabelaB.chave*  
WHERE <other_conditions>
```

JOIN VS WHERE

- JOINS **são considerados mais legíveis**, especialmente quando existe a necessidade de agregar várias tabelas numa única pesquisa
- WHERE não favorece do mesmo nível de controlo que os JOINS (INNER, LEFT, RIGHT, etc).
- A otimização na construção inicial das pesquisas podem trazer benefícios no futuro.



www.cesae.pt

