Consultas no SQL Server - SELECT

O comando SELECT é usado para **consultar dados** no Server. Ele permite escolher **quais colunas mostrar**, **de quais tabelas extrair os dados**, e **como filtrar, agrupar ou ordenar os resultados**.

Estrutura de Comando

```
SELECT [DISTINCT|TOP N] colunas
FROM tabela
[WHERE condição]
[ORDER BY coluna ASC|DESC];
```

Explicação das Partes do Comando

Parte	Descrição
SELECT	Define quais colunas serão exibidas
DISTINCT	Elimina valores duplicados
TOP N	Mostra apenas os primeiros N registros
FROM	Informa a tabela de origem
WHERE	Filtra os dados com base em condições
ORDER BY	Ordena os dados por uma ou mais colunas

Exemplos de SELECT considerando a Tabela Alunos abaixo:

ID	Nome	Idade	Cidade
1	João Silva	20	São Paulo
2	Maria Lima	22	Rio de Janeiro
3	Ana Costa	19	Belo Horizonte

Selecionar todas as colunas

```
SELECT * FROM Alunos;
```

Selecionar colunas específicas

```
SELECT Nome, Idade FROM Alunos;
```

Filtrar por condição (WHERE)

```
SELECT * FROM Alunos WHERE Idade > 20;
```

Ordenar resultados (ORDER BY)

```
SELECT * FROM Alunos ORDER BY Idade ASC;
```

Eliminar duplicados (DISTINCT)

```
SELECT DISTINCT Cidade FROM Alunos;
```

Selecionar os 2 alunos mais velhos com idade acima de 18 anos

```
SELECT TOP 2 Nome, Idade
FROM Alunos
WHERE Idade > 18
ORDER BY Idade DESC;
```

WHERE – Filtrando Registros

A cláusula WHERE é usada para **restringir os resultados de uma consulta**, retornando **apenas os registros que atendem a uma ou mais condições**.

Operadores Relacionais

Operador	Significado	Exemplo
=	Igual a	WHERE Idade = 20
<> ou !=	Diferente de	WHERE Cidade <> 'São Paulo'
>	Maior que	WHERE Idade > 18
<	Menor que	WHERE Idade < 30
>=	Maior ou igual a	WHERE Idade >= 21
<=	Menor ou igual a	WHERE Idade <= 25

Exemplo:

```
SELECT * FROM Alunos WHERE Idade >= 20;
```

AND e OR – Operadores Lógicos

- **AND**: todas as condições devem ser verdadeiras
- **OR**: pelo menos uma condição deve ser verdadeira

Exemplo com AND

```
SELECT * FROM Alunos WHERE Idade > 18 AND Cidade = 'São Paulo';
```

Traz apenas alunos **com mais de 18 anos e que moram em São Paulo**.

Exemplo com OR

```
SELECT * FROM Alunos WHERE Cidade = 'São Paulo' OR Cidade = 'Rio de Janeiro';
```

Traz alunos **de São Paulo ou do Rio de Janeiro**.

BETWEEN – Faixa de Valores

O BETWEEN é usado para verificar se um valor está **entre dois valores (inclusive)**.

```
SELECT * FROM Alunos WHERE Idade BETWEEN 18 AND 21;
```

Traz alunos com **idade igual a 18, 19, 20 ou 21**.

LIKE – Busca por Padrão

O LIKE é usado para **buscar valores que seguem um determinado padrão**. Ele aceita **caracteres curingas**, sendo o mais comum o %.

Como funciona o %?

- % substitui **qualquer sequência de caracteres**
- Pode ser usado **antes**, **depois** ou **em ambos os lados** do texto

Padrão	Resultado
'A%'	Começa com A (ex: Ana, André, Alice)
'%a'	Termina com a (ex: Ana, Maria, Júlia)
'%ar%'	Contém “ar” em qualquer posição (ex: Carlos, Maria, Clara)

Exemplo prático:

```
SELECT * FROM Alunos WHERE Nome LIKE 'J%';
```

Traz alunos com nome **iniciando com a letra "J"** (João, Juliana, José...).

Combinação com AND e LIKE:

```
SELECT * FROM Alunos WHERE Nome LIKE '%Silva%' AND Idade > 18;
```

Traz alunos **que contém o termo "Silva" no nome e possui idade acima de 18 anos**.

Dicas Importantes

- Use parênteses para organizar consultas com AND e OR:

```
SELECT * FROM Alunos WHERE Cidade = 'São Paulo' AND (Idade = 20 OR Idade = 22);
```

Traz alunos **da cidade de São Paulo com idade igual a 20 ou igual a 22 anos.**

ORDER BY – Ordenando Resultados

Organiza os dados em ordem crescente (ASC) ou decrescente (DESC).

```
SELECT * FROM Alunos ORDER BY Cidade ASC, Nome DESC;
```

Traz todos os alunos **ordenados primeiro por cidade em ordem alfabética crescente (A–Z) e, dentro de cada cidade, os alunos são ordenados por nome em ordem alfabética decrescente (Z–A).**

JOINs – Combinando Tabelas

JOINs permitem **relacionar registros de diferentes tabelas** com base em colunas em comum (normalmente chaves primárias e estrangeiras).

Tabelas de Exemplo

Tabela Alunos

AlunoID	Nome
1	João
2	Maria
3	Ana
4	Pedro

Tabela Eventos

EventoID	AlunoID	NomeEvento
1	1	Feira de Ciências
2	2	Hackathon
3	1	Mostra de Robótica

João participou de dois eventos, **Maria** participou de um, **Ana** e **Pedro** não participaram de nenhum evento.

INNER JOIN

```
SELECT Alunos.Nome, Eventos.NomeEvento
FROM Alunos
INNER JOIN Eventos ON Alunos.AlunoID = Eventos.AlunoID;
```

Traz apenas os alunos que participaram de pelo menos um evento. Em outras palavras, traz apenas os registros que são comuns entre as duas tabelas, levando em consideração a coluna AlunoID.

Resultado:

Nome	NomeEvento
João	Feira de Ciências
João	Mostra de Robótica
Maria	Hackathon

LEFT JOIN

```
SELECT Alunos.Nome, Eventos.NomeEvento
FROM Alunos
LEFT JOIN Eventos ON Alunos.AlunoID = Eventos.AlunoID;
```

Retorna todos os alunos, inclusive aqueles que não participaram de nenhum evento. Para os alunos que não possuem eventos associados, a coluna NomeEvento será exibida como NULL. Em outras palavras, o LEFT JOIN retorna todos os registros da tabela à esquerda do comando JOIN. Já da tabela à direita, apenas os registros que possuem correspondência com base na condição especificada (neste caso, AlunoID) serão retornados. Caso não haja correspondência, as colunas da tabela da direita terão valores NULL.

Resultado:

Nome	NomeEvento
João	Feira de Ciências
João	Mostra de Robótica
Maria	Hackathon
Ana	NULL
Pedro	NULL

RIGHT JOIN

```
SELECT Alunos.Nome, Eventos.NomeEvento
FROM Alunos
RIGHT JOIN Eventos ON Alunos.AlunoID = Eventos.AlunoID;
```

Retorna todos os eventos, inclusive aqueles que não possuem alunos associados. Para os eventos que não têm alunos vinculados, a coluna Nome será exibida como NULL. Em outras palavras, o RIGHT JOIN retorna todos os registros da tabela à direita do comando JOIN. Já da tabela à esquerda, apenas os registros que possuem correspondência com base na condição especificada (neste caso, AlunoID) serão retornados. Caso não haja correspondência, as colunas da tabela da esquerda terão valores NULL.

Resultado:

Nome	NomeEvento
João	Feira de Ciências
João	Mostra de Robótica
Maria	Hackathon

Resumo Geral sobre JOIN

Comando	Função
INNER JOIN	Retorna registros com correspondência em ambas as tabelas
LEFT JOIN	Retorna todos da esquerda + os que batem na direita
RIGHT JOIN	Retorna todos da direita + os que batem na esquerda