Comando SELECT

O comando `select` recupera os dados de uma ou mais tabelas, sendo um dos comandos mais simples e, ao mesmo tempo, mais extenso da SQL devido as suas funções, operandos, comandos, sub-comandos e cláusulas não obrigatórias.

Para os exemplos que se seguem adotaremos as tabelas predefinidas apresentadas na **Figura 1**, conforme vemos nas **Tabelas 1** a **4**.

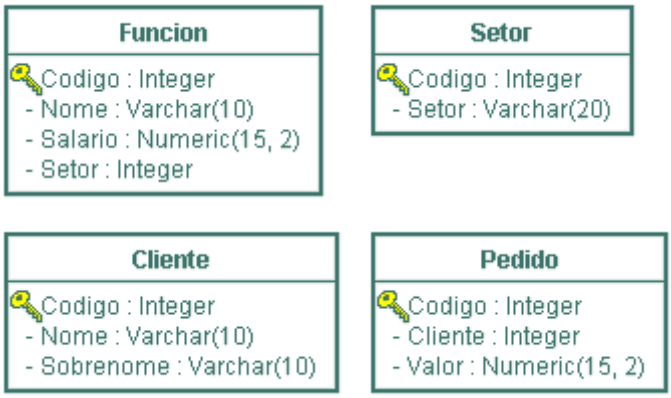

Figura 1 – Tabelas predefinidas

Function			
Codigo	**Nome**	**Salario**	**Setor**
1	Tadeu	1.500,00	1
2	Ylane	1.200,00	2
3	Julian	1.000,00	1
4	Ewerton	1.000,00	1
5	João	800,00	2
6	Geraldo	1.500,00	3
7	Maria	500,00	

Tabela 1 – Dados inseridos na tabela Function

Setor	
Codigo	**Setor**
1	Desenvolvimento

2	Manutenção
3	Financeiro
4	Vendas

Tabela 2 – Dados inseridos na tabela Setor

Cliente		
Codigo	**Nome**	**Sobrenome**
1	Francisco	Silva
2	José	Lima
3	Maria	Silva
4	Adriana	Ferreira
5	João	Oliveira
6	Eduarda	Souza

Tabela 3 – Dados inseridos na tabela Cliente

Pedido		
Codigo	**Cliente**	**Valor**
1	2	1.000,00
2	4	2.000,00
3	2	1.500,00
4	5	2.500,00
5	2	1.000,00

Tabela 4 – Dados inseridos na tabela Pedido

Select simples

A seguir vemos a sintaxe e exemplos com resultados apresentados nas **Tabelas 5** e **6**.

```
    select  Coluna, Coluna, ..., Coluna from  Tabela
```

- **Tabela** - Nome da tabela
- **Coluna** - Nome de uma coluna

Para mostrar todas as colunas pode-se colocar apenas a máscara `*` no lugar do nome das colunas.

```
select Codigo, Nome from Funcion
```

Codigo	Nome
1	Tadeu
2	Ylane
3	Julian
4	Ewerton
5	João
6	Geraldo
7	Maria

Tabela 5 – Resultado do Select simples anterior

```
select * from Funcion
```

Codigo	Nome	Salario	Setor
1	Tadeu	1500	1
2	Ylane	1200	2
3	Julian	1000	1
4	Ewerton	1000	1
5	João	800	2
6	Geraldo	1500	3

7	Maria	500	

Tabela 6 – Resultado do Select simples do exemplo 2

Cláusula where com condições simples

Podemos filtrar colunas para nos mostrar apenas os dados que nos interessa através da cláusula `where` em conjunto com os operadores comparativos. A seguir vemos a sintaxe:

```
select  Coluna, Coluna, ..., Coluna from  Tabela where Condição
```

- **Tabela** - Nome da tabela
- **Coluna** - Nome de uma coluna
  - Para mostrar todas as colunas pode-se colocar apenas a máscara “`*`” no lugar do nome das colunas
- **Condição** - Cria uma condição para filtrar os dados utilizando os operadores comparativos

Operador Comparativo “=” (Igual)

A seguir temos um exemplo com resultado apresentado na **Tabela 7.**

```
select Codigo, Nome from Funcion where Nome = ‘Tadeu’
```

Codigo	Nome
1	Tadeu

Tabela 7 – Resultado do exemplo anterior

Operador Comparativo “<>” (Diferente)

A seguir temos um exemplo com resultado apresentado na **Tabela 8**.

```
select Codigo, Nome from Funcion where Nome <> ‘Tadeu’
```

Codigo	Nome
2	Ylane
3	Julian
4	Ewerton
5	João
6	Geraldo
7	Maria

Tabela 8 – Resultado do exemplo anterior

Operador Comparativo “>” (Maior que)

A seguir temos um exemplo com resultado apresentado na **Tabela 9**.

```
select Codigo, Nome from Funcion where Nome > ‘Tadeu’
```

Codigo	Nome
2	Ylane

Tabela 9 – Resultado do exemplo anterior

Operador Comparativo “>=” (Maior que ou Igual)

A seguir temos um exemplo com resultado apresentado na **Tabela 10.**

```
select Codigo, Nome from Funcion where Nome >= ‘Tadeu’
```

Codigo	Nome
1	Tadeu
2	Ylane

Tabela 10 – Resultado do exemplo anterior

Operador Comparativo “<” (Menor que)

A seguir temos um exemplo com resultado apresentado na **Tabela 11.**

```
select Codigo, Nome from Funcion where Nome < ‘Tadeu’
```

Codigo	Nome
3	Julian
4	Ewerton
5	João
6	Geraldo
7	Maria

Tabela 11 – Resultado do exemplo anterior

Operador Comparativo “<=” (Menor que ou Igual)

A seguir temos um exemplo com resultado apresentado na **Tabela 12.**

```
select Codigo, Nome from Funcion where Nome <= ‘Tadeu’
```

Codigo	Nome
1	Tadeu
3	Julian
4	Ewerton
5	João
6	Geraldo
7	Maria

Tabela 12 – Resultado do exemplo anterior

Operador Comparativo “between ... and ...” (Entre dois valores)

A seguir temos um exemplo com resultado apresentado na **Tabela 13.**

```
select Codigo, Nome from Funcion where Nome between ‘João’ and ‘Tadeu’
```

Codigo	Nome
1	Tadeu
3	Julian
5	João
7	Maria

Tabela 13 – Resultado do exemplo anterior

Operador Comparativo “not between ... and ...” (Não está entre dois valores)

A seguir temos um exemplo com resultado apresentado na **Tabela 14**.

```
select Codigo, Nome from Funcion where Nome not between ‘João’ and ‘Tadeu
```

Codigo	Nome
2	Ylane
4	Ewerton
6	Geraldo/td>

Tabela 14 – Resultado do exemplo anterior

Operador Comparativo “in(lista)” (Igual a qualquer valor da lista)

A seguir temos um exemplo com resultado apresentado na **Tabela 15**.

```
select Codigo, Nome from Funcion where Nome in (‘João’,‘Tadeu’)
```

Codigo	Nome
1	Tadeu
5	João

Tabela 15 – Resultado do exemplo anterior

Operador Comparativo “not in(lista)” (Diferente de qualquer valor da lista)

A seguir temos um exemplo com resultado apresentado na **Tabela 16.**

```
select Codigo, Nome from Funcion where Nome not in (‘João’,‘Tadeu’)
```

Codigo	Nome
2	Ylane
3	Julian
4	Ewerton
6	Geraldo
7	Maria

Tabela 16 – Resultado do exemplo anterior

Operador Comparativo “like” (Pesquisa uma cadeia de caractere)

A seguir temos um exemplo com resultado apresentado na **Tabela 17.**

```
select Codigo, Nome from Funcion where Nome like‘J%’
```

Obs.: A máscara no operador `like` usada foi `%`, porém ele pode mudar de um SGBDR para outro

Codigo	Nome
3	Julian
5	João

Tabela 17 – Resultado do exemplo anterior

Operador Comparativo “is null” (Valor nulo)

A seguir temos um exemplo com resultado apresentado na **Tabela 18.**

```
select Codigo, Nome from Funcion where Setor is null
```

Codigo	Nome
7	Maria

Tabela 18 – Resultado do exemplo anterior

Operador Comparativo “is not null” (Valor não nulo)

A seguir temos um exemplo com resultado apresentado na **Tabela 19.**

```
select Codigo, Nome from Funcion where Setor is not null
```

Codigo	Nome
1	Tadeu
2	Ylane
3	Julian

4	Ewerton
5	João
6	Geraldo

Tabela 19 – Resultado do exemplo anterior

Cláusula where com condições complexas

Para filtrar dados que requerem condições complexas utilizamos a cláusula `where` junto com os operadores comparativos e lógicos. A seguir vemos um exemplo:

```
    select  Coluna, Coluna, ..., Coluna from  Tabela where Condição Operador_
```

- **Tabela** - Nome da tabela
- **Coluna** - Nome de uma coluna
  - Para mostrar todas as colunas pode-se colocar apenas a máscara "`*`" no lugar do nome das colunas
- **Condição** - Cria uma condição para filtrar os dados utilizando os operadores comparativos
- **Operador Lógico** - Operador lógico and ou or para unir as duas condições no mesmo filtro

As condições complexas seguem regras de precedência descritas abaixo:

1. Expressões entre parênteses "(...)"
2. Todos os operadores de comparação "**=, <>, >, >=, <, <=, in...**"
3. Operador lógico and
4. Operador lógico or

Caso duas condições estejam na mesma ordem de precedência, terá maior precedência a que estiver mais próxima da cláusula `where`.

Operador Lógico "and" (E)

A seguir temos um exemplo com resultado apresentado na **Tabela 20**.

```
select Codigo, Nome from Funcion where Setor is not null and Codigo = 1
```

Codigo	Nome
1	Tadeu

Tabela 20 – Resultado do exemplo anterior

Operador Lógico “or” (OU)

A seguir temos um exemplo com resultado apresentado na **Tabela 21**.

```
select Codigo, Nome from Funcion where Nome = ‘Tadeu’ or Nome = ‘Ylane’
```

Codigo	Nome
1	Tadeu
2	Ylane

Tabela 21 – Resultado do exemplo anterior

Cláusula order by

Os registros mostrados podem está com uma ou mais colunas ordenadas de modo ascendente ou descendente. Utilizando-se para isso a cláusula `order by`, como mostra a sintaxe a seguir:

```
select Coluna, Coluna, ..., Coluna from Tabela where Condição [Opcional]
order by Coluna_Ordenada Modo_Ordenação
```

- **Tabela** - Nome da tabela
- **Coluna** - Nome de uma coluna
  - Para mostrar todas as colunas pode-se colocar apenas a máscara `*` no lugar do nome das colunas
- **Condição** - Cria uma condição para filtrar os dados utilizando os operadores comparativos
- **Coluna_Ordenada** - Coluna que será ordenada
- **Modo_Ordenação** - Modo que a coluna será ordenada
  - **asc** - Modo ascendente [Opcional]
  - **desc** - Modo descendente

Modo “asc” (Ascendente)

A seguir temos um exemplo com resultado apresentado na **Tabela 22.**

```
select Codigo, Nome from Funcion order by Nome asc
```

Ou

```
select Codigo, Nome from Funcion order by Nome
```

Codigo	Nome
4	Ewerton
6	Geraldo
5	João

3	Julian
7	Maria
1	Tadeu
2	Ylane

Tabela 22 – Resultado do exemplo anterior

Modo “desc” (Descendente)

A seguir temos um exemplo com resultado apresentado na **Tabela 23.**

```
select Codigo, Nome from Funcion order by Nome desc
```

Codigo	Nome
2	Ylane
1	Tadeu
7	Maria
3	Julian
5	João
6	Geraldo
4	Ewerton

Tabela 23 – Resultado do exemplo anterior

Comando join

Com o comando select em conjunto com o join podemos fazer a junção de duas ou mais tabelas num mesmo resultado. A seguir vemos a sintaxe:

```
select  Tabela1.Coluna, Tabela1.Coluna, ..., Tabela2.Coluna
from  Tabela1
cláusula join Tabela2 on  Condição
```

Onde:

- Tabela n: Nome da tabela n;
- Tabela1.Coluna: Nome de uma coluna da Tabela 1
- Tabela2.Coluna: Nome de uma coluna da Tabela2
- Cláusula: Cláusula do comando `join`, inner [Opcional], outer
- Condição: Cria uma condição para filtrar os dados.

Cláusula inner

Com essa cláusula só serão mostrados os registros com referências nas tabelas da junção, como vemos no exemplo a seguir e resultado apresentado na **Tabela 24**.

```
select a.Codigo, a.Nome, b.Setor
from Funcion a
inner join Setor b on (b.Codigo = a.Setor)
```

Ou

```
select a.Codigo, a.Nome, b.Setor
from Funcion a
join Setor b on (b.Codigo = a.Setor)
```

Codigo	Nome	Setor
1	Tadeu	Desenvolvimento
2	Ylane	Manutenção

3	Julian	Desenvolvimento
4	Ewerton	Desenvolvimento
5	João	Manutenção
6	Geraldo	Financeiro

Tabela 24 – Resultado do exemplo anterior

Cláusula outer

Com essa cláusula serão mostrados todos os registros com ou sem referências nas tabelas da junção. A seguir vemos um exemplo de sintaxe e o resultado na **Tabela 25.**

```
select a.Codigo, a.Nome, b.Setor
from Funcion a
left outer join Setor b on (b.Codigo = a.Setor)
```

Codigo	Nome	Setor
1	Tadeu	Desenvolvimento
2	Ylane	Manutenção
3	Julian	Desenvolvimento
4	Ewerton	Desenvolvimento
5	João	Manutenção
6	Geraldo	Financeiro
7	Maria	

Tabela 25 – Resultado do exemplo anterior

A palavra reservada `left` indica que os registros da primeira tabela serão mostrados, independentemente deles terem ou não referência na segunda tabela.Trocando-se

o `left` por `rigth` a ordem é invertida, como vemos no exemplo a seguir e resultado apresentado na **Tabela 26**.

```
select a.Codigo, a.Nome, b.Setor
from Funcion a
right outer join Setor b on (b.Codigo = a.Setor)
```

Codigo	Nome	Setor
1	Tadeu	Desenvolvimento
2	Ylane	Manutenção
3	Julian	Desenvolvimento
4	Ewerton	Desenvolvimento
5	João	Manutenção
6	Geraldo	Financeiro
7		Vendas

Tabela 26 – Resultado do exemplo anterior

A palavra reservada `full` indica que todos os registros das duas tabelas serão mostrados, independentemente deles terem ou não referência, como vemos no exemplo a seguir e com resultado apresentado na **Tabela 27**.

```
select a.Codigo, a.Nome, b.Setor
from Funcion a
full outer join Setor b on (b.Codigo = a.Setor)
```

Codigo	Nome	Setor
1	Tadeu	Desenvolvimento
2	Ylane	Manutenção

3	Julian	Desenvolvimento
4	Ewerton	Desenvolvimento
5	João	Manutenção
6	Geraldo	Financeiro
7	Maria	
		Vendas

Tabela 27 – Resultado do exemplo anterior

Comando union

O comando `union` também faz junções de tabelas, só que nesse caso a união das colunas é na vertical, diferentemente das junções vistas anteriormente que eram na horizontal. O inconveniente nesse caso é que as colunas que se deseja mostra numa determinada posição deveram ter o mesmo tipo. A seguir vemos a sitaxe e um exemplo com resultado apresentado na **Tabela 28**.

```
select  Coluna, Coluna, ..., Coluna from  Tabela1
union
select  Coluna, Coluna, ..., Coluna from  Tabela2
```

Tabela n	Nome da tabelan
Coluna	Nome de uma coluna

Tabela 28 – Resultado do exemplo anterior

A seguir temos um exemplo com resultado apresentado na **Tabela 29**.

```
select Nome from Funcion
union
select Nome from Cliente
```

Nome
Adriana
Eduarda
Ewerton
Francisco
Geraldo
José
João
Julian
Maria
Tadeu
Ylane

Tabela 29 – Resultado do exemplo anterior

Cláusula all

Por default os registros duplicados são eliminados do resultado, para mostrar todos os registros, idênticos ou não, utilizamos a cláusula `all`. A seguir vemos um exemplo com resultado apresentado na **Tabela 30**.

```
select Nome from Funcion
union all
select Nome from Cliente
```

Nome

Tadeu
Ylane
Julian
Ewerton
João
Geraldo
Maria
Francisco
José
Maria
Adriana
João
Eduarda

Tabela 30 – Resultado do exemplo anterior

Ninhos de Pesquisa (Nested Queries)

Quando queremos restringir os dados mostrados em uma consulta principal, dependendo do resultado de uma sub-consulta, chamamos esse processo de ninhos de pesquisa. Confira a sintaxe:

```
select  Coluna, Coluna, ..., Coluna from  Tabela1
where coluna Condição (
select  Coluna, Coluna, ..., Coluna from  Tabela2
where Condição)
```

Onde:

- Tabela n: Nome da tabela n;
- Coluna: Nome de uma coluna;

- Condição: Cria uma Condição para filtrar os dados. A segunda condição é opcional;

A seguir vemos dois exemplos, com resultados apresentados nas **Tabelas 31** e **32.**

```
select Codigo, Nome from Cliente
where Codigo in ( select Cliente from Pedido )
```

Codigo	Nome
2	José
4	Adriana
5	João

Tabela 31 – Resultado do exemplo anterior

```
select Codigo, Nome from Cliente
where Codigo in ( select Cliente from Pedido where Valor >= 2000 )
```

Codigo	Nome
4	Adriana
5	João

Tabela 32 – Resultado do exemplo anterior

Função avg

Retorna a média aritmética da coluna informada. A seguir vemos a sintaxe e um exemplo com resultado apresentado na **Tabela 33.**

```
select  avg(Coluna) [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela
- Coluna: Nome de uma coluna
- [*as New_Coluna*]: Nome da coluna de retorno [Opcional]

```
select avg(Salario) as Media from Funcion
```

Media
1071,42

Tabela 33 – Resultado do exemplo anterior

Função max

Retorna o maior valor da coluna informada. A seguir vemos a sintaxe:

```
select  max(Coluna) [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela;
- Coluna: Nome de uma coluna
- [as New_Coluna]: Nome da coluna de retorno [Opcional]

A seguir temos um exemplo com resultado apresentado na **Tabela 34**.

```
select max(Codigo) as Maior_Codigo from Funcion
```

Maior_Codigo
7

Tabela 34 – Resultado do exemplo anterior

Função min

Retorna o menor valor da coluna informada, como vemos na sintaxe:

```
    select  min(Coluna) [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela
- Coluna: Nome de uma coluna
- [as New_Coluna]: Nome da coluna de retorno [Opcional]

A seguir temos um exemplo com resultado apresentado na **Tabela 35**.

```
    select min(Codigo) as Menor_Codigo from Funcion
```

Menor_Codigo
1

Tabela 35 – Resultado do exemplo anterior

Função sum

Retorna o somatório da coluna informada. A seguir vemos a sintaxe:

```
select  sum(Coluna) [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela
- Coluna: Nome de uma coluna
- [as New_Coluna]: Nome da coluna de retorno [Opcional]

A seguir temos um exemplo com resultado apresentado na **Tabela 36**.

```
select sum(Salario) as Total from Funcion
```

Total
7500

Tabela 36 – Resultado do exemplo anterior

Função count

Retorna a quantidade de registros existentes. A seguir vemos a sintaxe e o exemplo com resultado na **Tabela 37**.

```
select  count(Coluna) [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela;
- Coluna: Nome de uma coluna – Pode ser substituído pelo `*` (asterisco);
- [as New_Coluna]: Nome da coluna de retorno [Opcional]

```
    select count(Codigo) as QtTotal from Funcion
```

QtTotal
7

Tabela 37 – Resultado do exemplo anterior

Função distinct

Usamos `distinct` para evitar que um determinado valor seja repetido em uma consulta.

```
    select  distinct(Coluna) from  Tabela
```

- Tabela: Nome da tabela;
- Coluna: Nome de uma coluna.

A seguir temos um exemplo com resultado apresentado na **Tabela 38**.

```
    select distinct(Cliente) from Pedido
```

Cliente
2
4
5

Tabela 38 – Resultado do exemplo anterior

Cláusula group by

Quando queremos agrupar o resultado de uma ou mais funções com os dados de uma ou mais colunas devemos usar o `group by`. A seguir vemos a sintaxe e o exemplo com resultado apresentado na **Tabela 39**.

```
select  Função, Coluna from Tabela
group by Coluna
```

- Tabela: Nome da tabela;
- Função: Função a ser unida com a coluna;
- Coluna: Nome de uma coluna.

```
select sum(Valor) as Soma, Cliente from Pedido
group by Cliente
```

Soma	*Cliente*
2500	2
2000	4
2500	5

Tabela 39 – Resultado do exemplo anterior

Cláusula having

`Having` é utilizada para filtrar o resultado de uma função de agrupamento juntamente com a cláusula `group by`.

```
select  Função, Coluna from Tabela
group by Coluna
having Condição
```

- Tabela: Nome da tabela;
- Função: Função a ser unida com a coluna;
- Coluna: Nome de uma coluna;
- Condição: cria condição para filtrar o resultado da função.

A seguir temos um exemplo com resultado apresentado na **Tabela 40**.

```
select sum(Valor) as Soma, Cliente from Pedido
group by Cliente
having sum(Valor) >= 2500
```

Soma	*Cliente*
2500	2
2500	5

Tabela 40 – Resultado do exemplo anterior

Operador de concatenação (||)

Para combinar cadeias de caracteres utilizamos o operador `||`. A seguir vemos a sintaxe utilizada:

```
select  Coluna || [String ||] Coluna [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela;

- Coluna: Nome de uma coluna;
- String: String para separar as colunas [Opcional];
- [as New_Coluna]: Nome da coluna de retorno [Opcional].

A seguir temos um exemplo com resultado apresentado na **Tabela 41**.

```
select Nome || ' ' || Sobrenome as Nome_Completo from Cliente
```

Nome_Completo
Francisco Silva
José Lima
Maria Silva
Adriana Ferreira
João Oliveira
Eduarda Souza

Tabela 41 – Resultado do exemplo anterior

Operadores aritméticos

Podemos fazer cálculos com operações aritméticas em qualquer declaração de cláusula SQL, com exceção para a cláusula `from`.

```
select  Coluna, Operação_Aritmética [as New_Coluna] from  Tabela
```

- Tabela: Nome da tabela;
- Coluna: Nome de uma coluna;
- Operação_Aritmética: Operação aritmética desejada;
- [as New_Coluna]: Nome da coluna de retorno [Opcional]

Operadores aritméticos “+” (Adição)

A seguir temos um exemplo com resultado apresentado na **Tabela 42.**

```
select Nome, (Salario + 200) as Salario_Atual from Funcion
```

Nome	Salario_Atual
Tadeu	1700
Ylane	1400
Julian	1200
Ewerton	1200
João	1000
Geraldo	1700

Tabela 42 – Resultado do exemplo anterior de adição

Operadores aritméticos “-” (Subtração)

A seguir temos um exemplo com resultado apresentado na **Tabela 43.**

```
select Nome, (Salario - 200) as Salario_Atual from Funcion
```

Nome	Salario_Atual
Tadeu	1300
Ylane	1000
Julian	800
Ewerton	800
João	600
Geraldo	1300

Maria	300

Tabela 43 – Resultado do exemplo anterior de subtração

Operadores aritméticos “*” (Multiplicação)

A seguir temos um exemplo com resultado apresentado na **Tabela 44.**

```
select Nome, (Salario * 1.5) as Salario_Atual from Funcion
```

Nome	Salario_Atual
Tadeu	2250
Ylane	1800
Julian	1500
Ewerton	1500
João	1200
Geraldo	2250
Maria	750

Tabela 44 – Resultado do exemplo anterior de multiplicação

Operadores aritméticos “/” (Divisão)

A seguir temos um exemplo com resultado apresentado na **Tabela 45.**

```
select Nome, (Salario / 2) as Salario_Quinzena from Funcion
```

Nome	Salario_Atual
Tadeu	750
Ylane	600
Julian	500
Ewerton	500
João	400
Geraldo	750
Maria	250

Tabela 45 – Resultado do exemplo anterior de divisão
