7 – Introdução

- De nada adianta a possibilidade de inserir e manipular registros no banco de dados, se não for possível recuperá-los quando necessário.

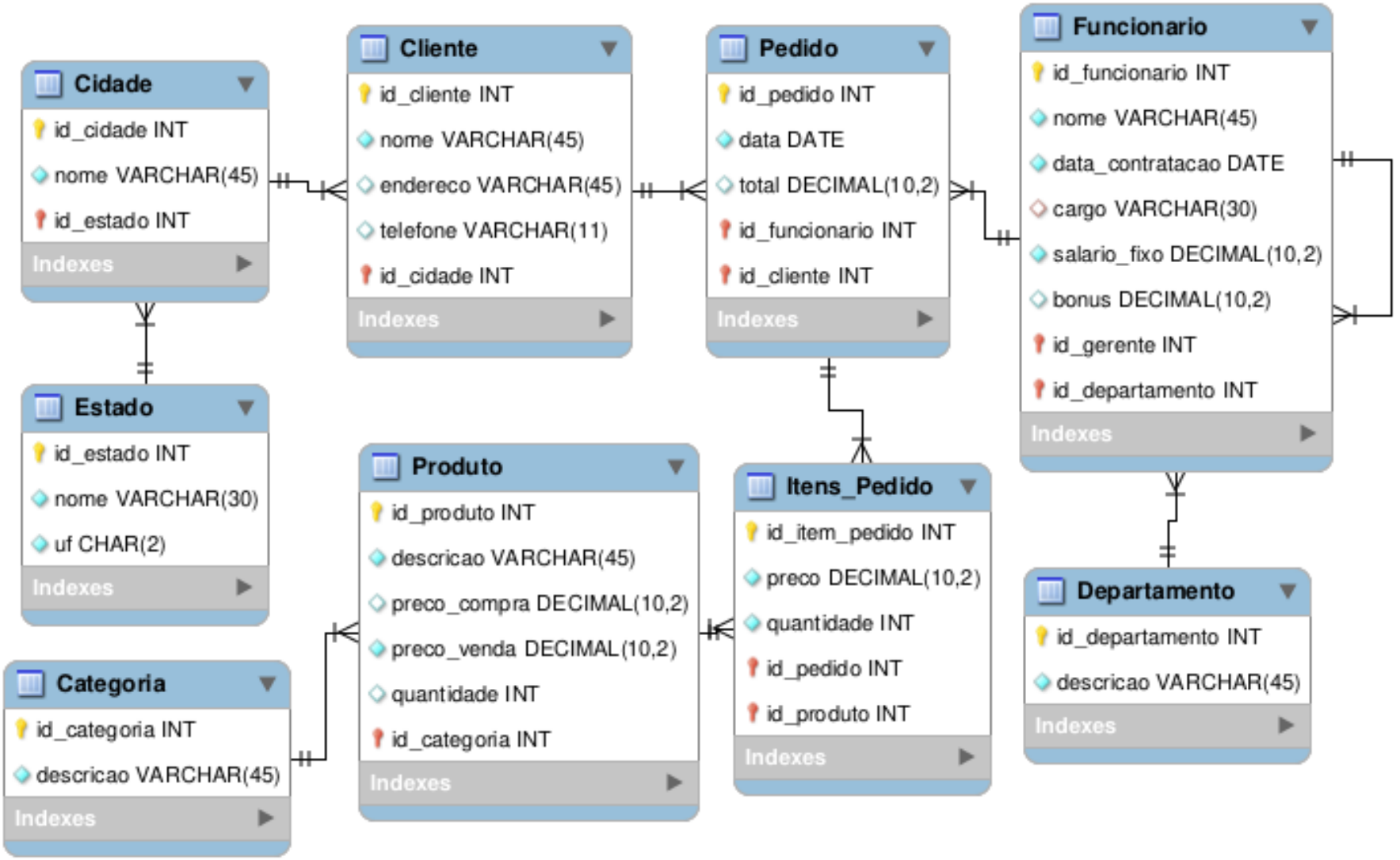
- Para recuperar informações no banco de dados, utiliza-se o comando **SELECT**, juntamente com diversos outros parâmetros que permitem ao desenvolvedor filtrar os dados de acordo com as suas necessidades.

- Existem diversas formas de se construir consultas SQL, todas retornando o mesmo resultado. Entretanto, deve-se tomar cuidado com o desempenho da consulta.

- Consultas SQL elaboradas de forma incorreta, e que forem executadas em um banco de dados com grande quantidade de registros, podem ser demoradas e comprometer o desempenho de todo o banco.

7 – Introdução

- Será usado o banco de dados comercial implementado anteriormente:



7.1 – Comando SELECT

- O comando SQL para efetuar consultas no banco de dados é o SELECT, seguido dos campos e do nome da tabela.

- A instrução de consulta segue a seguinte sintaxe:

SELECT <campos da tabela> **FROM** <nome da tabela>;

Onde:

<campos da tabela>: Nome dos campos que deverão ser retornados da consulta. Caso seja colocado o '*' (asterisco), todos os campos da tabela serão retornados.

<nome da tabela>: Nome da tabela cuja a consulta será realizada.

7.1 – Comando SELECT

- Como exemplo, o comando abaixo ilustra uma consulta onde são retornados o nome e o endereço de todos os clientes cadastrados na tabela cliente.

```
SELECT nome, endereco FROM cliente;
```

nome	endereco
Gabriel Jesus	Rua Juca, 45
Fernando Pras	Rua XV, 55
Dudu da Silva	Rua Guara, 45
Felipe Melo	Rua Brasil, 52
Roger Guedes	Rua Guedes, 45
William Bigode	Rua 19, 21
Miguel Borja	Rua João, 40
Zé Roberto da Silva	Rua XV, 53
Jean da Silva	Rua Atlantica, 9

7.1 – Comando SELECT

- A consulta abaixo ilustra uma consulta que retorna todos os campos de todos os clientes da tabela.

```
SELECT * FROM cliente;
```

id_cliente	nome	endereco	id_cidade	telefone
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
3	Dudu da Silva	Rua Guara, 45	2	44999066878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
5	Roger Guedes	Rua Guedes, 45	9	47988566878
6	William Bigode	Rua 19, 21	5	25999566878
7	Miguel Borja	Rua João, 40	6	44999566878
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
9	Jean da Silva	Rua Atlantica, 9	1	77699566878

7.2 – Comando DISTINCT

- O comando **DISTINCT** tem a função de remover linhas duplicadas de uma consulta. Sua sintaxe é ilustrada no quadro abaixo:

SELECT DISTINCT <campo 1>, <campo 2> **FROM** <nome da tabela>;

Onde:

<campos da tabela>: Nome dos campos que deverão ser filtrados, e no caso de repetição, não aparecerão na consulta.

<nome da tabela>: Nome da tabela cuja a consulta será realizada.

7.2 – Comando DISTINCT

- Um exemplo de utilização do "**DISTINCT**" seria a necessidade de selecionar todas as cidades que possuem pelo menos um cliente cadastrado.

- A princípio, a consulta SQL abaixo, onde são listadas apenas o campo "id_cidade" da tabela cliente resolveria o problema. Contudo, devido ao grande número de clientes, bem como os inúmeros clientes na mesma cidade, dificultaria o processo de identificar as cidades.

```
SELECT id_cidade FROM cliente;
```

id_cidade
1
1
1
2
3
5
6
9
9

7.2 – Comando DISTINCT

- Para resolver este problema, podemos usar o comando DISTINCT, que irá mostrar todas as cidades, excluindo as repetições.

```
SELECT DISTINCT id_cidade FROM cliente;
```

id_cidade
1
2
3
5
6
9

7.3 – Comando WHERE

- As consultas mostradas até agora, com exceção da cláusula DISTINCT, recuperam todos os dados da tabela.

- Consultas que recuperam todos os dados de uma tabela geram uma carga pesada no banco de dados. Portanto, este tipo de consulta deve ser feita somente quando necessário.

- Na maioria das vezes, as consultas deverão filtrar determinados dados da tabela de acordo com as necessidades do programador.

- A cláusula **WHERE** é o mecanismo de filtragem de registros de uma tabela, fazendo com que apenas as linhas desejadas sejam retornadas.

4.3 – Comando WHERE

- A sintaxe de um comando SELECT com a cláusula WHERE possui a sintaxe do quadro abaixo:

SELECT<campos da tabela>**FROM**<nome da tabela>**WHERE**<condição>;

Onde:

<campos da tabela>: Nome dos campos que deverão ser retornados da consulta. Caso seja colocado o '*' (asterisco), todos os campos da tabela serão retornados.

<nome da tabela>: Nome da tabela cuja a consulta será realizada.

<condição>: Condição para retorno dos dados.

7.3 – Comando WHERE

- Para elaboração das condições podem ser usados quaisquer um dos operadores de comparação da tabela abaixo:

Sinal	Descrição
=	igual
<>	diferente
<=	menor ou igual
>=	maior ou igual
>	maior
<	menor

7.3 – Comando WHERE

- Vamos supor que seja necessário mostrar apenas os clientes que residem na ‘Rua Brasil, 52’.

  Para isto, será necessário o uso do comando **WHERE** com a condição em específico:

```
SELECT * FROM cliente WHERE endereco=’Rua Brasil, 52’;
```

id_cliente	nome	endereco	id_cidade	telefone
4	Felipe Melo	Rua Brasil, 52	3	23999566878

7.3 – Comando WHERE

- Caso o objetivo fosse recuperar todos os funcionários cujo campo ‘salario_fixo’ seja menor ou igual a 5000, o seguinte comando deveria ser utilizado:

```
SELECT * FROM funcionario WHERE salario_fixo<=5000
```

id_funcionario	nome	data_contratacao	cargo	salario_fixo	bonus	id_departamento	id_gerente
4	Bill Gates	2015-05-11	Programador	5000.00	500.00	3	2
5	Tony Stark	2002-09-21	Técnico	2000.00	1000.00	3	4
6	José da Silva	2009-05-21	Gerente	2000.00	200.00	4	2
7	João da Garrincha	2001-01-22	Vendedor	1000.00	200.00	4	6
8	John Rambo	2001-04-21	Vendedor	1000.00	200.00	4	6
9	Joaquim Fernandez	2004-01-22	Vendedor	1000.00	200.00	4	6
10	Maria da Silva	2003-02-26	Técnico de RH	3000.00	300.00	1	2

7.3 – Comando WHERE

- A linguagem SQL permite que as consultas tenham mais de uma condição, para isso utiliza-se os operadores **OR** e **AND**, como ilustra a tabela abaixo:

Operador	Descrição
OR	Basta que uma das condições separadas pelo operador OR sejam verdadeiras, para que a condição seja verdadeira.
AND	Para que uma condição separada pelo operador AND seja verdadeira, todos os operadores devem ser verdadeiras.

7.3 – Comando WHERE

- Vamos imaginar que se deseja selecionar todos os pedidos que foram feitos em 2017 e que possuem valor inferior a 50000 reais. Nesse caso, devemos utilizar o operador AND, como mostra o comando abaixo:

```
SELECT * FROM pedido WHERE data>='2017-01-01' AND data<='2017-12-31' AND
total<=50000;
```

id_pedido	data	total	id_funcionario	id_cliente
1	2017-05-23	1400.00	7	4
2	2017-02-22	1400.00	7	5
5	2017-02-10	1400.00	8	2
6	2017-05-10	1800.00	8	8

7.3 – Comando WHERE

- Em consultas SQL mais complexas, podem existir dezenas de condições. Por isso, é uma boa prática de programação a inserção de parênteses “( )“ nas condições da consulta, como no exemplo abaixo:

```
SELECT * FROM pedido WHERE ((data>='2017-01-01') AND (data<='2017-12-31') AND (total<=50000));
```

id_pedido	data	total	id_funcionario	id_cliente
1	2017-05-23	1400.00	7	4
2	2017-02-22	1400.00	7	5
5	2017-02-10	1400.00	8	2
6	2017-05-10	1800.00	8	8

7.3 – Comando WHERE

- Sempre que uma consulta possuir mais de 3 condições e utilizar os **dois operadores** (**AND** e **OR**), o uso do parênteses “( )” é obrigatório, pois ele define a prioridade da expressão.

- O interpretador do banco de dados interpreta primeiro as condições de dentro do parênteses, somente depois interpreta as condições de fora.

- Como exemplo, vamos imaginar que se deseja selecionar todos os pedidos que possuem obrigatoriamente valor superior a 500 reais, e que sejam do ano de 2014 ou do ano de 2017:

```
SELECT * FROM pedido WHERE (total>500) AND (((data>='2014-01-01') AND
(data<='2014-12-31')) OR ((data>='2017-01-01') AND (data<='2017-12-31')));
```

id_pedido	data	total	id_funcionario	id_cliente
1	2017-05-23	1400.00	7	4
2	2017-02-22	1400.00	7	5
5	2017-02-10	1400.00	8	2
6	2017-05-10	1800.00	8	8
7	2014-09-15	1400.00	8	1

7.4 – Comando ORDER BY

- O comando SQL para ordenar os registros de uma seleção é o ORDER BY, que é inserido no final da cláusula SELECT, como ilustra a estrutura abaixo:

SELECT <campos da tabela> **FROM** <nome da tabela> **WHERE** <condição> **ORDER BY** <campo para ordenação>

Onde:

<campo para ordenação>: É o campo da tabela que servirá de base para a ordenação. Esse campo pode ser string, date, integer, etc.

7.4 – Comando ORDER BY

- O exemplo abaixo ilustra uma seleção na tabela de clientes, onde os mesmos são ordenados por nome. Por padrão, os campos são ordenados de forma ascendente (é indiferente colocar o ASC ou não).

SELECT * **FROM** cliente **ORDER BY** nome;

Ou

SELECT * **FROM** cliente **ORDER BY** nome **ASC;**

id_cliente	nome ▲ 1	endereco	id_cidade	telefone
3	Dudu da Silva	Rua Guara, 45	2	44999066878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
10	Felipe Melo	Rua Brasil, 23	3	23999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
9	Jean da Silva	Rua Atlantica, 9	1	77699566878
7	Miguel Borja	Rua João, 40	6	44999566878
5	Roger Guedes	Rua Guedes, 45	9	47988566878
6	William Bigode	Rua 19, 21	5	25999566878
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870

4.4 – Comando ORDER BY

- O exemplo abaixo ilustra uma seleção de todos os clientes, mas ordenados de forma decrescente pela coluna nome:

```
SELECT * FROM cliente ORDER BY nome DESC;
```

id_cliente	nome ▾ 1	endereco	id_cidade	telefone
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
6	William Bigode	Rua 19, 21	5	25999566878
5	Roger Guedes	Rua Guedes, 45	9	47988566878
7	Miguel Borja	Rua João, 40	6	44999566878
9	Jean da Silva	Rua Atlantica, 9	1	77699566878
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
4	Felipe Melo	Rua Brasil, 52	3	23999566878
10	Felipe Melo	Rua Brasil, 23	3	23999566878
3	Dudu da Silva	Rua Guara, 45	2	44999066878

7.4 – Comando ORDER BY

- Também é possível ordenar usando como base mais de um campo, ou seja, caso um campo tenha valores repetidos, o segundo campo é usado como critério de desempate para ordenação.

```
SELECT * FROM cliente ORDER BY nome DESC, endereco ASC;
```

id_cliente	nome 1	endereco 2	id_cidade	telefone
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
6	William Bigode	Rua 19, 21	5	25999566878
5	Roger Guedes	Rua Guedes, 45	9	47988566878
7	Miguel Borja	Rua João, 40	6	44999566878
9	Jean da Silva	Rua Atlantica, 9	1	77699566878
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
10	Felipe Melo	Rua Brasil, 23	3	23999566878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
3	Dudu da Silva	Rua Guara, 45	2	44999066878

7.4 – Comando ORDER BY

- Ao invés do nome dos campos, é possível inserir o número da ordem que ele é declarado ou que está inserido na tabela, como no exemplo abaixo:

```
SELECT * FROM cliente ORDER BY 2 DESC, 3 ASC;
```

id_cliente	nome 1	endereco 2	id_cidade	telefone
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
6	William Bigode	Rua 19, 21	5	25999566878
5	Roger Guedes	Rua Guedes, 45	9	47988566878
7	Miguel Borja	Rua João, 40	6	44999566878
9	Jean da Silva	Rua Atlantica, 9	1	77699566878
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
10	Felipe Melo	Rua Brasil, 23	3	23999566878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
3	Dudu da Silva	Rua Guara, 45	2	44999066878

7.5 – Comando IN

- Quando é necessário que um filtro com diversos valores seja aplicado em uma determinada coluna, é possível a utilização do comando IN. O exemplo abaixo mostra uma consulta que retorna todos os funcionários que possuem salário igual a 500, 1000 e 2000:

```
SELECT * FROM funcionario WHERE salario_fixo IN (500,1000,2000) ORDER BY nome;
```

id_funcionario	nome ▲ 1	data_contratacao	cargo	salario_fixo	bonus	id_departamento	id_gerente
7	João da Garrincha	2001-01-22	Vendedor	1000.00	200.00	4	6
9	Joaquim Fernandez	2004-01-22	Vendedor	1000.00	200.00	4	6
8	John Rambo	2001-04-21	Vendedor	1000.00	200.00	4	6
6	José da Silva	2009-05-21	Gerente	2000.00	200.00	4	2
5	Tony Stark	2002-09-21	Técnico	2000.00	1000.00	3	4

7.6 – Comando BETWEEN

- O comando BETWEEN tem a função de selecionar um intervalo de valores. No exemplo abaixo, serão listados todos os pedidos feitos entre os anos de 2014 e 2017:

```
SELECT * FROM pedido WHERE data BETWEEN '2014-01-01' AND '2017-12-31'  ORDER BY data;
```

id_pedido	data ▲ 1	total	id_funcionario	id_cliente
7	2014-09-15	1400.00	8	1
4	2016-02-12	1400.00	7	6
3	2016-03-22	1400.00	7	5
5	2017-02-10	1400.00	8	2
2	2017-02-22	1400.00	7	5
6	2017-05-10	1800.00	8	8
1	2017-05-23	1400.00	7	4

7.7 – Comando NULL e NOT NULL

- O comando NULL é utilizado para selecionar campos que possuem valor nulo. Um campo nulo não é diferente de um campo vazio.

- O exemplo abaixo irá selecionar os clientes que possuem o campo endereco igual a nulo:

```
SELECT * FROM cliente WHERE endereco IS NULL;
```

id_cliente	nome	endereco	id_cidade	telefone
5	Roger Guedes	*NULL*	*NULL*	47988566878

7.7 – Comando NULL e NOT NULL

- Como processo inverso, onde se deseja selecionar registros que não são nulos, utiliza-se o comando IS NOT NULL. O exemplo abaixo selecionará todos os clientes que não possuem o campo endereco nulo:

```
SELECT * FROM cliente WHERE endereco IS NOT NULL;
```

id_cliente	nome	endereco	id_cidade	telefone
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
3	Dudu da Silva	Rua Guara, 45	2	44999066878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
6	William Bigode	Rua 19, 21	5	25999566878
7	Miguel Borja	Rua João, 40	6	44999566878
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
9	Jean da Silva	Rua Atlantica, 9	1	77699566878
10	Felipe Melo	Rua Brasil, 23	3	23999566878

7.8 – Comando LIKE

- O comando LIKE é utilizado quando se deseja selecionar registros usando partes de uma string como filtro. Vejamos alguns exemplos de onde pode ser utilizado o comando LIKE:

→ Selecionar todos os clientes que possuem a letra “B” em seu nome
→ Selecionar todos os clientes cujo o nome começa com a letra “P”
→ Selecionar todos os clientes cuja a segunda letra do nome é “R”

- Muitos desenvolvedores utilizam uma igualdade na cláusula WHERE para efetuar buscas no nome, contudo essa técnica funciona somente se o valor do filtro for exatamente igual ao do banco de dados

- No exemplo abaixo serão selecionados somente os clientes que tiverem o nome “Paulo”, ou seja, se tiver algum cliente que se chame “João Paulo”, este não será selecionado.

```
SELECT * FROM cliente WHERE nome='Paulo';
```

7.8 – Comando LIKE

- A sintaxe para utilização do comando LIKE é:

SELECT <campos da tabela> **FROM** <nome da tabela> **WHERE** <campo> **LIKE** <string de filtro> **ORDER BY** <campo para ordenação>

Onde:

<campo>:É o campo da tabela que servirá de base para a filtragem.

<string de filtro>: Texto que será buscado no campo especificado no banco de dados.

7.8 – Comando LIKE

- Para se utilizar o comando LIKE, temos alguns operadores que são usados para definir o tipo de busca, são eles:

Operador	Descrição
%	É um coringa, serve para qualquer quantidade de caracteres.
_	Representa um caractere.

7.8 – Comando LIKE

- A tabela abaixo ilustra a utilização destes caracteres:

Exemplo	Função
NOME **LIKE** 'P%'	Seleciona todos os nomes que começam com a letra “P”, independente do que vem depois.
NOME **LIKE** 'P%O'	Seleciona todos os nomes que começam com a letra “P”e terminam com a letra “O”, independente do que vem entre estes caracteres.
NOME **LIKE** '%P%'	Seleciona todos os nomes que tenham a letra “P” em alguma posição, não importando aonde.
NOME **LIKE** '_A%'	Seleciona todos os nomes que possuem a letra “A” na segunda posição.
NOME **LIKE** '_A_L%'	Seleciona todos os nomes que possuem a letra “A” na segunda posição e a letra “L” na quarta posição.

7.8 – Comando LIKE

- A tabela abaixo ilustra a utilização destes caracteres:

```
SELECT * FROM cliente WHERE nome LIKE '%Silva%';
```

id_cliente	nome	endereco	id_cidade	telefone
3	Dudu da Silva	Rua Guara, 45	2	44999066878
8	Zé Roberto da Silva	Rua XV, 53	9	44599566870
9	Jean da Silva	Rua Atlantica, 9	1	77699566878

7.8 – Comando LIKE

- O exemplo abaixo seleciona todos os clientes que possuem a letra “i” na segunda posição e a letra “a” na última posição do nome:

```
SELECT * FROM cliente WHERE nome LIKE '_I%A';
```

id_cliente	nome	endereco	id_cidade	telefone
7	Miguel Borja	Rua João, 40	6	44999566878

7.9 – Comando LIMIT

- O comando **LIMIT** é utilizado para limitar os resultados que irão aparecer nas consultas.

- Por exemplo, de um conjunto de 1000 clientes selecionados em uma consulta, deseja-se que seja retornado apenas 10 clientes contando a partir do terceiro. Isso somente será possível utilizando o comando **LIMIT** do SQL.

- Sua Sintaxe é:

SELECT <campos da tabela> **FROM** <nome da tabela> **WHERE** <condição> **ORDER BY** <campo de ordenação> **LIMIT** N,M;

Onde

N: Número do registro que se deseja iniciar. Começa a partir do 0(zero).

M: Quantidade de registros que se deseja exibir após o N.

7.9 – Comando LIMIT

Por exemplo, deseja-se selecionar todos os clientes que possuem o campo “id_cliente” menor que 6, ordenados por pelo campo “nome”. A consulta dede trazer somente o segundo, terceiro e quarto nome:

O comando abaixo traz todos os registros que possuem o campo “id_cliente” com valor menor do que 6 ordenados por “nome”:

```
SELECT * FROM cliente WHERE id_cliente<6 ORDER BY nome;
```

id_cliente	nome ▲ 1	endereco	id_cidade	telefone
3	Dudu da Silva	Rua Guara, 45	2	44999066878
4	Felipe Melo	Rua Brasil, 52	3	23999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
1	Gabriel Jesus	Rua Juca, 45	1	44999566878
5	Roger Guedes	*NULL*	*NULL*	47988566878

7.9 – Comando LIMIT

- Entretanto, deseja-se apenas o segundo, terceiro e quarto cliente, por isso utilizaremos a cláusula LIMIT:

```
SELECT * FROM cliente WHERE id_cliente<6 ORDER BY nome LIMIT 1,3;
```

id_cliente	nome ▲ 1	endereco	id_cidade	telefone
4	Felipe Melo	Rua Brasil, 52	3	23999566878
2	Fernando Pras	Rua XV, 55	1	44999564378
1	Gabriel Jesus	Rua Juca, 45	1	44999566878