8 – Introdução

- Não faria muito sentido fazer uma seleção na tabela de clientes sem mostrar campos de outras tabelas (chaves estrangeiras), como o nome da cidade do cliente, estado, entre outros.

- Uma das grandes vantagens de um banco de dado relacional é a possibilidade de cruzar os dados de tabelas diferentes em uma única consulta.

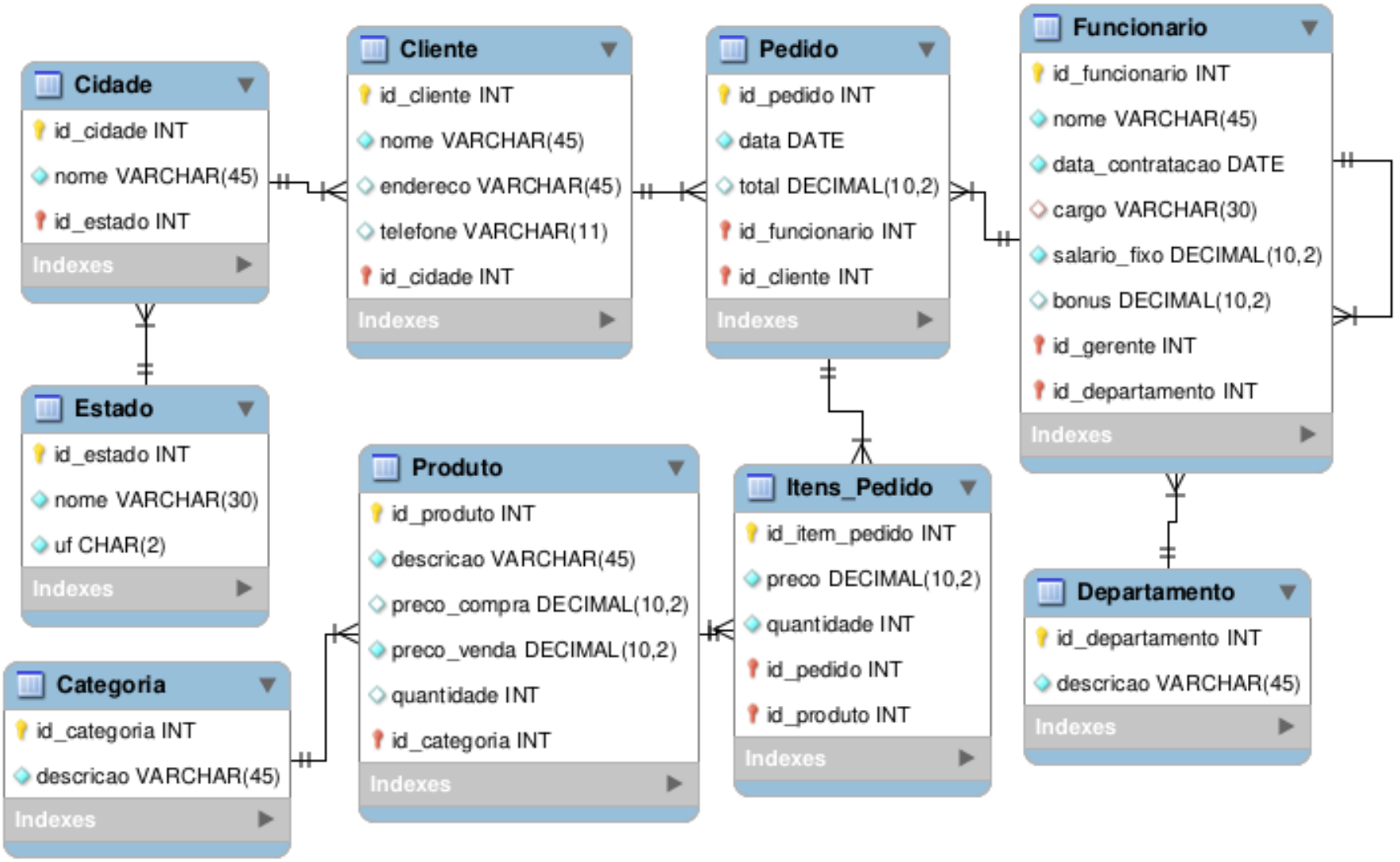
- Desta forma, na mesma consulta, é possível exibir dados tanto de uma tabela de clientes, como sua respectiva cidade, estado, dependentes e todas as demais tabelas relacionadas.

- Essa consulta envolvendo diversas tabelas é feita por meio das chaves primárias e estrangeiras. Essas chaves são as colunas que as tabelas possuem em comum.

- A união entre as tabelas de um banco de dados apresentadas em uma consulta, por meio do comando SELECT, é denominada de **JOIN**.

8.1 – Junção de Tabelas

- Para exemplificar consultas envolvendo diversas tabelas, vamos utilizar o banco de dados n



8.1 – Junção de Tabelas

- Consultas em uma única tabela não são raras, mas a grande maioria das consultas envolverão duas ou mais tabelas.

- Existem diversas formas de se efetuar uma junção de tabelas. O quadro abaixo mostra a sintaxe clássica (ou forma antiga) para junção de tabelas:

SELECT [tabela campo1].<campo1>, [tabela campo2].<campo2>, ... , [tabela campo N].<campoN> **FROM** <tabela1>, <tabela2> ... <tabela N> **WHERE** <condição> **ORDER BY** <campo para ordenação>

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.

8.1 – Junção de Tabelas

- O exemplo abaixo irá mostrar uma consulta na tabela de clientes que retorne o nome, endereço, telefone e o nome da cidade de todos os clientes:

```
SELECT cliente.nome, endereco, telefone, cidade.nome FROM cliente, cidade WHERE
cliente.id_cidade=cidade.id_cidade;
```

nome	endereco	telefone	nome
Gabriel Jesus	Rua Juca, 45	44999566878	Guarapuava
Fernando Pras	Rua XV, 55	44999564378	Guarapuava
Dudu da Silva	Rua Guara, 45	44999066878	Curitiba
Felipe Melo	Rua Brasil, 52	23999566878	Londrina
William Bigode	Rua 19, 21	25999566878	São José dos Campos
Miguel Borja	Rua João, 40	44999566878	São Paulo
Zé Roberto da Silva	Rua XV, 53	44599566870	Florianópolis
Jean da Silva	Rua Atlantica, 9	77699566878	Guarapuava
Felipe Melo	Rua Brasil, 23	23999566878	Londrina

8.1 – Junção de Tabelas

- Outra forma de efetuar junções na linguagem SQL, é por meio do operador JOIN. Esta sintaxe foi definida pelo ANSI92 do padrão ANSI-SQL.

- O quadro abaixo ilustra a sintaxe de junção com o operador JOIN:

SELECT [tabela campo1].<campo1>, [tabela campo2].<campo2> **FROM** <tabela1> **JOIN** <tabela2> **ON** <condição de junção> **WHERE** <condição> **ORDER BY** <campo para ordenação>

8.1 – Junção de Tabelas

- O exemplo abaixo mostra uma consulta na tabela de clientes que retorna o nome, endereço, telefone e o nome da cidade de todos os clientes usando a sintaxe de junção SQL92:

```
SELECT cliente.nome, endereco, telefone, cidade.nome FROM cliente JOIN cidade ON
cliente.id_cidade=cidade.id_cidade;
```

nome	endereco	telefone	nome
Gabriel Jesus	Rua Juca, 45	44999566878	Guarapuava
Fernando Pras	Rua XV, 55	44999564378	Guarapuava
Dudu da Silva	Rua Guara, 45	44999066878	Curitiba
Felipe Melo	Rua Brasil, 52	23999566878	Londrina
William Bigode	Rua 19, 21	25999566878	São José dos Campos
Miguel Borja	Rua João, 40	44999566878	São Paulo
Zé Roberto da Silva	Rua XV, 53	44599566870	Florianópolis
Jean da Silva	Rua Atlantica, 9	77699566878	Guarapuava
Felipe Melo	Rua Brasil, 23	23999566878	Londrina

8.2 – Junção com Filtro

- Mesmo efetuando a junção entre duas ou mais tabelas, também é possível inserir nas consultas condições de filtragem, como no exemplo abaixo, onde deseja-se selecionar o nome e a cidade de todos os clientes que possuem o nome começando com a letra ‘M’:

```
SELECT cliente.nome, endereco, telefone, cidade.nome FROM cliente, cidade WHERE
((cliente.id_cidade=cidade.id_cidade) AND (cliente.nome LIKE ‘M%’)) ORDER BY cliente.nome;
```

nome	endereco	telefone	nome
Miguel Borja	Rua João, 40	44999566878	São Paulo

8.2 – Junção com Filtro

- A mesma consulta na tabela de clientes é feita usando a sintaxe ANSI92. Alguns especialistas preferem esta sintaxe pelo fato dela separar a junção (feita pelo comando JOIN) dos filtros de busca (localizado no WHERE):

SELECT cliente.nome, endereco, telefone, cidade.nome **FROM** cliente **JOIN** cidade **ON** cliente.id_cidade=cidade.id_cidade **WHERE** (cliente.nome **LIKE** 'M%')) **ORDER BY** cliente.nome;

nome	endereco	telefone	nome
Miguel Borja	Rua João, 40	44999566878	São Paulo

8.3 – Junção de Três ou Mais Tabelas

- Junção de três tabelas é semelhante a junção de duas tabelas, havendo pequenas diferenças. O exemplo abaixo mostra uma seleção envolvendo três tabelas (cliente, cidade e estado) usando a sintaxe clássica, onde é possível observar que existirão duas condições de junção:

```
SELECT cliente.nome, endereco, telefone, cidade.nome, uf FROM cliente, cidade, estado WHERE
((cliente.id_cidade=cidade.id_cidade) AND (cidade.id_estado=estado.id_estado) AND
(cliente.nome LIKE 'M%')) ORDER BY cliente.nome;
```

nome	endereco	telefone	nome	uf
Miguel Borja	Rua João, 40	44999566878	São Paulo	SP

8.3 – Junção de Três ou Mais Tabelas

- O quadro abaixo mostra um exemplo de consulta mostrando campos das tabelas de cliente, cidade e estado usando a sintaxe ANSI92:

SELECT cliente.nome, endereco, telefone, cidade.nome, uf **FROM** cliente **JOIN** cidade **ON** cliente.id_cidade=cidade.id_cidade **JOIN** estado **ON** cidade.id_estado=estado.id_estado **WHERE** cliente.nome **LIKE** 'M%' **ORDER BY** cliente.nome;

nome	endereco	telefone	nome	uf
Miguel Borja	Rua João, 40	44999566878	São Paulo	SP

8.4 – Apelidando Tabelas

- Com o objetivo reduzir o tamanho das instruções SQL, é possível criar uma alias (apelido) para as tabelas, como no exemplo abaixo, onde apelidou-se a tabela de cliente de “a”, cidade de “b” e estado de “c”, sendo que o objetivo da consulta é mostrar os clientes que possuem o telefone começando com ‘44’:

```
SELECT a.nome, endereco, telefone, b.nome, uf FROM cliente a JOIN cidade b ON
a.id_cidade=b.id_cidade JOIN estado c  ON b.id_estado=c.id_estado WHERE a.telefone LIKE
‘44%’ ORDER BY a.nome;
```

nome ▲ 1	endereco	telefone	nome	uf
Dudu da Silva	Rua Guara, 45	44999066878	Curitiba	PR
Fernando Pras	Rua XV, 55	44999564378	Guarapuava	PR
Gabriel Jesus	Rua Juca, 45	44999566878	Guarapuava	PR
Miguel Borja	Rua João, 40	44999566878	São Paulo	SP
Zé Roberto da Silva	Rua XV, 53	44599566870	Florianópolis	SC

8.5 – Produto Cartesiano

- O produto cartesiano é a situação em que as linhas da primeira tabela são combinadas com as linhas da segunda tabela, fechando todas as combinações possíveis.

- O produto cartesiano é utilizado em pouquíssimas situações, como quando se quer testar o desempenho do banco, pois os resultados desse tipo de consulta geralmente é muito grande, visto que combina todas as linhas das duas tabelas.

- O produto cartesiano acontece sempre que:

  1. A condição de união das tabelas for omitida

  2. A condição de união entre as tabelas for inválida

8.5 – Produto Cartesiano

- O exemplo abaixo irá ilustrar um exemplo de produto cartesiano. Como é possível observar, não existe nenhuma condição de junção relacionando as duas tabelas, gerando um produto cartesiano:

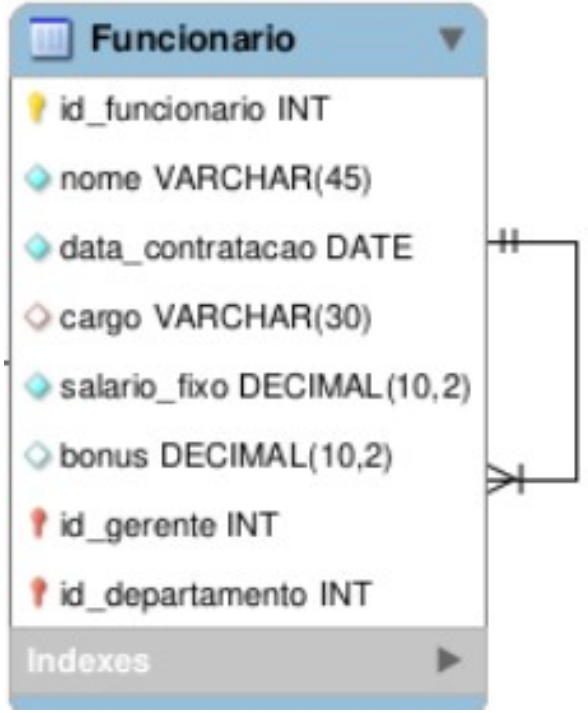
```
SELECT cliente.nome, endereco, telefone, cidade.nome FROM cliente, cidade;
ou
SELECT cliente.nome, endereco, telefone, cidade.nome FROM cliente JOIN cidade;
```

nome	endereco	telefone	nome
Felipe Melo	Rua Brasil, 23	23999566878	Guarapuava
Jean da Silva	Rua Atlantica, 9	77699566878	Guarapuava
Zé Roberto da Silva	Rua XV, 53	44599566870	Guarapuava
Miguel Borja	Rua João, 40	44999566878	Guarapuava
William Bigode	Rua 19, 21	25999566878	Guarapuava
Roger Guedes	*NULL*	47988566878	Guarapuava
Felipe Melo	Rua Brasil, 52	23999566878	Guarapuava
Dudu da Silva	Rua Guara, 45	44999066878	Guarapuava
Fernando Pras	Rua XV, 55	44999564378	Guarapuava
Gabriel Jesus	Rua Juca, 45	44999566878	Guarapuava
Felipe Melo	Rua Brasil, 23	23999566878	Curitiba
Jean da Silva	Rua Atlantica, 9	77699566878	Curitiba
Zé Roberto da Silva	Rua XV, 53	44599566870	Curitiba

8.6 – Auto Junção

- Em determinadas situações pode ser necessário a implementação de um JOIN de uma tabela com ela mesma (auto junção).

- Como exemplo, temos a tabela de funcionário mostrada na figura abaixo, onde existe um auto relacionamento entre os campos ‘id_funcionario’ e ‘id_gerente’ (um gerente é um funcionário):



8.6 – Auto Junção

- Para fazer uma consulta, mostrando todos os empregados juntamente com seu respectivo gerente, será preciso declarar a tabela de empregado mais de uma vez no mesmo código SQL, como mostra o comando abaixo:

```
SELECT func.nome, func.cargo, ger.nome FROM funcionario func, funcionario ger WHERE
func.id_gerente=ger.id_funcionario ORDER BY func.nome;
```

nome ▲ 1	cargo	nome
Bill Gates	Programador	Luigi Bros
João da Garrincha	Vendedor	José da Silva
Joaquim Fernandez	Vendedor	José da Silva
John Rambo	Vendedor	José da Silva
José da Silva	Gerente	Luigi Bros
Luigi Bros	Diretor	Mario Bros
Maria da Silva	Técnico de RH	Luigi Bros
Paulo Nobre	Gerente	Luigi Bros
Tony Stark	Técnico	Bill Gates

8.7 – Tipos de Junções

- O comando abaixo retorna todos os registros da tabela de cidade. Se observarmos com atenção, a cidade de ‘Ponta Grossa’, cujo id_cidade é igual a 11, possui o campo id_estado com valor NULL, ou seja, não existe nenhum estado atribuído a esta cidade:

```
SELECT * FROM cidade;
```

id_cidade	nome	id_estado
1	Guarapuava	1
2	Curitiba	1
3	Londrina	1
4	Maringá	1
5	São José dos Campos	2
6	São Paulo	2
7	Campinas	2
8	São José dos Campos	2
9	Florianópolis	3
10	Blumenau	3
11	Ponta Grossa	*NULL*

8.7 – Tipos de Junções

- Supondo uma situação onde seja necessário um relatório com todas as cidades cadastradas e seus respectivos estados. Neste caso, será preciso que uma operação de JOIN entre as tabelas de cidade e estado seja realizada.

- Mas como é possível observar, a cidade de Ponta Grossa não irá aparecer na consulta, visto que ela não possui um estado correspondente, não atendendo a condição de junção.

```
SELECT id_cidade, cidade.nome, uf FROM cidade JOIN estado
ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;

Ou

SELECT id_cidade, cidade.nome, uf FROM cidade, estado
WHERE cidade.id_estado=estado.id_estado ORDER BY
id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.1 – Junção Interna

- Se um valor existe em uma das tabelas do JOIN, mas não existe na outra, este valor será excluído do resultado da busca. Esse tipo de junção é conhecida como junção interna, e consiste no tipo mais comum de junção. Os exemplos de junção que fizemos até agora são todos internos:

```
SELECT id_cidade, cidade.nome, uf FROM cidade JOIN estado ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;

SELECT id_cidade, cidade.nome, uf FROM cidade, estado WHERE cidade.id_estado=estado.id_estado ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.1 – Junção Interna

- Se fizermos uma junção sem especificar seu tipo, ela será por padrão junção interna. Entretanto, para deixar o código mais claro e facilitar futuras manutenções, o ideal é que se deixe explicito o tipo de junção executada. Para junção interna, pode-se utilizar o comando **INNER JOIN**. Se utilizarmos somente o **JOIN**, este por padrão também será interno, como ilustram os exemplos abaixo:

```
SELECT id_cidade, cidade.nome, uf FROM cidade JOIN estado ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;

SELECT id_cidade, cidade.nome, uf FROM cidade INNER JOIN estado ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.1.1 – Cláusula USING

- Em uma junção, quando as chaves primárias e estrangeiras possuem o mesmo nome, é possível abreviar o comando SQL, usando o comando **USING** seguido do nome do campo utilizado para a junção, como mostra o exemplo abaixo, onde na tabela de cidade e de estado, o campo de junção é id_estado:

```
SELECT id_cidade, cidade.nome, uf FROM cidade INNER JOIN estado USING(id_estado)
ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.2 – Junção Externa

- As junções externas incluem todas as linhas de uma das tabelas e inclui os dados da segunda tabela apenas quando linhas correspondentes são encontradas.

- No caso da tabela de cidade, caso fosse necessário uma consulta que retornasse todas as cidades, não seria possível a utilização de uma junção interna, visto que existem cidades que não possuem um estado correspondente. Desta forma, estas cidades ficariam de fora do resultado, não cumprindo com o objetivo real da consulta, que seria mostrar todas as cidades.

```
SELECT id_cidade, cidade.nome, uf FROM cidade INNER JOIN estado ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.2 – Junção Externa

- As junções externas incluem todas as linhas de uma das tabelas e inclui os dados da segunda tabela apenas quando linhas correspondentes são encontradas.

- No caso da tabela de cidade, caso fosse necessário uma consulta que retornasse todas as cidades, não seria possível a utilização de uma junção interna, visto que existem cidades que não possuem um estado correspondente. Desta forma, estas cidades ficariam de fora do resultado, não cumprindo com o objetivo real da consulta, que seria mostrar todas as cidades.

- Para esses casos, é necessário o uso das junções externas, que permitem mostrar registros mesmo que estes não possuam valores correspondentes com as demais tabelas durante as operações de JOIN. Existem 3 tipos de junções externas, são elas:

  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

8.7.2.1 – LEFT OUTER JOIN

- Uma junção com o operador **LEFT OUTER JOIN** indica que a tabela no lado esquerdo da junção será responsável por determinar o número de linhas do resultado, enquanto a tabela do lado direito é usada para fornecer os valores de uma coluna sempre que uma correspondência é encontrada.

- A consulta abaixo deseja mostrar todas as cidades, com seu respectivo estado (caso possua estado cadastrado), e como é possível observar a cidade de Ponta Grossa é exibida na consulta, mesmo sem ter sido atribuído um estado a ela:

```
SELECT id_cidade, cidade.nome, uf FROM cidade LEFT OUTER JOIN estado ON cidade.id_estado=estado.id_estado ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC
11	Ponta Grossa	*NULL*

8.7.2.2 – RIGHT OUTER JOIN

- Uma junção com o operador **RIGHT OUTER JOIN** indica que a tabela no lado direito da junção será responsável por determinar o número de linhas do resultado, enquanto a tabela do lado esquerdo é usada para fornecer os valores de uma coluna sempre que uma correspondência é encontrada.

- A consulta abaixo deseja mostrar todos os estados com suas respectivas cidades, entretanto, mesmo que um determinado estado não possua nenhuma cidade atribuída, ele será exibido no resultado (no caso temos o estado do RJ):

```
SELECT id_cidade, cidade.nome, uf FROM cidade RIGHT
OUTER JOIN estado ON cidade.id_estado=estado.id_estado
ORDER BY id_cidade;
```

id_cidade ▲ 1	nome	uf
NULL	*NULL*	RS
1	Guarapuava	PR
2	Curitiba	PR
3	Londrina	PR
4	Maringá	PR
5	São José dos Campos	SP
6	São Paulo	SP
7	Campinas	SP
8	São José dos Campos	SP
9	Florianópolis	SC
10	Blumenau	SC

8.7.2.3 – Junções Externas entre 3 Tabelas

- Junções externas podem ser feitas entre várias tabelas (não existe uma limitação específica). No exemplo abaixo, temos junção externa entre as tabelas de cliente, cidade e estado. Desta forma, todos os clientes (tendo cidade atribuída ou não), todas as cidades (tendo estado atribuído ou não) e os estados que possuam cidades correspondentes serão exibidos:

```
SELECT cliente.nome, endereco, cidade.nome, uf FROM cliente LEFT OUTER JOIN
cidade USING(id_cidade) LEFT OUTER JOIN estado USING(id_estado) ORDER BY
cliente.nome;
```

nome ▲ 1	endereco	nome	uf
Dudu da Silva	Rua Guara, 45	Curitiba	PR
Felipe Melo	Rua Brasil, 52	Londrina	PR
Felipe Melo	Rua Brasil, 23	Londrina	PR
Fernando Pras	Rua XV, 55	Guarapuava	PR
Gabriel Jesus	Rua Juca, 45	Guarapuava	PR
Jean da Silva	Rua Atlantica, 9	Guarapuava	PR
Miguel Borja	Rua João, 40	São Paulo	SP
Roger Guedes	*NULL*	*NULL*	*NULL*
William Bigode	Rua 19, 21	São José dos Campos	SP
Zé Roberto da Silva	Rua XV, 53	Florianópolis	SC

8.7.3 – Junções Cruzadas – CROSS JOIN

- O **CROSS JOIN** é semelhante ao produto cartesiano gerado quando não é inserido a uma consulta as condições de junção. Este tipo de junção faz o cruzamento de cada uma das linhas da primeira tabela todas as linhas da segunda tabela, como ilustra o exemplo abaixo:

```
SELECT cidade.nome, uf FROM cidade CROSS JOIN estado;
```

nome	uf
Guarapuava	RS
Guarapuava	SC
Guarapuava	SP
Guarapuava	PR
Curitiba	RS
Curitiba	SC
Curitiba	SP
Curitiba	PR
Londrina	RS
Londrina	SC
Londrina	SP
Londrina	PR
Maringá	RS
Maringá	SC
Maringá	SP
Maringá	PR
São José dos Campos	RS
São José dos Campos	SC

8.8 – Exemplos Junções

- A consulta abaixo faz a seleção de todos os pedidos feitos no ano de 2017. É retornado o id_pedido, nome do cliente, nome do vendedor e nome do gerente do vendedor:

```
SELECT id_pedido, data, cliente.nome as 'Cliente', vend.nome as 'Vendedor', ger.nome as 'Gerente' FROM pedido JOIN cliente USING(id_cliente) JOIN funcionario vend USING(id_funcionario) JOIN funcionario ger ON vend.id_gerente=ger.id_funcionario WHERE data BETWEEN '2017-01-01' and '2017-12-31';
```

id_pedido	data	Cliente	Vendedor	Gerente
1	2017-05-23	Felipe Melo	João da Garrincha	José da Silva
2	2017-02-22	Roger Guedes	João da Garrincha	José da Silva
5	2017-02-10	Fernando Pras	John Rambo	José da Silva
6	2017-05-10	Zé Roberto da Silva	John Rambo	José da Silva

8.8 – Exemplos Junções

- A consulta abaixo faz a seleção de todos os pedidos feitos por clientes que residem no Paraná. É retornado o id_pedido, data, nome do cliente e uf:

```
SELECT id_pedido, data, cliente.nome, uf FROM pedido join cliente USING(id_cliente)
JOIN cidade USING(id_cidade) JOIN estado USING(id_estado) WHERE uf='PR' ORDER
BY data;
```

id_pedido	data ▲ 1	nome	uf
7	2014-09-15	Gabriel Jesus	PR
5	2017-02-10	Fernando Pras	PR
1	2017-05-23	Felipe Melo	PR

8.8 – Exemplos Junções

- A consulta abaixo faz a seleção de todos os pedidos feitos no ano de 2017, por clientes que residem no Paraná e que possuem algum produto da Intel nos seus itens. É retornado o id_pedido, data, nome do cliente, uf, nome do vendedor e nome do gerente do vendedor:

```
SELECT id_pedido, data, cliente.nome as 'Cliente', vend.nome as 'Vendedor', ger.nome as 'Gerente' FROM pedido JOIN cliente USING(id_cliente) JOIN cidade USING(id_cidade) JOIN estado USING(id_estado) JOIN funcionario vend USING(id_funcionario) JOIN funcionario ger ON vend.id_gerente=ger.id_funcionario JOIN itens_pedido USING(id_pedido) JOIN produto USING(id_produto) WHERE data BETWEEN '2017-01-01' and '2017-12-31' and uf='PR' and descricao LIKE '%intel%' ORDER BY data;
```

id_pedido	data ▲ 1	Cliente	Vendedor	Gerente
5	2017-02-10	Fernando Pras	John Rambo	José da Silva
1	2017-05-23	Felipe Melo	João da Garrincha	José da Silva