Universidade POSITIVO

SQL DQL: JOIN
JUNÇÃO DE TABELAS

BANCO DE DADOS

Prof. Luciano Xiscatti

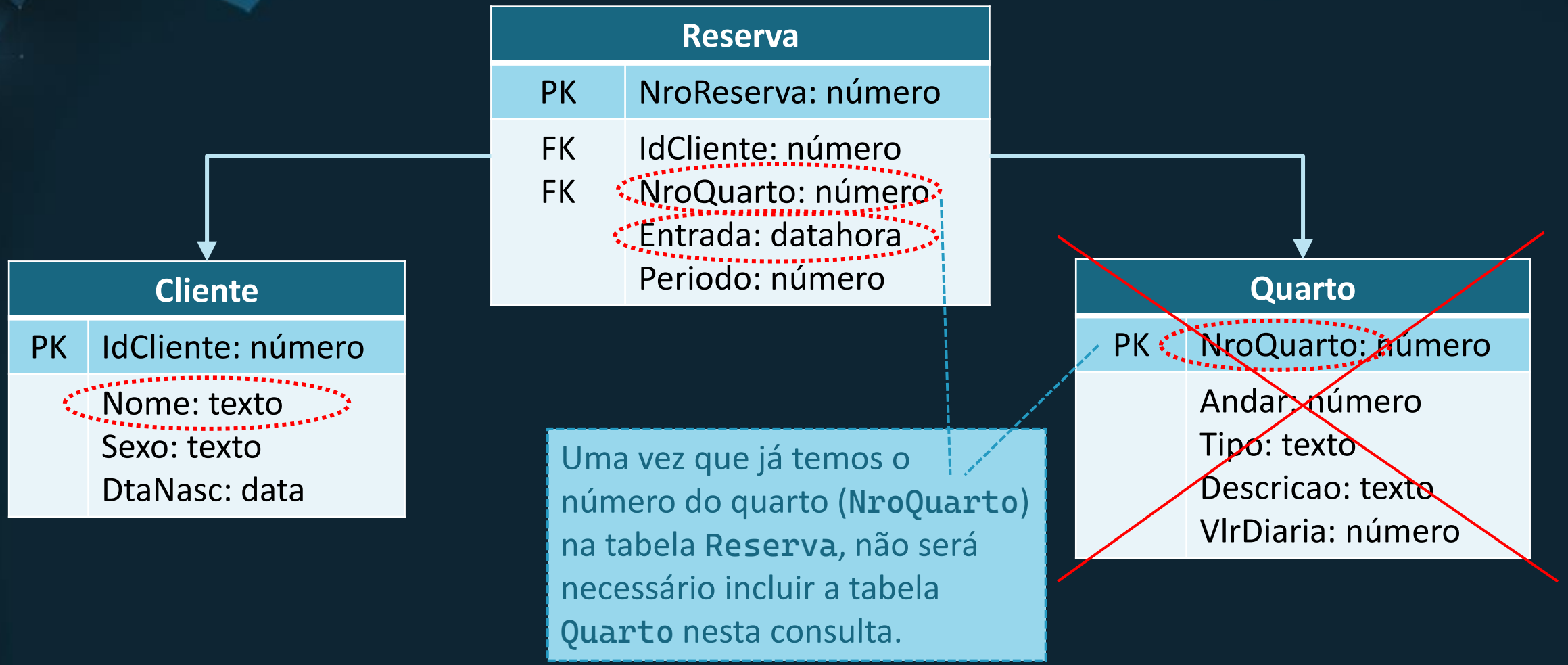
JUNÇÃO DE TABELAS

- Utilizando um único comando SELECT, é possível trazer dados de mais de uma tabela;
- Isso é necessário quando precisamos listar, em um mesmo relatório, dados que estão em tabelas diferentes.
- Por exemplo: “Listar no nome do cliente e o número do quarto de todas as reservas com entrada hoje.”

JUNÇÃO DE TABELAS

- Para atender ao relatório solicitado:
  - ✓ O nome do cliente (`Nome`) só existe na tabela `Cliente`;
  - ✓ O número do quarto (`NroQuarto`) existe tanto na tabela `Quarto` quanto na tabela `Reserva`;
  - ✓ A data de entrada (`Entrada`) só existe na tabela `Reserva`.
  - ✓ Sendo assim, para gerar os dados a serem listados, será necessário juntar registros da tabela `Cliente` e `Reserva`.

JUNÇÃO DE TABELAS



JUNÇÃO DE TABELAS

- Agora vejamos o possível comando SELECT que irá resolver a consulta:

```
SELECT Nome, NroQuarto
FROM Cliente, Reserva
WHERE Entrada = CURDATE();
```

- O comando apresentado tem, incialmente, um problema, além de algumas particularidades, que serão explorados a seguir.

JUNÇÃO DE TABELAS

- Imagine que as tabelas em questão tenham os seguintes dados:

Cliente

IdCliente	Nome	Sexo	DtaNasc
1	Evandro Zatti	M	1976-01-22
2	Augusta Ada Byron King	F	1815-12-10
3	Charles Babbage	M	1791-12-26

Supondo que hoje seja 09/10/2023, estes são os únicos registros de `Reserva` com `Entrada` hoje

Reserva

NroReserva	IdCliente	NroQuarto	Entrada	Período
1	1	11	2023-10-09	10
2	2	22	2023-10-09	7

PRODUTO CARTESIANO

- Como não foi passado nenhum parâmetro (campo) que estabeleça uma relação entre as duas tabelas, o comando `SELECT` irá fazer o que chamamos de produto cartesiano das duas tabelas, isto é, será feita uma multiplicação de todos os registros de uma tabela por todos os da outra tabela (considerando o filtro da cláusula `WHERE`).
  Observe o resultado:

PRODUTO CARTESIANO

- A seguir é mostrado o que seria o resultado do comando apresentado anteriormente, considerando os dados sugeridos:

Nome	NroQuarto
Evandro Zatti	11
Evandro Zatti	22
Augusta Ada Byron King	11
Augusta Ada Byron King	22
Charles Babbage	11
Charles Babbage	22

Observe que foi feita uma "multiplicação" entre as duas tabelas, isto é, uma combinação (produto cartesiano) de todos os **nomes** (`Nome`) da tabela `Cliente` com todos **números de quarto** (`NroQuarto`) da tabela `Reserva` (atendendo apenas ao filtro da `Entrada` com a data de hoje); inclusive veio o nome do Charles Babbage, que não possuía nenhuma reserva.

PRODUTO CARTESIANO

- Observe o resultado com todos os campos de ambas as tabelas:

IdCliente	Nome	Sexo	DtaNasc
1	Evandro Zatti	M	1976-01-22
2	Augusta Ada Byron King	F	1815-12-10
3	Charles Babbage	M	1791-12-26

NroReserva	IdCliente	NroQuarto	Entrada	Período
1	1	11	2023-10-09	10
2	2	22	2023-10-09	7

IdCliente	Nome	Sexo	DtaNasc	NroReserva	IdCliente	NroQuarto	Entrada	Periodo
1	Evandro Zatti	M	1976-01-22	1	1	11	2023-10-09	10
1	Evandro Zatti	M	1976-01-22	2	2	22	2023-10-09	7
2	Augusta Ada Byron King	F	1815-12-10	1	1	11	2023-10-09	10
2	Augusta Ada Byron King	F	1815-12-10	2	2	22	2023-10-09	7
3	Charles Babbage	M	1791-12-26	1	1	11	2023-10-09	10
3	Charles Babbage	M	1791-12-26	2	2	22	2023-10-09	7

HIERARQUIA TABELA.CAMPO

- Você deve ter observado que no resultado anterior, tanto a tabela `Cliente` quanto a tabela `Reserva` possuem o campo `IdCliente`;
- Para que não haja ambiguidade (que em alguns casos pode resultar em erro), ao fazer junção de tabelas, sempre forneça qual a tabela do campo que está sendo usado, fazendo uso da hierarquia: `Tabela.Campo`
- Veja a seguir:

HIERARQUIA TABELA.CAMPO

- Ao invés de:

```
SELECT Nome, NroQuarto
FROM Cliente, Reserva
WHERE Entrada = CURDATE();
```

- Prefira:

```
SELECT Cliente.Nome, Reserva.NroQuarto
FROM Cliente, Reserva
WHERE Reserva.Entrada = CURDATE();
```

ALIASES (APELIDOS)

- Fazer uso da hierarquia `Tabela.Campo` pode resultar em um comando bastante extenso;
- É possível fazer uso de ***alias*** (apelido), a fim de reduzir a quantidade de caracteres do comando, facilitando a leitura.
- Para tanto, basta colocar o *alias* ao lado do nome da tabela. Observe:

HIERARQUIA TABELA.CAMPO + ALIASES

- Ao invés de:

```
SELECT Cliente.Nome, Reserva.NroQuarto
FROM Cliente, Reserva
WHERE Reserva.Entrada = CURDATE();
```

- Pode-se utilizar:

```
SELECT C.Nome, R.NroQuarto
FROM Cliente C, Reserva R
WHERE R.Entrada = CURDATE();
```

ALIASES EM CAMPOS

- Também é possível atribuir apelidos aos campos:

```
SELECT C.Nome Cliente, R.NroQuarto Quarto
FROM Cliente C, Reserva R
WHERE R.Entrada = CURDATE();
```

- O comando apresentado irá resultar em uma tabela com os campos:

Cliente	Quarto
Evandro Zatti	11
...	...

CHAVE DE JUNÇÃO

- Voltando ao problema do produto cartesiano: ao realizar a junção de tabelas, é necessário fornecer o(s) campo(s) que irá(ão) direcionar a junção, o que normalmente será uma relação de chaves primárias com estrangeiras;
- Existem duas maneiras de estabelecer a junção das tabelas:

JUNÇÃO PELO PREDICADO (WHERE)

- A junção das tabelas `Cliente` e `Reserva` pode ser estabelecida adicionando-se a chave no predicado, da seguinte maneira:

```
SELECT C.Nome, R.NroQuarto
FROM Cliente C, Reserva R
WHERE R.Entrada = CURDATE()
   AND R.IdCliente = C.IdCliente;
```

Neste caso, o nome precisa ser do cliente que fez a reserva, e o campo que estabelece unicamente esta relação (e que existe em ambas as tabelas) é `IdCliente`

JUNÇÃO PELO PREDICADO (WHERE)

- A seguir é mostrado o que seria o resultado do comando apresentado, considerando a junção pelo `IdCliente`:

Nome	NroQuarto
Evandro Zatti	11
Augusta Ada Byron King	22

Observe que neste caso foi respeitada a relação cliente x reserva (qual cliente fez a reserva de qual quarto), sem multiplicar os registros.

JUNÇÃO PELA CLÁUSULA JOIN

- Em consultas mais complexas, é comum que seja necessário juntar mais de duas tabelas, e sob diferentes cláusulas, o que compromete a legibilidade do comando e também o plano de execução da consulta;
- Para separar, do predicado, o que define a chave (ou cláusula) de junção e tornar mais legível o que é, de fato, filtro da consulta, criou-se a cláusula `JOIN`.

JUNÇÃO PELA CLÁUSULA JOIN

- Veja a consulta anterior:

```
SELECT C.Nome, R.NroQuarto
FROM Cliente C, Reserva R
WHERE R.Entrada = CURDATE()
   AND R.IdCliente = C.IdCliente;
```

- Como fica com a cláusula JOIN:

```
SELECT C.Nome, R.NroQuarto
FROM Cliente C
INNER JOIN Reserva R
   ON R.IdCliente = C.IdCliente
WHERE R.Entrada = CURDATE();
```

JUNÇÃO PELA CLÁUSULA JOIN

- A variação da cláusula `JOIN` que foi utilizada anteriormente foi o `INNER JOIN`, que apresenta somente os registros de ambas as tabelas que contenham valor na chave de junção;
- Existem outras formas de se fazer junção:
  - ✓ `LEFT JOIN`: todos os registros da tabela da esquerda, mesmo que não tenham correspondência na tabela da direita;
  - ✓ `RIGHT JOIN`: todos os registros da tabela da direita, mesmo que não tenham correspondência na tabela da esquerda.

LEFT JOIN

```
SELECT C.Nome, R.NroQuarto
FROM Cliente C
LEFT JOIN Reserva R
   ON R.IdCliente = C.IdCliente
WHERE R.Entrada = CURDATE();
```

análogo a:

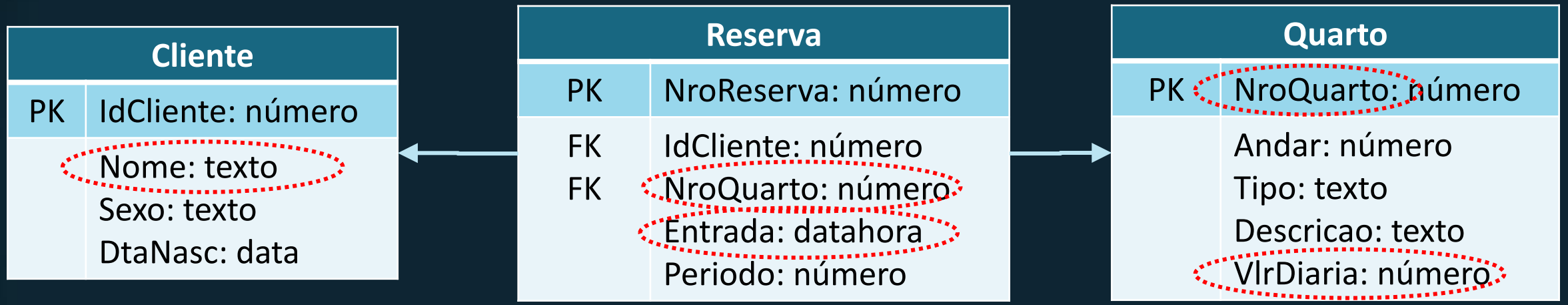
```
SELECT C.Nome, R.NroQuarto
FROM Cliente C, Reserva R
WHERE R.Entrada = CURDATE()
   AND (R.IdCliente = C.IdCliente
      OR R.IdCliente IS NULL);
```

- Traria como resultado:

Nome	NroQuarto
Evandro Zatti	11
Augusta Ada Byron King	22
Charles Babbage	NULL

JUNÇÃO DE MAIS DE DUAS TABELAS

- "Listar no **nome do cliente**, o **número do quarto** e o **valor da diária** de todas as reservas com **entrada** hoje."



JUNÇÃO DE MAIS DE DUAS TABELAS

- "Listar no **nome do cliente**, o **número do quarto** e o **valor da diária** de todas as reservas com **entrada** hoje."

```
SELECT C.Nome, Q.NroQuarto, Q.VlrDiaria
FROM Cliente C
INNER JOIN Reserva R
   ON R.IdCliente = C.IdCliente
INNER JOIN Quarto Q
   ON Q.NroQuarto = R.NroQuarto
WHERE R.Entrada = CURDATE();
```

Universidade POSITIVO

ATIVIDADE PRÁTICA

Prof. Luciano Xiscatti

REFERÊNCIAS

- MySQL. Disponível em https://mysql.com. Acesso em: 26 ago. 2023.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. ***Database System Concepts***. 6th Ed. New York: McGraw-Hill, 2011.