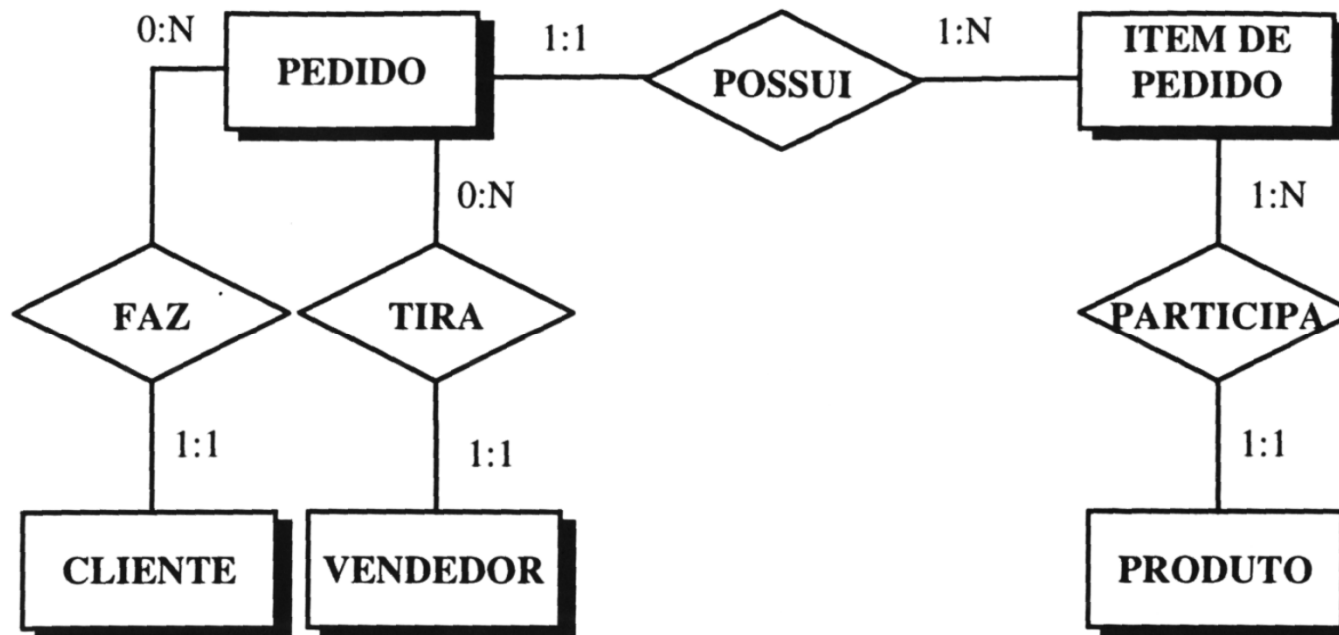




# SQL Avançado

Fabiano Santos de Oliveira

# Exemplo de Modelo de Dados



## Joins (Recuperando dados de várias tabelas)

➤ Existem consultas que necessitam realizar uma junção (JOIN) entre tabelas, para extrair dessa junção as informações necessárias para a consulta formulada.

➤ Qualificador de Nome:

➤ Consiste no nome da tabela seguido de um ponto e o nome da coluna na tabela.

➤ Exemplo: Produto.descrição

# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
SELECT <nome_da_tabela.nome_da_coluna  
[nome_da_tabela.nome_da_coluna... . .]>  
FROM {nome_da_tabela [tipo de join]  
nome_da_tabela ON condição de pesquisa  
WHERE [condição de pesquisa... . . ]
```

## ➤ MySQL

```
SELECT <nome_da_tabela.nome_da_coluna  
[nome_da_tabela.nome_da_coluna... . .]>  
FROM <nome_da_tabela, nome_da_tabela>  
WHERE <nome_da_tabela.nome_da_coluna [operador de join]  
nome_da_tabela.nome_da_coluna
```

# INNER JOIN

- Com o INNER JOIN serão incluídas somente as linhas que satisfazem a condição do join.
- Problema: Ver os pedidos de cada cliente.
- Diagrama gráfico:

CLIENTE
<b>Código_do_cliente</b> ☒
✓ Nome_do_cliente
Endereço
Cidade
CEP
UF
CGC
IE

PEDIDO
✓ <b>Número_do_pedido</b> ☒
Prazo_de_entrega
✓ Código_do_cliente ☐
Código_do_vendedor ☐

# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
SELECT Cliente.nome_cliente,  
        pedido.cod_cliente,  
        pedido.num_pedido  
FROM cliente INNER JOIN pedido  
ON cliente.codigo_do_cliente  
= pedido.codigo_do_cliente
```

## ➤ MySQL

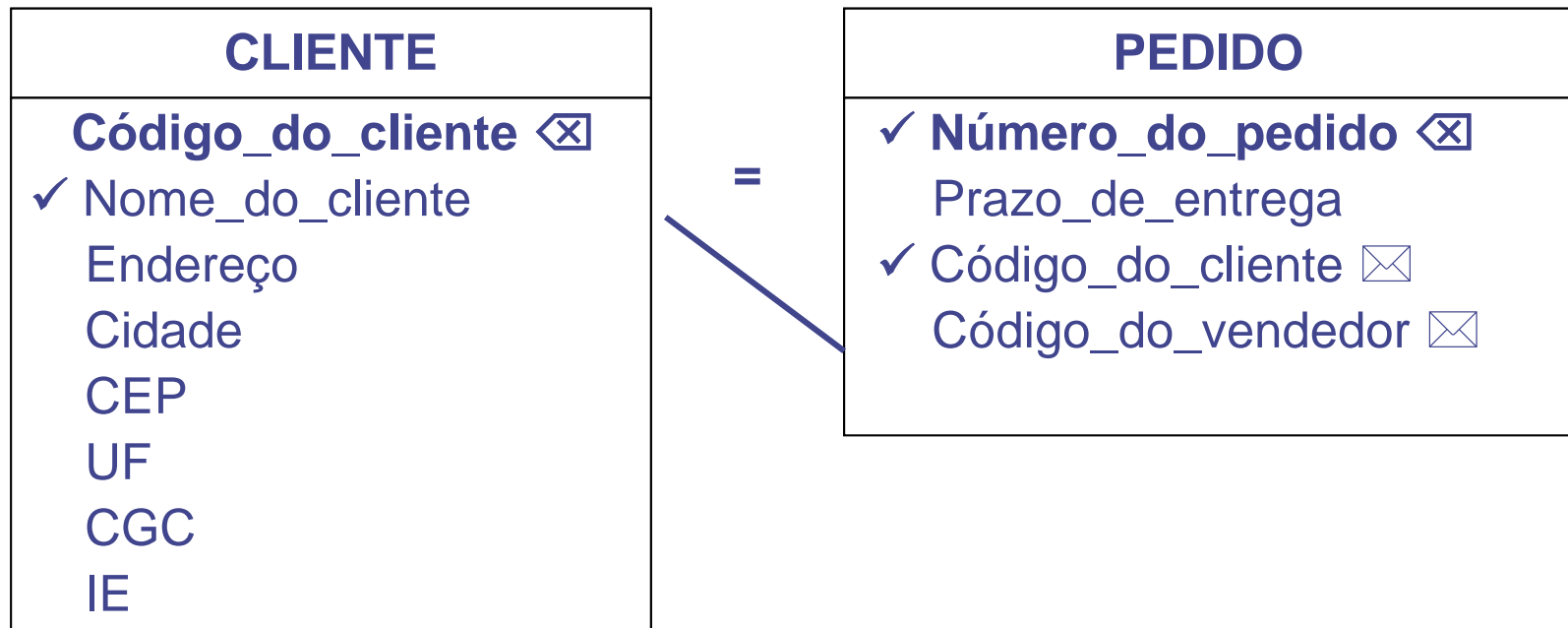
```
SELECT Cliente.nome_cliente,  
        pedido.cod_cliente,  
        pedido.num_pedido  
FROM cliente, pedido  
WHERE cliente.codigo_do_cliente  
= pedido.codigo_do_cliente
```

# Resultado:

Nome_Cliente	Pedido.Codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Maurício	830	203
Rodolfo	410	121
Rodolfo	410	98
Rodolfo	410	127
Beth	20	143
Lívio	180	105
Susana	260	111
Susana	260	103
Susana	260	91
Susana	260	138

# CROSS JOIN ou Produto Cartesiano

- Com o CROSS JOIN serão incluídas cada uma das combinações de todas as linhas entre as tabelas.
- Problema: Juntar Clientes com Pedidos.
- Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
SELECT nome_cliente,  
        pedido.cod_cliente,  
        num_pedido  
FROM cliente CROSS JOIN pedido
```

## ➤ MySQL

```
SELECT nome_cliente,  
        pedido.cod_cliente,  
        num_pedido  
FROM cliente, pedido
```

## Resultado:

Nome_Cliente	Pedido.Codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	260	111
Ana	870	54
Ana	390	119
Flávio	720	97
Flávio	260	111
Flávio	870	54
Flávio	390	119
Jorge	720	97
Jorge	260	111
Jorge	870	54
Jorge	390	119
Lúcia	720	97
Lúcia	260	111
Lúcia	870	54

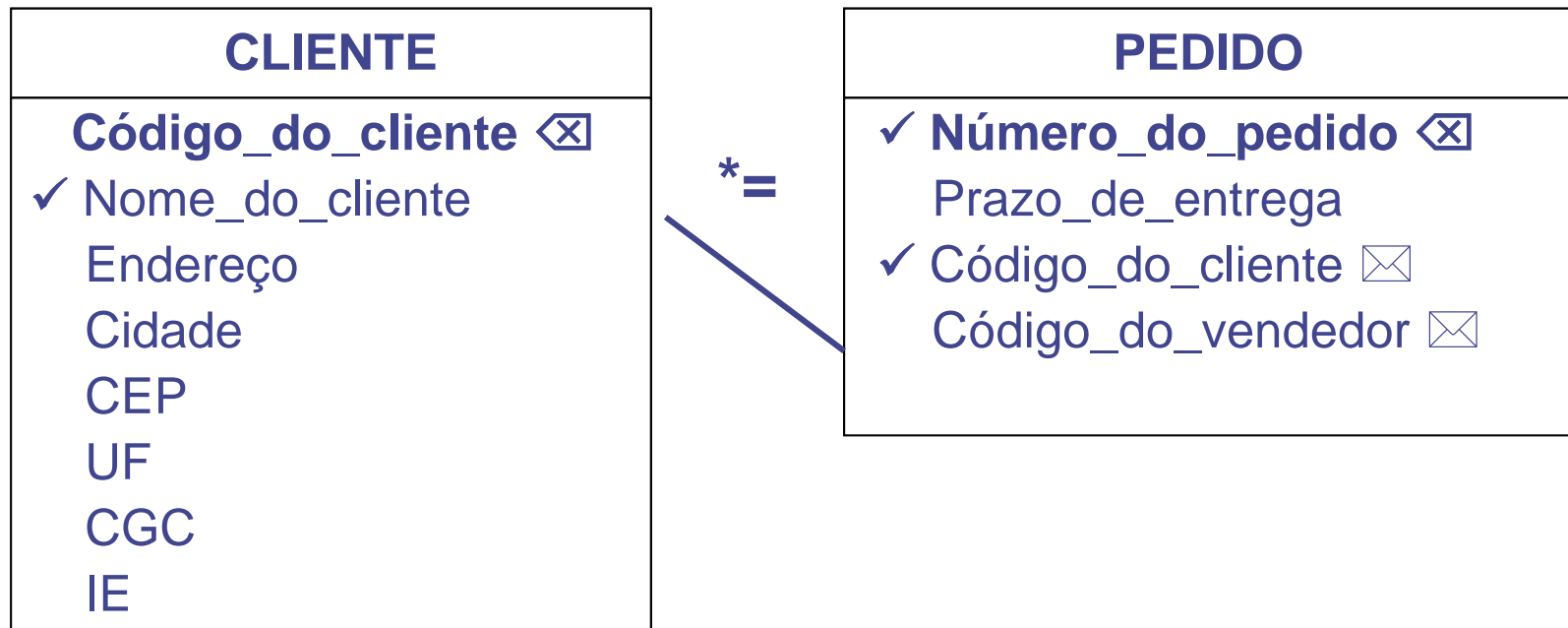
Observa-se que não existe muito proveito do resultado desse tipo de JOIN, excetuando-se quando queremos fazer referência cruzada entre duas tabelas e todas as suas linhas.

# OUTER JOIN

- Seleção em que são restritas as linhas que interessam em uma tabela, mas são consideradas todas as linhas de outra tabela.
- Exemplo: Verificar quais clientes tem pedidos e quais não têm nenhum pedido.
- Pode ser utilizado só entre duas tabelas.
- Possui três tipos de qualificadores:
  - **LEFT OUTER JOIN** – inclui todas as linhas da tabela do primeiro nome de tabela (tabela mais à esquerda da expressão).
  - **RIGHT OUTER JOIN** – inclui todas as linhas da tabela do segundo nome de tabela da expressão.
  - **FULL OUTER JOIN** – inclui as linhas que não satisfazem a expressão tanto da primeira tabela quanto da segunda tabela.

# OUTER JOIN

- Problema: Quais são os clientes que têm pedido e os que não têm pedido.
- Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
SELECT nome_cliente,  
        pedido.cod_cliente,  
        num_pedido  
FROM cliente LEFT OUTER JOIN pedido  
ON cliente.codigo_do_cliente =  
        Pedido.codigo_do_cliente
```

## Resultado:

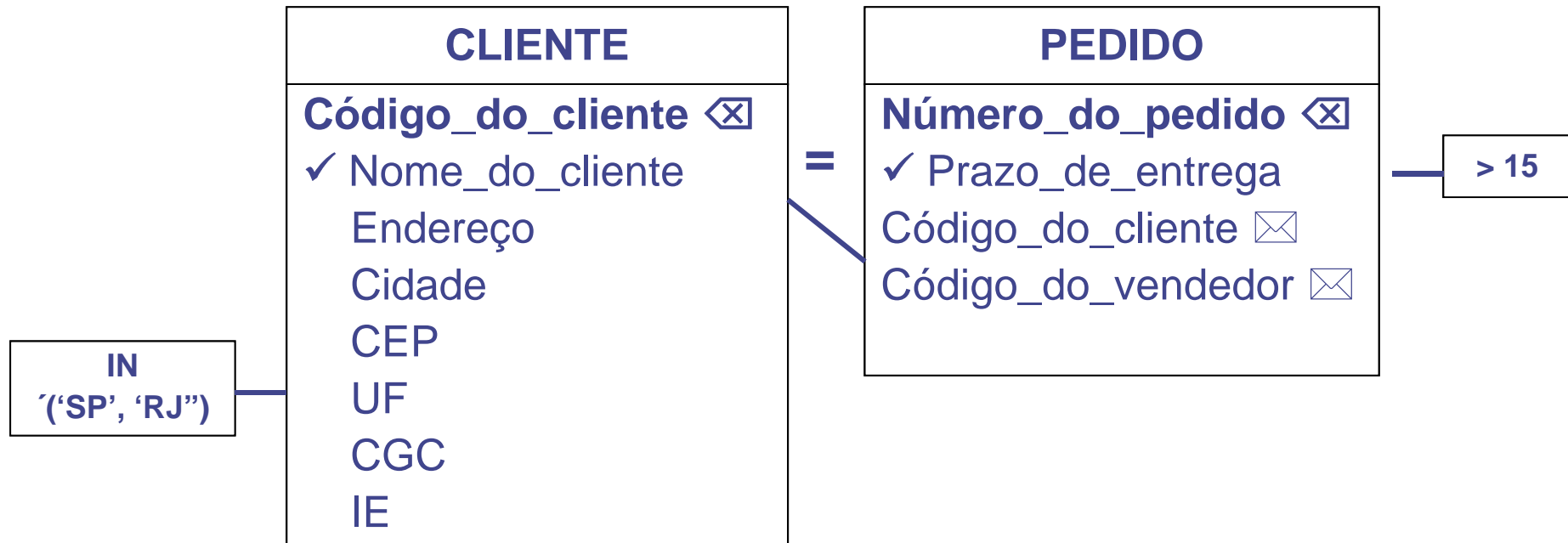
Nome_Cliente	Pedido.Codigo_do_cliente	Pedido.num_pedido
Ana	720	97
Ana	720	101
Ana	720	137
Ana	720	148
Flávio	870	189
Jorge	110	104
Maurício	830	203
Rodolfo	410	121
Rodolfo	410	98
Rodolfo	410	127
Beth	20	143
Lúcia	NULL	NULL
Edmar	NULL	NULL
Paulo	NULL	NULL
José	NULL	NULL

# OUTER JOIN

- Podemos utilizar as cláusulas LIKE, NOT LIKE, IN, NOT IN, NULL, NOT NULL e misturá-las com os operadores AND, OR e NOT, dentro de uma cláusula WHERE na junção entre tabelas.
- Problema: Quais clientes têm prazo de entrega superior a 15 dias e pertencem aos estados de São Paulo ('SP') ou Rio de Janeiro ('RJ')?

# OUTER JOIN

➤ Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

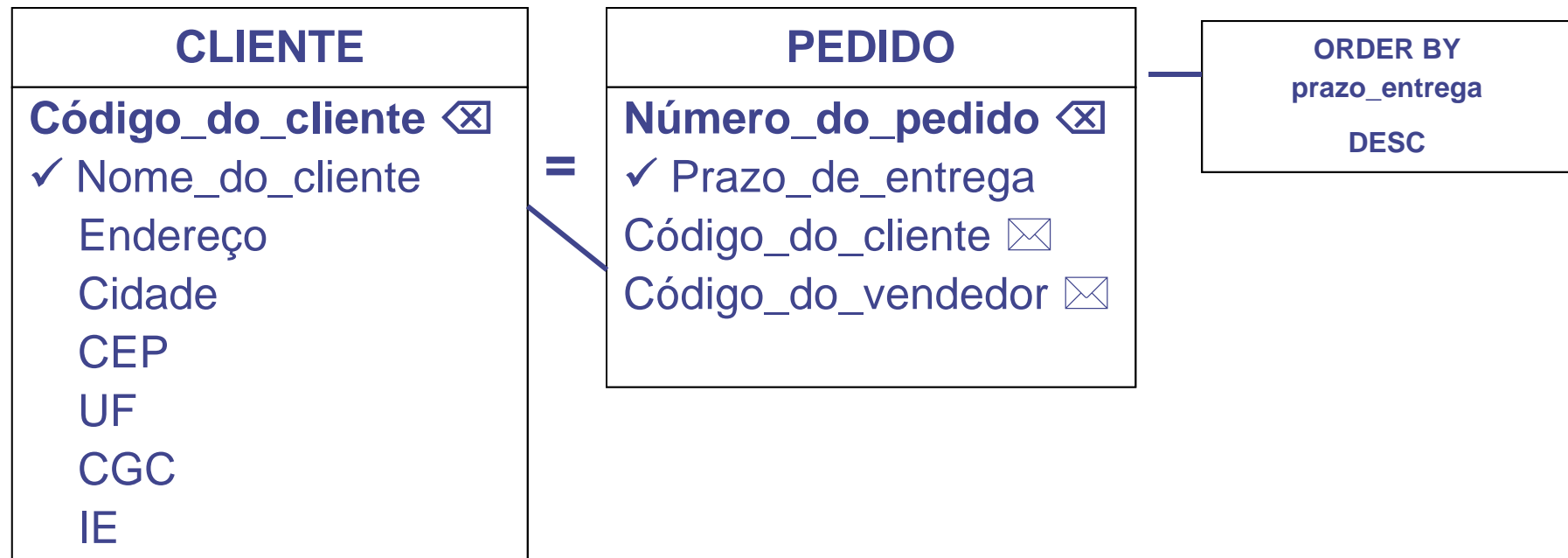
```
SELECT Cliente.nome_cliente,  
        pedido.cod_cliente,  
        pedido.num_pedido  
FROM cliente INNER JOIN pedido  
ON cliente.codigo_do_cliente  
      = Pedido.codigo_do_cliente  
WHERE UF IN ('SP', 'RJ') AND  
prazo_entrega > 15
```

## Resultado:

<b>NOME CLIENTE</b>	<b>UF</b>	<b>PRAZO ENTREGA</b>
Ana	RJ	20
Maurício	SP	30
Rodolfo	RJ	20
Beth	SP	30
Susana	RJ	20

# OUTER JOIN

- Problema: Mostrar os clientes e seus respectivos prazos de entrega, ordenados do maior para o menor.
- Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

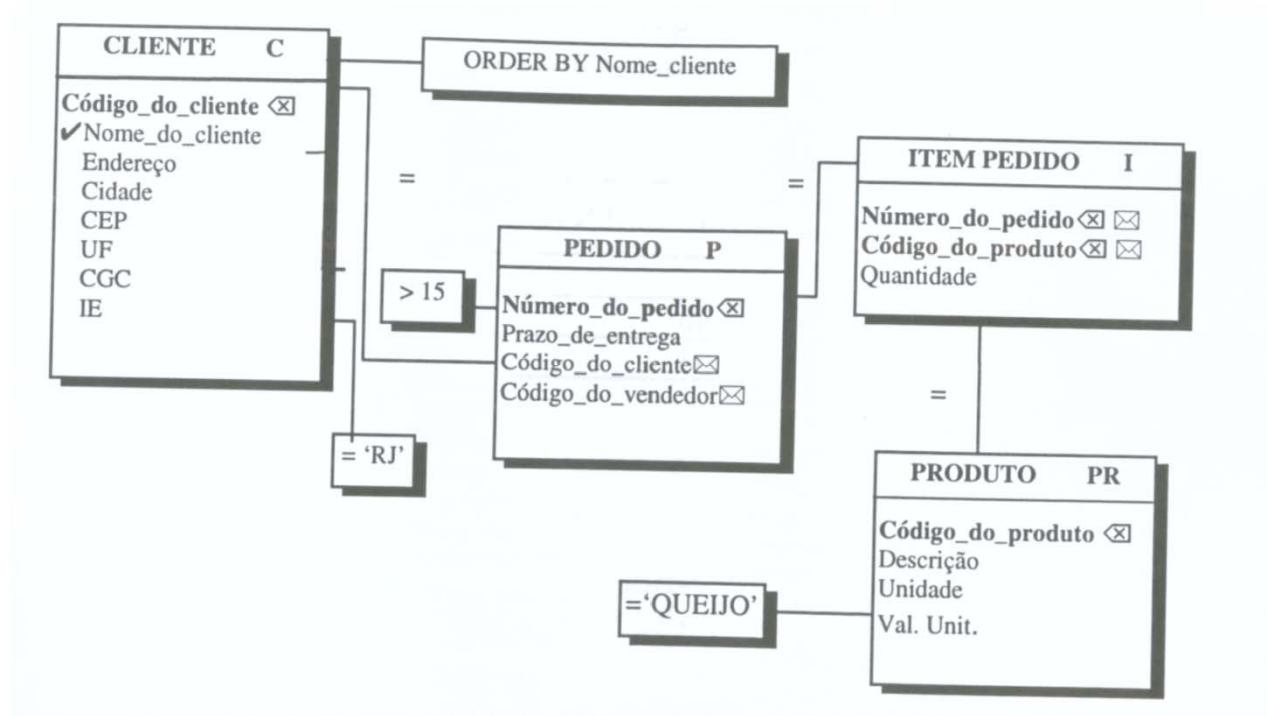
```
SELECT nome_cliente, prazo_entrega  
FROM cliente, pedido  
ON cliente.cod_cliente = pedido.cod_cliente  
ORDER BY prazo_entrega desc;
```

# Resultado:

<b>NOME CLIENTE</b>	<b>PRAZO ENTREGA</b>
Jorge	30
Maurício	30
Beth	30
Sebastião	30
Rodolfo	20
Ana	20
Susana	20
Ana	15
Flávio	15
Lívio	15
Renato	15
Rodolfo	10

# Juntando mais de duas Tabelas

- Problema: Mostre os clientes (ordenados) que têm prazo de entrega maior que 15 dias para o produto 'QUEIJO' e sejam do Rio de Janeiro..
- Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
SELECT Cliente.nome_cliente,  
        FROM cliente INNER JOIN pedido  
ON cliente.codigo_do_cliente = pedido.codigo_do_cliente  
        INNER JOIN item_de_pedido  
ON pedido_num_pedido = item-de_pedido.num_pedido  
        INNER JOIN produto  
ON item_de_pedido.cod_produto=produto.cod_produto  
        WHERE Pedido.prazo_entrega > 15 AND  
        Produto.Descrição = 'queijo' AND  
        Cliente.UF = 'RJ'  
        ORDER BY Cliente.nome_cliente
```

## ➤ MySQL

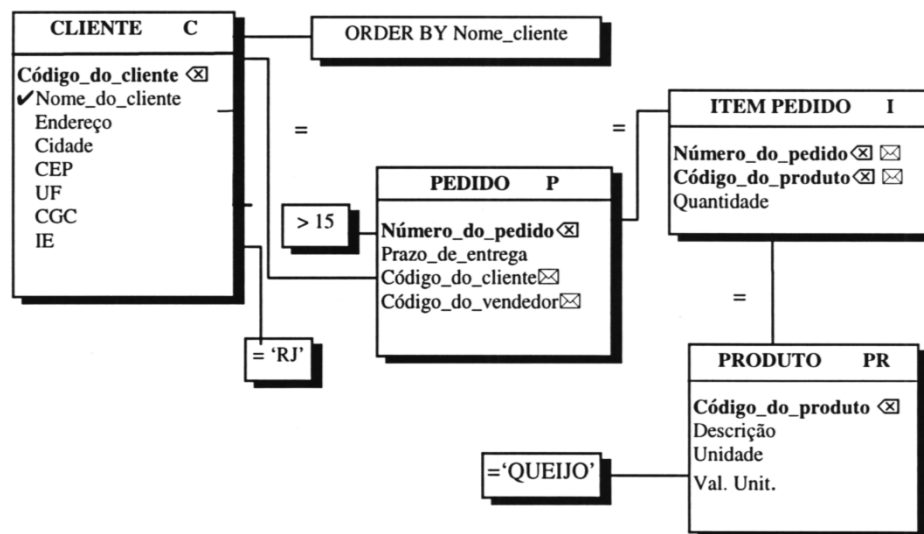
```
SELECT nome_cliente  
FROM cliente, pedido, item_pedido, produto  
WHERE Cliente.cod_cliente = Pedido.cod_cliente  
AND Pedido_num_pedido = Item_de_pedido.num_pedido  
AND Item_de_pedido.cod_produto = Produto.cod_produto  
AND Pedido.prazo_entrega > 15  
AND Produto.Descrição = 'queijo'  
AND Cliente.UF = 'RJ'  
ORDER BY Cliente.nome_cliente;
```

# Resultado:

NOME CLIENTE
Ana
Rodolfo
Susana

# Utilizando Consultas Encadeadas (Subqueries)

- Subquery: quando o resultado de uma consulta é utilizado por outra consulta, de forma encadeada e contida no mesmo comando SQL.
- Problema – **Utilizando IN**
  - Que produtos participam de qualquer pedido cuja quantidade seja 10?
- Diagrama gráfico:



# Sintaxe

```
SELECT descrição  
      FROM produto  
      WHERE cod_produto IN  
      (SELECT cod_produto  
      FROM item_pedido  
      WHERE quantidade = 10)
```

# Resultado:

DESCRIÇÃO
Queijo
Vinho
Linho

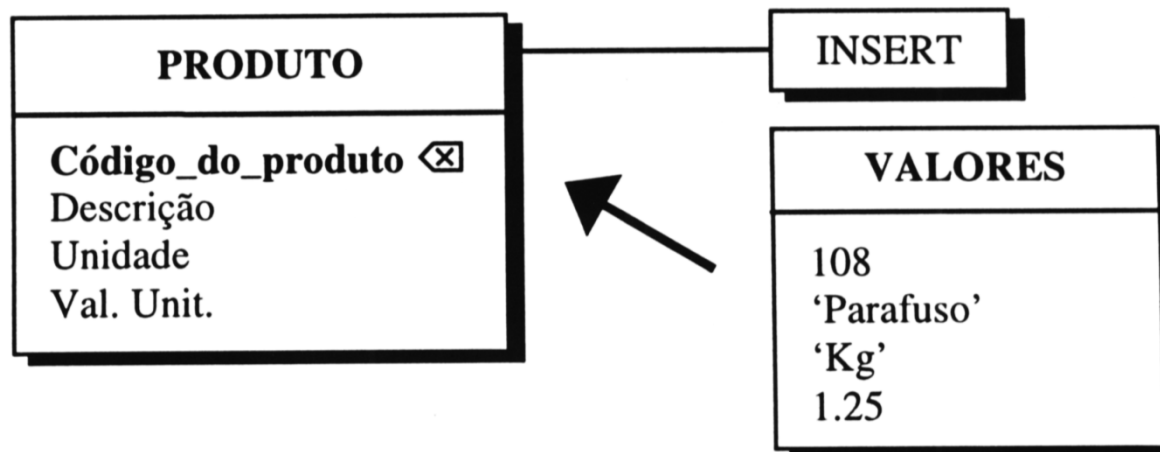
# Adicionando Registro à Tabela

➤ Forma:

```
INSERT INTO <nome da tabela>  
                <nome da(s) coluna(s)>  
VALUES (<valores>);
```

➤ Problema: Adicionar o produto 'parafuso' à tabela produto.

➤ Diagrama gráfico:



# Sintaxe

```
INSERT into produto  
  VALUES (108,  
  'Parafuso',  
  'Kg',  
  1.25);
```

# Adicionando Registros usando SELECT

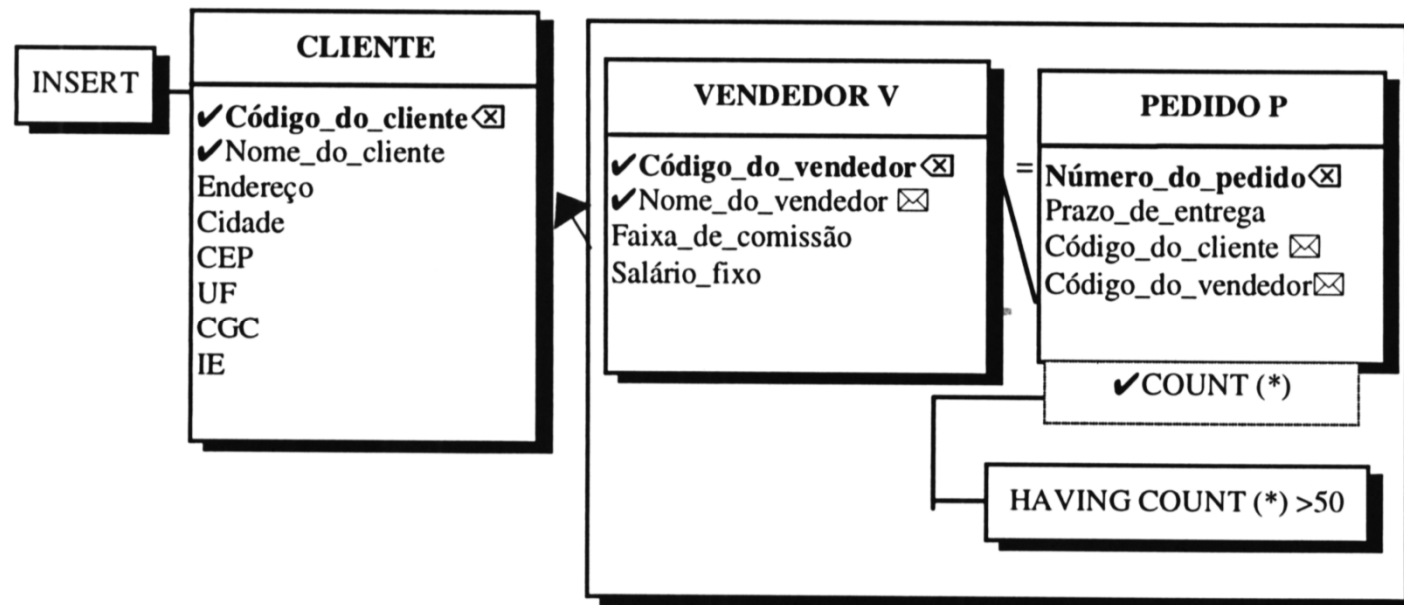
➤ Formato:

```
INSERT INTO <nome da tabela>  
    (<nome da(s) coluna(s)>)  
SELECT <nome da(s) coluna(s)>  
FROM <nome da tabela>  
WHERE <condição>;
```

➤ Problema: Cadastrar como cliente os vendedores que emitiram mais de 50 pedidos. Usar para código de cliente o mesmo código de vendedor.

# Adicionando Registro à Tabela

➤ Diagrama gráfico:



# Sintaxe ANSI SQL e MySQL

## ➤ ANSI SQL

```
INSERT into cliente (cod_cliente, nome_cliente)
SELECT cod_vendedor, nome_vendedor, COUNT(*)
FROM vendedor, pedido
WHERE Vendedor.cod_vendedor = Pedido.cod_vendedor
HAVING COUNT(*) > 50
```

## ➤ MySQL

```
INSERT cliente (cod_cliente, nome_cliente)
SELECT cod_vendedor, nome_vendedor, COUNT(*)
FROM vendedor, pedido
WHERE Vendedor.cod_vendedor = Pedido.cod_vendedor
HAVING COUNT(*) > 50
```

# Exercícios

1. Listar todos os produtos com código, descrições, nome de categoria e preço;
2. Listar da tabela USUÁRIOS o nome do cliente e seu endereço;
3. Listar o CÓDIGO do pedido, o código do produto e a quantidade dos itens do pedido, cuja quantidade seja maior que 25;
4. Quais os usuários que moram em Umuarama;
5. Mostrar os pedidos criados no mês de janeiro de 2011;
6. Listar o código e a descrição dos produtos que tenham o valor unitário na faixa de R\$ 5,00 até R\$ 32,00;
7. Listar todos os produtos que tenham o seu nome começando por “Q”;
8. Mostrar os usuários com email´s com @uol.com.br;
9. Mostrar em ordem alfabética a lista de usuários;

# Exercícios

1. Listar os nomes, cidades e estados de todos os usuários, ordenados por estado e cidade de forma descendente
2. Mostrar a quantidade total pedida para o produto “VINHO” de código 78 na tabela item\_de\_pedido
3. Listar o número de produtos que cada pedido contém
4. Listar os pedidos que têm mais do que três produtos
5. Alterar o valor unitário do produto “parafuso” de R\$ 1.25 para R\$ 1.62
6. Acrescentar 2,5% ao preço unitário dos produtos