

Banco de Dados

Aula 13 - Linguagem SQL – Subconsultas

Apresentação

Na aula anterior, começamos nossos estudos sobre ambientes de banco de dados com múltiplas tabelas, através da especificação de atributos como sendo chave primária e chave estrangeira (PRIMARY KEY e FOREIGN KEY). Em seguida, estudamos o processo de consulta no contexto multitabelas, usando as conexões cartesianas definidas pela cláusula CROSS JOIN.

Nesta aula, você vai aprender como pegar um resultado de uma consulta e usá-lo como entrada para outra consulta, ou seja, irá trabalhar com consultas aninhadas, denominadas subconsultas. A utilização de subconsultas permite realizar consultas mais dinâmicas e evitar dados duplicados.



Vídeo 01 - Apresentação

Objetivos

- Consultar dados em tabelas usando subconsultas.
- Diferenciar subconsultas e conexões.
- Realizar subconsultas com uma coluna na instrução SELECT.
- Diferenciar subconsultas correlacionadas e não correlacionadas.
- Realizar subconsultas usando as cláusulas EXISTS e NOT EXISTS.

Subconsulta

Para você realizar os estudos desta aula, vamos considerar um banco de dados, denominado **sistvendas**, que representa um sistema de controle de vendas de uma determinada loja de departamentos com as seguintes tabelas:

- produtos (codigo_Produto [chave primária], nome, marca e preço);
- clientes (codigo_Clientes [chave primária], nome, CPF, sexo e dataNascimento);
- compras (codigo_Compras [chave primária], codigo_Produto [chave estrangeira], codigo_Clientes [chave estrangeira] e total_de_unidades).

As estruturas e os dados das tabelas **produtos**, **clientes** e **compras** são ilustradas na **Figura 1** e na **Figura 2**. Analise com cuidado essas tabelas e não se esqueça de implementá-las em seu SGBD e inserir dados nelas para posterior utilização, essa é uma ótima maneira de fixar os conceitos aprendidos.

Figura 01 - Tela do MySQL após os comandos DESC **produtos**, DESC **clientes** e DESC **compras**.

```

mysql> USE sistvendas;
Database changed
mysql> DESC produtos;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| prod_codigo | int(11)   | NO   | PRI | NULL    |       |
| prod_nome   | varchar(100) | NO   |     | NULL    |       |
| prod_marca  | varchar(40) | NO   |     | NULL    |       |
| prod_preco  | decimal(9,2) | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

mysql> DESC clientes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cli_codigo  | int(11)   | NO   | PRI | NULL    |       |
| cli_nome    | varchar(50) | NO   |     | NULL    |       |
| cli_CPF     | varchar(11) | NO   |     | NULL    |       |
| cli_sexo    | char(1)   | NO   |     | NULL    |       |
| cli_dataNascimento | datetime | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> DESC compras;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| comp_codigo | int(11)   | NO   | PRI | NULL    |       |
| comp_codproduto | int(11) | NO   | MUL | NULL    |       |
| comp_codcliente | int(11) | NO   | MUL | NULL    |       |
| comp_total  | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

Fonte: MySQL Server 5.7 Command Line Client

Figura 02 - Tela do MySQL mostrando os registros presentes nas tabelas **produtos**, **clientes** e **compras**.

```

mysql> SELECT * FROM produtos;
+-----+-----+-----+-----+
| prod_codigo | prod_nome      | prod_marca | prod_preco |
+-----+-----+-----+-----+
| 1 | Ventilador | ARNO | 90.00 |
| 2 | Celular N97 | NOKIA | 1200.00 |
| 3 | Chocolate sonho de valsa | Lacta | 0.80 |
| 4 | Geladeira | Brastemp | 2000.00 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT * FROM clientes;
+-----+-----+-----+-----+-----+
| cli_codigo | cli_nome      | cli_CPF | cli_sexo | cli_dataNascimento |
+-----+-----+-----+-----+-----+
| 1 | José Josemar | 3252541 | M | 1980-10-03 00:00:00 |
| 2 | Luciana | 4812072 | F | 1972-04-30 00:00:00 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT * FROM compras;
+-----+-----+-----+-----+
| comp_codigo | comp_codproduto | comp_codcliente | comp_total |
+-----+-----+-----+-----+
| 1 | 3 | 2 | 30 |
| 2 | 2 | 2 | 1 |
| 3 | 4 | 1 | 1 |
| 4 | 3 | 1 | 100 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

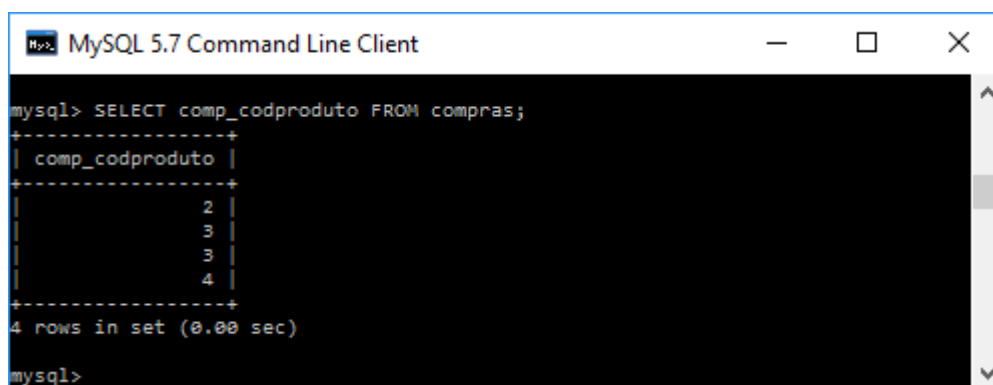
Fonte: MySQL Server 5.7 Command Line Client

Suponha que precisamos saber quais são os códigos dos produtos que foram vendidos. O comando utilizado, conforme visto em aulas anteriores, para realizar essa simples pesquisa é mostrado a seguir.

```
1 mysql> SELECT comp_codproduto
2   FROM compras;
```

A resposta do sistema SGBD a essa pesquisa é 2, 3, 3 e 4, conforme é ilustrado na **Figura 3**.

Figura 03 - Tela do MySQL após o comando SELECT para determinar os códigos dos produtos vendidos.



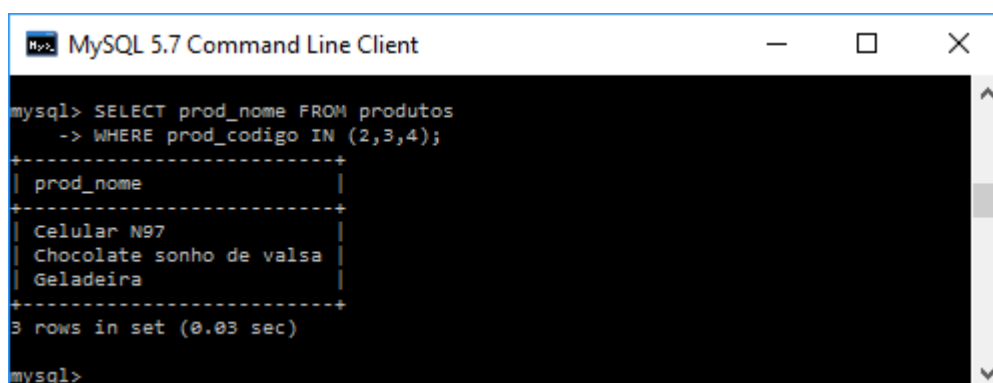
Fonte: MySQL Server 5.7 Command Line Client

Esses são os códigos dos produtos que foram vendidos, mas essa resposta não fornece muita informação acerca de quais são esses produtos. Então, uma nova pesquisa é necessária para identificar quais são os nomes dos produtos que possuem código 2, 3 e 4. Para tanto, podemos utilizar o seguinte comando.

```
1 mysql> SELECT prod_nome
2   FROM produtos
3  WHERE prod_codigo IN (2,3,4);
```

A resposta do sistema SGBD a essa pesquisa é ilustrada na **Figura 4**.

Figura 04 - Tela do MySQL após o comando SELECT para determinar o nome dos produtos vendidos.



```
mysql> SELECT prod_nome FROM produtos
-> WHERE prod_codigo IN (2,3,4);
+-----+
| prod_nome |
+-----+
| Celular N97 |
| Chocolate sonho de valsa |
| Geladeira |
+-----+
3 rows in set (0.03 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client

Será que não é possível combinar essas duas consultas para que não se tenha de escrever e solicitar duas consultas separadas ao banco de dados? A resposta a essa pergunta é a **subconsulta**, o assunto que estudaremos agora.

Subconsulta é uma instrução SELECT na forma (SELECT... FROM... WHERE...) adicionada dentro de outra instrução SELECT. Podendo também ser utilizada nas instruções INSERT, DELETE e UPDATE como parâmetro da cláusula WHERE.

Vamos combinar as duas consultas realizadas anteriormente ao banco de dados **sistvendas** em única consulta usando uma subconsulta? A consulta que retorna os códigos dos produtos que foram vendidos será a consulta interna ou subconsulta. E a consulta que identifica os nomes dos produtos vendidos será a consulta externa. Para ver como isso funciona, analise o seguinte comando, que realiza uma consulta com a subconsulta, descrito no quadro a seguir.

```
1 mysql> SELECT prod_nome
2   FROM produtos
3   WHERE prod_codigo IN (SELECT comp_codproduto FROM compras);
```

O que a instrução SELECT mais interna faz é retornar os códigos dos produtos que já foram vendidos. Esses códigos são utilizados pela expressão da consulta mais externa para filtrar os nomes dos produtos que devem ser visualizados.

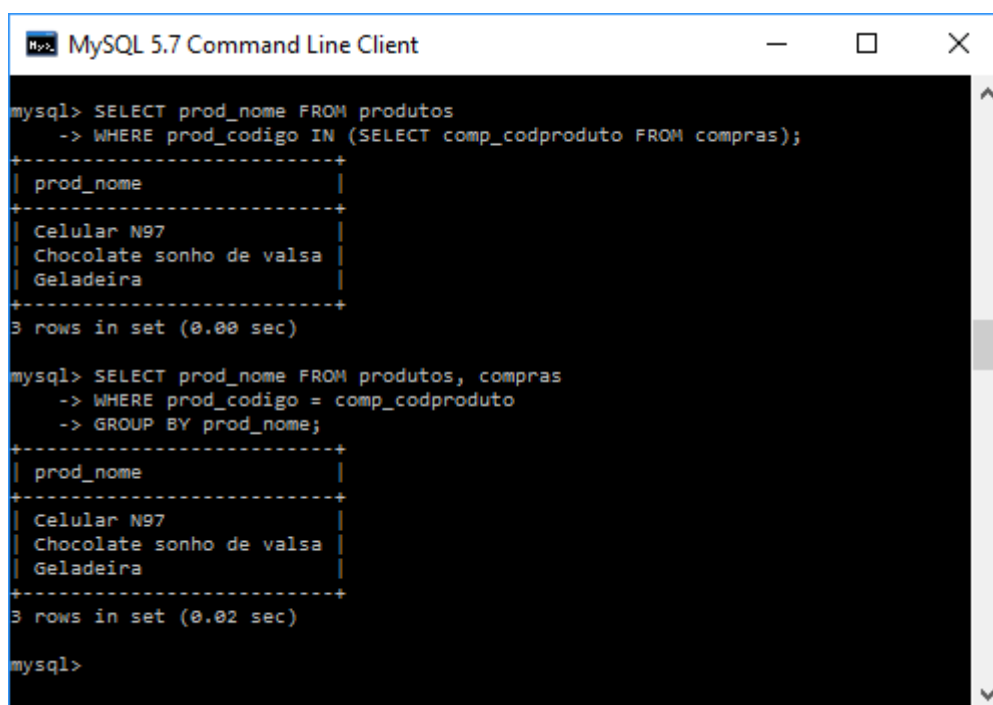
Observe no exemplo anterior alguns pontos importantes. Em primeiro lugar, a subconsulta está entre parênteses "()". Sempre devemos colocar uma subconsulta entre parênteses, pois é assim que o SGBD consegue fazer sua identificação. Segundo, a subconsulta não tem seu próprio sinal de ponto e vírgula. O sinal de ponto e vírgula é utilizado somente ao final de uma consulta completa (consulta externa + consulta interna). Terceiro, a subconsulta em questão está retornando uma lista de valores. Por isso, utilizamos o operador IN, que é um operador de agregação que indica que o sistema está procurando um conjunto de valores, conforme foi visto em aulas anteriores. Se tivéssemos utilizado na cláusula WHERE um operador de comparação, como o sinal de igualdade (=), a subconsulta deve retornar apenas um valor simples.

Uma pergunta que pode surgir nesse momento é se a consulta feita não poderia ser realizada utilizando conexões. A verdade é que uma consulta contendo uma subconsulta não é o único jeito de realizar a pesquisa em questão. Poderíamos ter realizado a mesma consulta utilizando conexão cruzada, conforme é descrito no quadro a seguir.

```
1 mysql> SELECT prod_nome
2   FROM produtos, compras
3  WHERE prod_codigo = comp_codproduto GROUP BY prod_nome;
```

O termo GROUP BY agrupa as linhas com valores iguais de uma determinada coluna, conforme apresentado anteriormente. Observe na **Figura 5** que ambas as consultas produzem o mesmo resultado impresso na tela.

Figura 05 - Tela do MySQL após diversos comandos SELECT para listar o nome dos produtos vendidos.



```
mysql> SELECT prod_nome FROM produtos
-> WHERE prod_codigo IN (SELECT comp_codproduto FROM compras);
+-----+
| prod_nome |
+-----+
| Celular N97 |
| Chocolate sonho de valsa |
| Geladeira |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT prod_nome FROM produtos, compras
-> WHERE prod_codigo = comp_codproduto
-> GROUP BY prod_nome;
+-----+
| prod_nome |
+-----+
| Celular N97 |
| Chocolate sonho de valsa |
| Geladeira |
+-----+
3 rows in set (0.02 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client

Atividade 01

1. Vamos praticar um pouco para que você se familiarize com o comando de consulta com subconsulta. Entre no banco de dados do sistvendas. Elabore as seguintes consultas utilizando subconsultas.
 - a. Nome dos clientes que efetuaram compras.
 - b. Nome do cliente com código 1 e suas compras.
 - c. Nome do cliente com código 2 e suas compras.
 - d. Nome de todos os clientes e suas respectivas compras.

Subconsultas versus conexões

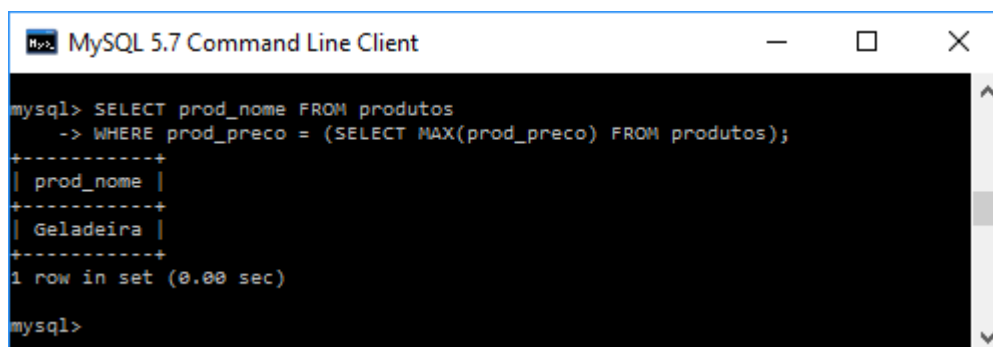
Visto que é possível realizar uma pesquisa, como, por exemplo, o nome dos produtos que já foram vendidos, usando uma consulta com subconsulta ou com conexão, você saberia dizer qual é a melhor estratégia?

De uma maneira geral, consultas usando conexões são mais eficientes que consultas com subconsultas, pois o *Query Optimizer* (mecanismo interno do SGBD que otimiza as instruções enviadas para o banco de dados) pode executar algumas operações a mais quando se utiliza uma subconsulta degradando assim a performance da execução. Ou seja, uma consulta com subconsulta pode ser mais demorada do que uma consulta com conexão. Entretanto, as consultas com subconsultas são apropriadas para pesquisas que envolvam comparações com agregações, como a consulta a seguir:

```
1 mysql> SELECT prod_nome
2   FROM produtos
3  WHERE prod_preco = (SELECT MAX(prod_preco) FROM produtos);
```

Nessa consulta, a instrução `SELECT` mais interna retorna o preço máximo dos produtos da tabela `PRODUTOS`, através da função de agregação `MAX()`. A seguir, esse valor é comparado com todos os produtos e somente aqueles que possuírem o **prod_preco** com valor máximo serão retornados pela consulta mais externa. A resposta do SGBD a essa consulta é ilustrada na **Figura 6**.

Figura 06 - Tela do MySQL após o comando `SELECT` para listar os produtos com preço máximo.



```
MySQL 5.7 Command Line Client
mysql> SELECT prod_nome FROM produtos
  -> WHERE prod_preco = (SELECT MAX(prod_preco) FROM produtos);
+-----+
| prod_nome |
+-----+
| Geladeira |
+-----+
1 row in set (0.00 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client



Vídeo 02 - Subconsultas

Atividade 02

1. Elabore uma consulta ao banco de dados do sistvendas usando subconsulta e faça o mesmo usando conexões. Em sua opinião, qual é a mais fácil de implementar? Justifique.

Subconsultas como uma coluna na instrução SELECT

Podemos também utilizar o resultado de uma subconsulta no lugar de uma coluna que será retornada na instrução SELECT, conforme é descrito a seguir.

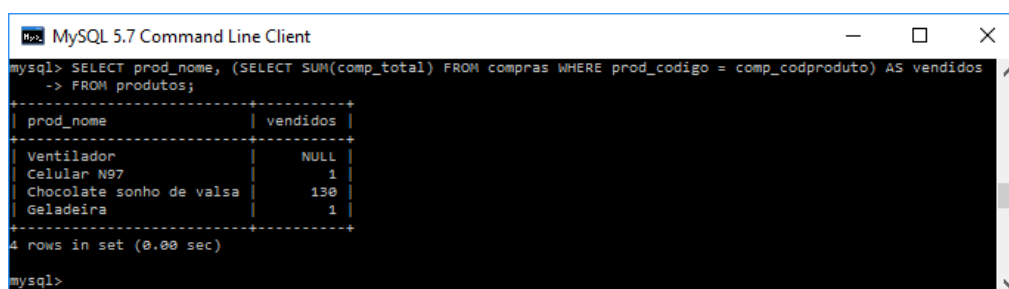
```
1 mysql> SELECT tabela1.atributo1,  
2   (SELECT tabela2.atributoX FROM tabela2 WHERE condição), tabela1.atributo2 ...  
3   FROM tabela1;
```

Observe que os atributos *tabela1.atributo1* e *tabela1.atributo2* são atributos pertencentes à tabela 1 que se deseja visualizar. Além desses atributos, deseja-se visualizar o resultado da subconsulta que retorna um determinado atributo (*tabela2.atributoX*) da tabela 2. É importante destacar que mesmo que a tabela 2 possua mais de uma coluna (atributo), somente pode-se retornar uma única coluna. Para um melhor entendimento acerca desse tópico, examine o exemplo a seguir que lista todos os nomes dos produtos e quantidade de itens vendidos.

```
1 mysql> SELECT prod_nome,  
2   (SELECT SUM(comp_total) FROM compras WHERE prod_codigo= comp_codproduto) AS vendidos  
3   FROM produtos;
```

Em termos simples, a consulta feita está exibindo em cada linha o nome de um produto e a quantidade de itens vendidos desse produto. Para ser executada, a consulta interna necessita da informação do atributo **prod_codigo**, porém, essa informação pertence à tabela **produtos**, que é acessada a partir da consulta externa. O termo AS é utilizado no SELECT externo para nomear o resultado da subconsulta como sendo vendidos com a finalidade de deixar a consulta mais legível. A função SUM retorna a soma dos itens vendidos de um determinado produto, conforme pode ser verificado na **Figura 7**. Lembre-se de que a subconsulta nesse caso deve retornar um valor único. Então, cada vez que ela é executada, uma única linha é retornada.

Figura 07 - Tela do MySQL após o comando SELECT para listar a quantidade total de produtos vendidos.



```
mysql> SELECT prod_nome, (SELECT SUM(comp_total) FROM compras WHERE prod_codigo = comp_codproduto) AS vendidos
-> FROM produtos;
+-----+-----+
| prod_nome | vendidos |
+-----+-----+
| Ventilador | NULL     |
| Celular N97 | 1        |
| Chocolate sonho de valsa | 130      |
| Geladeira  | 1        |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client

Atividade 03

1. Seu SGBD avisa a você quando algo está errado no seu código, mas às vezes a resposta é um tanto vaga. Examine, a seguir, os comandos SELECT e tente descobrir o que há de errado com ele. Em seguida, digite no seu sistema e observe a mensagem exibida.
 - a. SELECT prod_nome, (SELECT cli_nome FROM clientes) FROM produtos;
 - b. SELECT prod_nome, (SELECT cli_nome, cli_cpf FROM clientes WHERE cli_codigo=1) AS cliente1 FROM produtos;
 - c. SELECT prod_nome, (SELECT cli_nome, FROM clientes WHERE cli_codigo=1;) AS cliente1 FROM produtos;

Subconsulta correlacionada e subconsulta não correlacionada

No exemplo anterior, que lista todos os nomes dos produtos e quantidade de itens vendidos, vimos que a consulta interna necessita da informação sobre o atributo **prod_codigo** da tabela **produtos** para ser executada, que está disponível para acesso a partir da consulta externa. Sendo assim, a subconsulta não pode ser executada como uma consulta independente. A consulta externa tem que ser executada antes para sabermos qual é o valor de **prod_codigo**. Quando isso ocorre, dizemos que a subconsulta é correlacionada.

Subconsulta correlacionada é quando a consulta interna depende dos valores retornados pela consulta externa para ser processada. Na subconsulta não correlacionada, a consulta interna funciona sozinha, não necessitando de nenhuma informação da consulta externa, podendo ser executada como uma consulta independente.

Para entendermos melhor a diferença entre subconsulta correlacionada e subconsulta não correlacionada, vamos analisar os seguintes exemplos. Examine com cuidado e não deixe que nenhuma dúvida fique sem ser esclarecida.

1º exemplo

Subconsulta correlacionada

Pesquisar o nome e o total de produtos comprados por cada cliente.

```
1 mysql> SELECT cli_nome,  
2   (SELECT SUM(comp_total)  
3   FROM compras  
4   WHERE cli_codigo= comp_codcliente) AS itenscomprados  
5   FROM clientes;
```

Nesse exemplo, a subconsulta necessita, para ser executada, da informação de qual é o código do cliente, que será informada pela consulta externa a cada linha que a consulta externa processe. O resultado dessa pesquisa é ilustrado na **Figura 8**.

2º exemplo

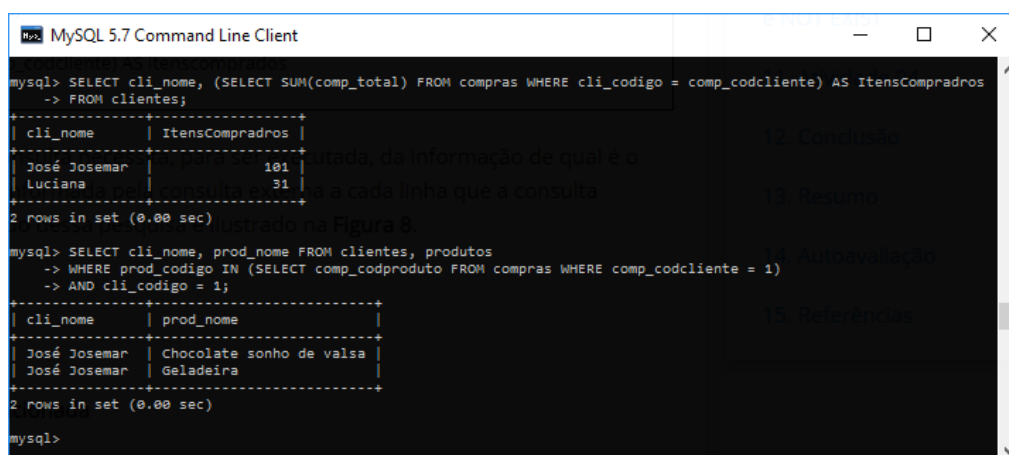
Subconsulta não correlacionada

Pesquisar o nome do cliente com o atributo **cli_codigo=1** e os produtos por ele comprados.

```
1 mysql> SELECT cli_nome, prod_nome
2   FROM clientes, produtos
3   WHERE prod_codigo IN (SELECT comp_codproduto FROM compras
4   WHERE comp_codcliente=1) and cli_codigo=1;
```

Nesse exemplo, tem-se uma junção de conexões e subconsulta não correlacionada em uma única consulta. A subconsulta não depende de nenhuma informação da consulta externa para ser executada, trabalhando apenas com as informações provenientes da tabela **compras**, sendo, portanto, uma subconsulta não correlacionada. A consulta externa realiza uma pesquisa empregando conexão cartesiana. O resultado dessa pesquisa é ilustrado na **Figura 8**. Observe que as linhas listadas correspondem ao produto cartesiano entre as tabelas clientes e produtos que possuem o código do produto pertencente ao conjunto retornado pela subconsulta.

Figura 08 - Tela do MySQL após os comandos SELECTs dos exemplos 1 e 2.



```
mysql> SELECT cli_nome, (SELECT SUM(comp_total) FROM compras WHERE cli_codigo = comp_codcliente) AS ItensComprados
-> FROM clientes;
+-----+-----+
| cli_nome | ItensComprados |
+-----+-----+
| José Josemar | 101 |
| Luciana | 31 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT cli_nome, prod_nome FROM clientes, produtos
-> WHERE prod_codigo IN (SELECT comp_codproduto FROM compras WHERE comp_codcliente = 1)
-> AND cli_codigo = 1;
+-----+-----+
| cli_nome | prod_nome |
+-----+-----+
| José Josemar | Chocolate sonho de valsa |
| José Josemar | Geladeira |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client



Vídeo 03 - Subconsultas como Coluna

Subconsulta com EXIST e NOT EXIST

Uma das grandes aplicações da subconsulta correlacionada é a realização de testes de existência e inexistência, que são realizados com as cláusulas EXISTS e NOT EXISTS, respectivamente, juntamente com a subconsulta.

A realização de um teste de inexistência permite encontrar todas as linhas de uma tabela (referenciada na consulta externa), que não contém os dados fornecidos pela subconsulta.

Vamos supor que precisamos determinar quais foram os produtos que não foram vendidos no nosso banco de dados **sistvendas**. Uma forma de realizar essa pesquisa é utilizar um NOT EXISTS para encontrar os produtos.

```
1 mysql> SELECT prod_nome
2   FROM produtos
3   WHERE NOT EXISTS (SELECT * FROM compras
4     WHERE prod_codigo= comp_codproduto);
```

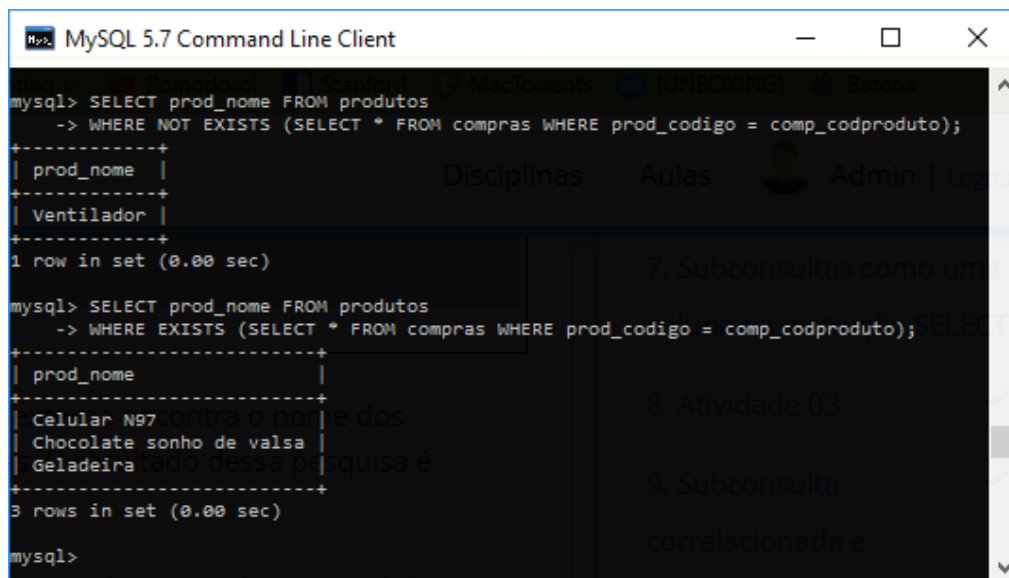
Com a utilização da cláusula NOT EXISTS, a consulta externa encontra o nome dos produtos que ainda não estão listados na tabela **compras**. O resultado dessa pesquisa é ilustrado na **Figura 9**.

De maneira semelhante, podemos listar os nomes dos produtos que foram vendidos utilizando o termo EXISTS.

```
1 mysql> SELECT prod_nome
2   FROM produtos
3   WHERE EXISTS (SELECT * FROM compras
4     WHERE prod_codigo= comp_codproduto);
```

A consulta feita retorna os nomes dos produtos em que o atributo **prod_codigo** aparece pelo menos uma vez na tabela **compras**. O resultado dessa pesquisa encontra-se na **Figura 9**.

Figura 09 - Tela do MySQL após os testes de inexistência e existência .



```
mysql> SELECT prod_nome FROM produtos
-> WHERE NOT EXISTS (SELECT * FROM compras WHERE prod_codigo = comp_codproduto);
+-----+
| prod_nome |
+-----+
| Ventilador |
+-----+
1 row in set (0.00 sec)

mysql> SELECT prod_nome FROM produtos
-> WHERE EXISTS (SELECT * FROM compras WHERE prod_codigo = comp_codproduto);
+-----+
| prod_nome |
+-----+
| Celular N97 |
| Chocolate sonho de valsa |
| Geladeira |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Fonte: MySQL Server 5.7 Command Line Client



Vídeo 04 - EXIST e NOT EXIST

Atividade 04

1. Acesse o banco de dados do **cineOnline**. Elabore as seguintes consultas utilizando subconsultas com teste de existência e inexistência, conforme for o caso.
 - a. Nome dos filmes que tiveram ingressos vendidos.
 - b. Nome dos clientes que não compraram ingressos

Conclusão

Encerramos por aqui mais uma aula sobre a linguagem SQL. Na próxima, aprenderemos uma forma alternativa de olhar os dados contidos em uma ou mais tabelas usando VISÕES ou VIEWS. Com VISÕES, é possível tratar os resultados de uma consulta como sendo tabelas virtuais. As VISÕES são ótimas para transformar consultas complexas em consultas simples.

Para ajudar na apreensão do conteúdo estudado nesta aula, pegue seu caderno e faça um resumo do que foi visto, não deixe de praticar bastante para não esquecer. Lembre-se de fazer sua autoavaliação. Bons estudos e boa sorte!

Resumo

Nesta aula, você estudou como realizar uma consulta com uma subconsulta anexada. Viu que o uso de subconsultas é similar ao uso de conexões. Estudou como realizar uma consulta em que o resultado da subconsulta é uma das informações a serem exibidas pela consulta externa. Aprendeu o que é uma subconsulta correlacionada e não correlacionada e como realizar subconsultas correlacionadas com teste de inexistência e existência.

Autoavaliação

1. Qual(ais) a(s) diferença(s) entre subconsultas e conexões?
2. O que você entendeu por subconsultas correlacionadas e não correlacionadas?
3. Altere as seguintes tabelas do banco de dados locadora e insira dados nelas:
 - a. Clientes (codigo [chave primária], nome, cpf, sexo, profissao, salario)
 - b. Filmes (codigo [chave primária], titulo, genero, duracao, ano, situacao, preco)
 - c. Locacoes (codigo, codigo cliente [chave estrangeira], codigo do filme [chave estrangeira], data)
4. Resolva as seguintes consultas utilizando a linguagem SQL:
 1. Qual o nome de todos os clientes que já alugaram filmes?
 2. Qual o título e o gênero de todos os filmes alugados?
 3. Qual a profissão e o sexo de todos os clientes que alugaram filmes de comédia?
 4. Qual o gênero dos filmes alugados por estudantes?

5. Qual a quantidade de pessoas de cada sexo que alugaram filmes de suspense?
6. Qual a média salarial das pessoas que alugaram filmes de aventura?
7. Para cada locação do filme E O VENTO LEVOU, liste o nome do cliente.
8. Para cada locação, exiba o nome e o CPF do cliente, o título, o gênero, o preço e a data da locação.
9. Para cada locação, exiba o nome do cliente, a data da locação, a quantidade de filmes alugados e o preço total.

Referências

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Editora AltaBooks, 2008.

MySQL 5.1 Reference Manual. Disponível em:
<<http://dev.mysql.com/doc/refman/5.1/en/>>. Acesso em: 24 set. 2010.