

BANCO DE DADOS RELACIONAL

Prof. Nilton



Aula de hoje



The diagram features a large dark grey rounded rectangle on the left containing the text 'Aula de hoje'. To its right is a black hexagon with the text 'SUBCONSULTAS'. Above this hexagon are two smaller white hexagons with black outlines. A black line connects the bottom of the 'SUBCONSULTAS' hexagon to a larger white hexagon with a black outline on the right side of the image.

SUBCONSULTAS

SUBCONSULTA

Chamamos de **subconsulta** ou ***consulta aninhada***, a utilização de uma consulta dentro de outra consulta. É utilizada para retornar dados que serão utilizados na consulta principal, realizar teste de membro de conjunto, criar comparações de conjunto e determinar cardinalidade de conjunto.

```
SELECT NOME_DA_COLUNA
FROM TABELA
WHERE NOME_DA_COLUNA = (SELECT NOME_DA_COLUNA
                        FROM TABELA
                        WHERE CONDIÇÕES)
```



SUBCONSULTA - Exemplo

SELECT nome, preco

FROM Produto

WHERE preco > (**SELECT AVG**(preco) **FROM** produto)

Quando uma subconsulta é utilizada em uma consulta, a subconsulta é resolvida primeiro e depois a consulta principal é resolvida de acordo com a(s) condição(ões) resolvida(s) pela subconsulta.



SUBCONSULTA

Os resultados da subconsulta são utilizados para processar expressões na cláusula WHERE ou na cláusula HAVING da consulta principal.

Operadores relacionais e lógicos, como =, >, < , < >, **IN**, **NOT IN**, **EXISTS**, **NOT EXISTS**, **AND**, **OR**, etc., podem ser utilizados dentro da subconsulta bem como para avaliar uma subconsulta na clausula WHERE ou HAVING.

SUBCONSULTA - Regras

- Devem ser incluídas entre parênteses.
- Pode ter apenas uma coluna na cláusula SELECT, a menos que múltiplas colunas estejam na consulta principal.
- Order By não pode ser utilizado na subconsulta.
- Subconsultas que retornam mais de uma linha podem ser utilizadas somente com operador do tipo IN.
- Between não pode ser utilizado com uma subconsulta, mas pode ser utilizado dentro da subconsulta.

SUBCONSULTA - Exemplo

SELECT P.*

FROM projetos P

WHERE P.id **IN** (**SELECT** C.id_projeto **FROM** comentario C **WHERE**
C.id_projeto = P.id)

SUBCONSULTA

As subconsultas também podem ser utilizadas em conjunção com instruções DML.

```
INSERT INTO gerente(nome, departamento, data_admissao)  
(SELECT nome, departamento, data_admissao FROM supervisor WHERE  
YEAR(data_admissao) < 2016 )
```

```
DELETE FROM supervisor  
WHERE nome IN (SELECT nome FROM gerente G WHERE G.nome = nome)
```


SUBCONSULTA

UPDATE GERENTE

SET SALARIO = (**SELECT** SALARIO **FROM** PLANOCARREIRA **WHERE** IDFUNCAO = 2)



SUBCONSULTA

Uma subconsulta pode ser embutida dentro de outra subconsulta.

```
SELECT NOME_DA_COLUNA
FROM TABELA 1
WHERE NOME_DA_COLUNA = (SELECT NOME_DA_COLUNA
                        FROM TABELA
                        WHERE NOME_DA_COLUNA = (SELECT NOME_DA_COLUNA
                                                FROM TABELA
                                                WHERE NOME_DA_COLUNA =  VALOR)))
```

SUBCONSULTAS CORRELACIONADAS

As subconsultas correlacionadas são comuns em muitas implementações de SQL. Esse tipo de subconsulta é dependente das informações da consulta principal.

```
SELECT A.NOME  
FROM GERENTE A  
WHERE 5 < (SELECT COUNT(B.*)  
           FROM PROJETOS B  
           WHERE B.IDGERENTE = A.ID)
```

SELECT AS FIELD – SELECT COMO COLUNA

Esta variação da instrução [SELECT](#) permite que uma outra subconsulta do tipo SELECT seja utilizada como uma coluna da consulta principal, permitindo a construção de comandos SQL mais flexíveis, suportando o uso de [funções agregadoras](#).

Essa abordagem de subconsulta é capaz de trazer informações que foram coletadas de outras consultas (sejam em outras tabelas ou até em resultados) utilizando funções agregadas e exibi-las como uma nova coluna, o que não é possível utilizando a prática de JOINS, pois a função agregada precisa agir em um todo, e não registro a registro, como acontece com junções.

SELECT AS FIELD

Sintaxe:

```
SELECT [ coluna1, coluna2, [ SELECT [ coluna ] FROM [ tabela1 ] ] AS coluna 3 ... | * ]  
FROM [ tabela1, tabela2, ... ]
```

Exemplo: */* Quantas pessoas participaram no evento */*

```
SELECT E.nome, E.data, ( SELECT COUNT(id) FROM entrada_evento EV WHERE  
                        EV.idevento = E.id ) AS total  
FROM evento E  
GROUP BY E.ID  
ORDER BY E.data DESC;
```

SELECT AS FIELD

```
SELECT V.ID, V.VALOR, V.DATA, (SELECT COUNT(P.ID_VENDA) FROM VENDAS_PARCELAS P  
    WHERE (V.ID = P.ID_VENDA) AND (P.DATA_PAGAMENTO is  
    NOT NULL) ) AS QTD_PARCELAS_PAGAS  
  
FROM VENDAS V
```

```
SELECT V.ID, V.VALOR, V.DATA, ( SELECT COUNT(P.ID_VENDA) FROM VENDAS_PARCELAS P  
    WHERE (V.ID = P.ID_VENDA) AND (P.DATA_PAGAMENTO is  
    NOT NULL) ) AS QTD_PARCELAS_PAGAS,  
    ( SELECT COUNT(P.ID_VENDA) FROM VENDAS_PARCELAS P  
    WHERE (V.ID = P.ID_VENDA) AND (P.DATA_PAGAMENTO is  
    NULL) ) AS QTD_PARCELAS_ABERTAS  
  
FROM VENDAS V
```

SELECT FROM SELECT

Esta variação da instrução [SELECT](#) permite que uma consulta seja feita dentro do resultado de outra consulta, permitindo a construção de comandos SQL mais flexíveis, suportando o uso de funções agregadoras.



SELECT FROM SELECT

Sintaxe:

SELECT [coluna1, coluna2, ...]

FROM (**SELECT** [coluna1, coluna2, ...] **FROM** [tabela1, tabela2, ...])

SELECT FROM SELECT

```
SELECT SUM(VALOR) as TOTAL  
FROM ( SELECT V.ID, V.VALOR, P.DATA_PAGAMENTO,  
        P.NUMERO_PARCELA  
        FROM VENDAS V, VENDAS_PARCELAS P  
        WHERE (V.ID = P.ID_VENDA) AND (P.NUMERO_PARCELA = 1)  
        AND (P.DATA_PAGAMENTO IS NOT NULL) AND  
        (month(P.DATA_PAGAMENTO) = 7) ) as tabela
```

SELECT FROM SELECT

A consulta anterior tem como objetivo saber quantas comissões serão pagas no mês de julho Para isso, primeiro precisamos pegar todas as vendas que tiveram a primeira parcela paga, e em seguida, verificar o total dessa consulta.

WHERE

Exists

- O EXISTS é uma cláusula SQL que testa quando há um ou mais resultados em uma SUBQUERY e retorna o valor **TRUE**, permitindo filtrar colunas dentro de uma subconsulta.
- A cláusula EXISTS faz uma verificação se existe algum resultado para a subquery informada. Caso haja, o resultado da consulta principal é exibido. É muito comum sua utilização quando se deseja trazer resultados onde um valor específico existe dentro de outra tabela.

WHERE

```
SELECT [ coluna1, coluna2, ... | * ] FROM [ tabela1, tabela2, ... ] WHERE EXISTS (  
SELECT [ coluna1, coluna2, ... | * ] FROM [ tabela1, tabela2, ... ] WHERE [ condicao ]  
)
```

```
SELECT P.ID, P.nome  
FROM produto P  
WHERE EXISTS (SELECT V.ID_PRODUTO  
                FROM venda_produto V  
                WHERE V.ID_PRODUTO = P.ID )
```

WHERE

NOT EXISTS

```
SELECT P.ID, P.nome FROM produto P WHERE NOT EXISTS ( SELECT V.ID_PRODUTO  
FROM venda_produto V WHERE V.ID_PRODUTO = P.ID )
```

WHERE

IN

IN é um operador para especificar vários valores em uma cláusula [WHERE](#). Com ele podemos verificar se determinada coluna está contida em um determinado grupo de valores, seja ele definido manualmente ou através de subqueries.

WHERE

NOT IN

NOT IN é um operador para especificar vários valores em uma cláusula [WHERE](#). Com ele podemos verificar se determinada coluna **não** está contida em um determinado grupo de valores, seja ele definido manualmente ou através de subquerys.

CASE



A instrução CASE permite que sejam utilizadas condições e retornado um valor quando a primeira condição é atendida (como uma instrução IF-THEN-ELSE). Então, uma vez que uma condição é verdadeira, ela irá parar de ler e retornar o resultado.



Se nenhuma condição for verdadeira, retornará o valor na cláusula ELSE.



Se não houver nenhuma parte do ELSE e nenhuma condição for verdadeira, ela retornará NULL.

CASE

- **Syntax**

- CASE

- WHEN *condição1* THEN *resultado1*
 - WHEN *condição2* THEN *resultado2*
 - WHEN *condiçãoN* THEN *resultadoN*
 - ELSE *resultado*

- END;

CASE

SELECT ID, NOME,

CASE

WHEN TIPO = 'F' THEN "Pessoa Fisica"

WHEN TIPO = 'J' THEN "Pessoa Juridica"

ELSE "Outros"

END

FROM PESSOA;