


Comando SELECT

Consultas aninhadas



Profa. Alessandra Aparecida Paulino
Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo

CLÁUSULA WHERE OPERADOR IN

IN: Determina se o valor de uma expressão é igual a algum dos vários valores em uma lista especificada. Se *expr for encontrado na lista de valores*, o operador IN retornará True; caso contrário, retornará False.

EXEMPLO:

/ Listar todos os alunos provenientes de São Paulo, São Carlos ou Rio de Janeiro*/*

SELECT *

FROM ALUNO

WHERE UPPER(Cidade) IN ('SAO PAULO', 'SAO CARLOS', 'RIO DE JANEIRO'); -- **UPPER** transforma os caracteres do atributo Cidade em maiúsculo



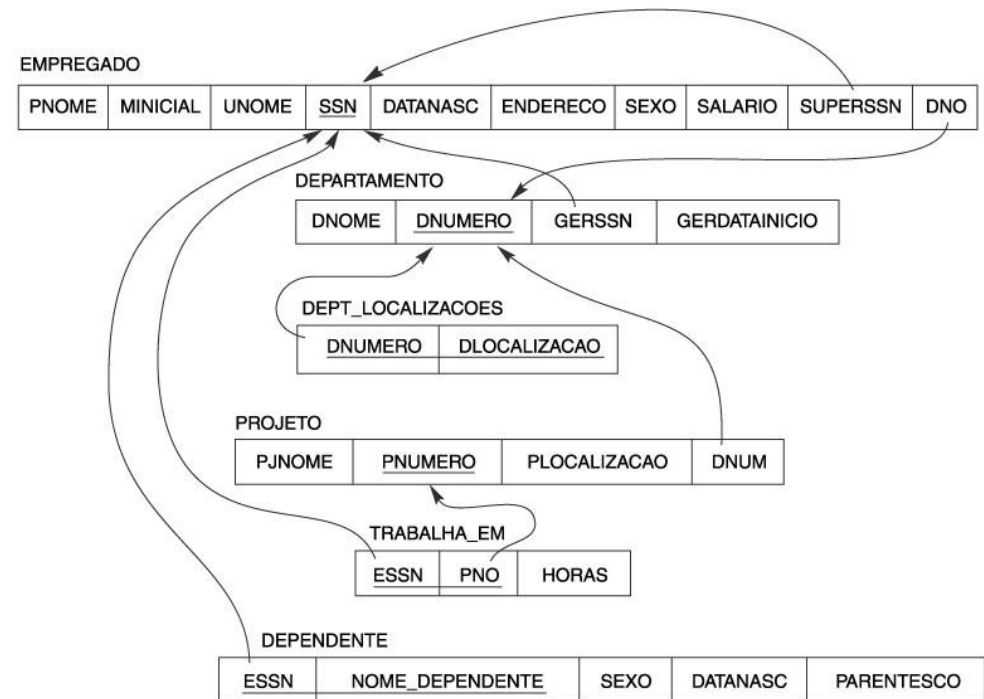
CLÁUSULA WHERE

OPERADOR IN

IN + subconsulta: O operador IN pode trabalhar com subconsulta

/ Listar os nomes funcionários que trabalham em projetos'*/*

SELECT nome
FROM empregado
WHERE ssn **IN** (
 SELECT essn
 FROM trabalha_em)

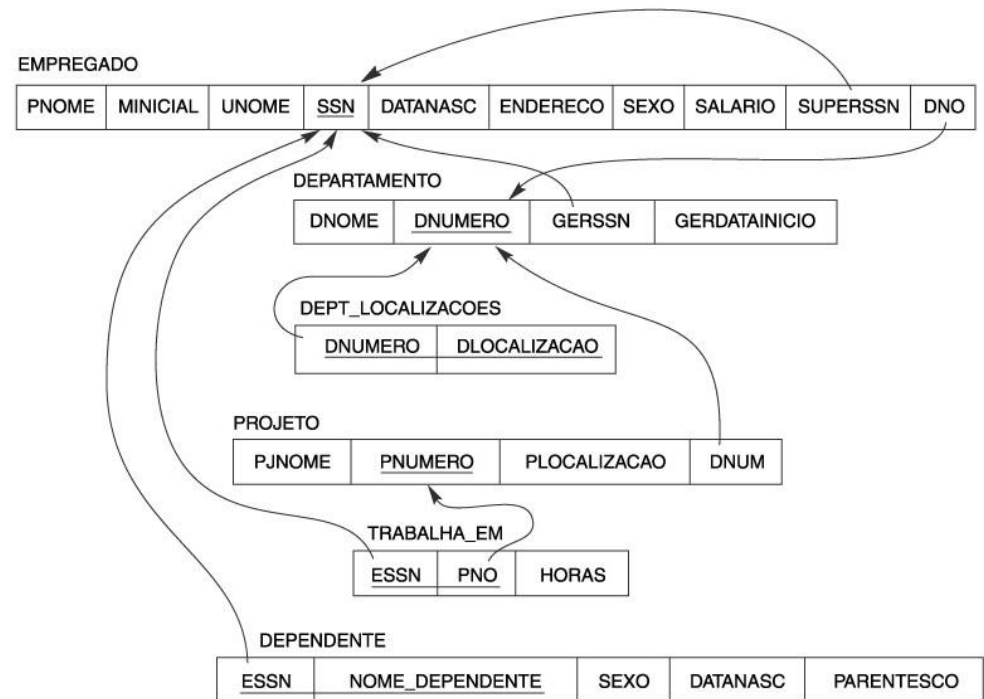


CLÁUSULA WHERE OPERADOR IN

NOT IN + subconsulta: O operador NOT IN pode trabalhar com subconsulta.

/ Listar os nomes funcionários que não trabalham em nenhum projeto */*

SELECT nome
FROM empregado
WHERE ssn **NOT IN** (
 SELECT essn
 FROM trabalha_em)



Comando SELECT

Consultas aninhadas

- ▶ Muitas vezes é preciso buscar uma informação no banco para, então, usá-la na condição de comparação de uma consulta
- ▶ Isso pode ser feito por meio de consultas aninhadas:
 - ▶ Forma-se um bloco SELECT-FROM-WHERE dentro da cláusula WHERE de outra consulta.
 - ▶ Essa outra consulta é chamada de consulta interna.

/ Listar os projetos dos funcionários de nome 'José'*/*

```
SELECT projeto_pnumero  
FROM trabalha_em  
WHERE essn IN (  
    SELECT ssn  
    FROM empregado  
    WHERE pnome = 'José')
```



Consulta aninhada
(consulta interna)



Comando SELECT

Consultas aninhadas – Operador IN

/ Listar os nomes funcionários que não trabalham em nenhum projeto'*/*

SELECT nome

FROM empregado

WHERE ssn **NOT IN** (

SELECT essn

FROM trabalha_em)



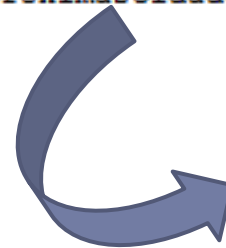
Comando SELECT

Consultas aninhadas – Operador IN

Tabela Filiais

| Nome | Cidade | UF |
|----------|----------------|----|
| Brás | São Paulo | SP |
| Limão | São Paulo | SP |
| Penha | São Paulo | SP |
| Tijuca | Rio de Janeiro | RJ |
| Barra | Rio de Janeiro | RJ |
| Angra | Angra dos Reis | RJ |
| Pampulha | Belo Horizonte | MG |
| Glória | Belo Horizonte | MG |
| Tibery | Uberlândia | MG |
| Centro | Uberlândia | MG |
| Internet | | |

```
CREATE TABLE ProximasCidades(  
    cidade varchar(50));  
DELETE FROM ProximasCidades;  
INSERT INTO ProximasCidades  
    Select Distinct cidade  
    from filiais  
    where cidade is not null;  
INSERT INTO ProximasCidades VALUES ('Araguari'),('Ituiutaba');  
SELECT * FROM proximascidades
```



/ Qual o resultado? */*

```
SELECT *  
FROM filiais  
WHERE cidade IN (  
    SELECT cidade  
    FROM ProximasCidades)
```

| | cidade character varying(50) |
|---|----------------------------------------|
| 1 | Rio de Janeiro |
| 2 | Belo Horizonte |
| 3 | Uberlândia |
| 4 | Angra dos Reis |
| 5 | São Paulo |
| 6 | Araquari |
| 7 | Ituiutaba |

Comando SELECT

Consultas aninhadas – Operador IN

/ Qual o resultado?*

São as cidades que já temos filiais mas que ainda estão na lista de próximas cidade

*E a filial da Internet? */*

```
SELECT *  
FROM filiais  
WHERE cidade IN (  
  SELECT cidade  
  FROM ProximasCidades)
```

| | nome character varying(30) | cidade character varying(50) | uf character(2) |
|----|-------------------------------|---------------------------------|--------------------|
| 1 | Penha | São Paulo | SP |
| 2 | Limão | São Paulo | SP |
| 3 | Brás | São Paulo | SP |
| 4 | Glória | Belo Horizonte | MG |
| 5 | Pampulha | Belo Horizonte | MG |
| 6 | Centro | Uberlândia | MG |
| 7 | Tibery | Uberlândia | MG |
| 8 | Barra | Rio de Janeiro | RJ |
| 9 | Tijuca | Rio de Janeiro | RJ |
| 10 | Angra | Angra dos Reis | RJ |



Comando SELECT

Consultas aninhadas – Operador IN

Tabela Filiais

| Nome | Cidade | UF |
|----------|----------------|----|
| Brás | São Paulo | SP |
| Limão | São Paulo | SP |
| Penha | São Paulo | SP |
| Tijuca | Rio de Janeiro | RJ |
| Barra | Rio de Janeiro | RJ |
| Angra | Angra dos Reis | RJ |
| Pampulha | Belo Horizonte | MG |
| Glória | Belo Horizonte | MG |
| Tibery | Uberlândia | MG |
| Centro | Uberlândia | MG |
| Internet | | |

```
CREATE TABLE ProximasCidades(  
  cidade varchar(50));  
DELETE FROM ProximasCidades;  
INSERT INTO ProximasCidades  
  Select Distinct cidade  
  from filiais  
  where cidade is not null;  
INSERT INTO ProximasCidades VALUES ('Araguari'), ('Ituiutaba');  
SELECT * FROM proximascidades
```



/ Qual o resultado? */*

```
SELECT *  
FROM ProximasCidades  
WHERE cidade IN (  
  SELECT cidade  
  FROM filiais)
```

| | cidade character varying(50) |
|---|----------------------------------------|
| 1 | Rio de Janeiro |
| 2 | Belo Horizonte |
| 3 | Uberlândia |
| 4 | Angra dos Reis |
| 5 | São Paulo |
| 6 | Araguari |
| 7 | Ituiutaba |

Comando SELECT

Consultas aninhadas – Operador IN

/ Qual o resultado?*

São as cidades que já temos filiais/*

```
SELECT *  
FROM ProximasCidades  
WHERE cidade IN (  
    SELECT cidade  
    FROM filiais)
```

| | cidade character varying(50) |
|----------|----------------------------------------|
| 1 | Rio de Janeiro |
| 2 | Belo Horizonte |
| 3 | Uberlândia |
| 4 | Angra dos Reis |
| 5 | São Paulo |



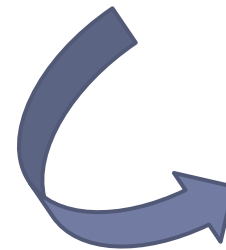
Comando SELECT

Consultas aninhadas – Operador IN

Tabela Filiais

| Nome | Cidade | UF |
|----------|----------------|----|
| Brás | São Paulo | SP |
| Limão | São Paulo | SP |
| Penha | São Paulo | SP |
| Tijuca | Rio de Janeiro | RJ |
| Barra | Rio de Janeiro | RJ |
| Angra | Angra dos Reis | RJ |
| Pampulha | Belo Horizonte | MG |
| Glória | Belo Horizonte | MG |
| Tibery | Uberlândia | MG |
| Centro | Uberlândia | MG |
| Internet | | |

```
CREATE TABLE ProximasCidades(  
  cidade varchar(50));  
DELETE FROM ProximasCidades;  
INSERT INTO ProximasCidades  
  Select Distinct cidade  
  from filiais  
  where cidade is not null;  
INSERT INTO ProximasCidades VALUES ('Araguari'), ('Ituiutaba');  
SELECT * FROM proximascidades
```



/ Qual o resultado? */*

```
SELECT *  
FROM ProximasCidades  
WHERE cidade NOT IN (  
  SELECT cidade  
  FROM filiais)
```


| | cidade character varying(50) |
|---|----------------------------------------|
| 1 | Rio de Janeiro |
| 2 | Belo Horizonte |
| 3 | Uberlândia |
| 4 | Angra dos Reis |
| 5 | São Paulo |
| 6 | Araguari |
| 7 | Ituiutaba |

Comando SELECT

Consultas aninhadas – Operador IN


/ Qual o resultado? */*

```
SELECT *  
FROM ProximasCidades  
WHERE cidade NOT IN (  
  SELECT cidade  
  FROM filiais)
```



| | cidade character varying(50) |
|--|---------------------------------|
| | |

```
SELECT *  
FROM ProximasCidades  
WHERE cidade NOT IN (  
  SELECT cidade  
  FROM filiais  
  WHERE cidade IS NOT NULL)
```



| | cidade character varying(50) |
|---|---------------------------------|
| 1 | Araquari |
| 2 | Ituiutaba |

Operadores booleanos

| a | b | a AND b | a OR b |
|-------|-------|---------|--------|
| TRUE | TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE |
| TRUE | NULL | NULL | TRUE |
| FALSE | FALSE | FALSE | FALSE |
| FALSE | NULL | FALSE | NULL |
| NULL | NULL | NULL | NULL |

| a | NOT a |
|-------|-------|
| TRUE | FALSE |
| FALSE | TRUE |
| NULL | NULL |

| | | | Resultado |
|----------|---|----------------|-----------|
| Araguari | = | São Paulo | False |
| Araguari | = | São Paulo | False |
| Araguari | = | São Paulo | False |
| Araguari | = | Rio de Janeiro | False |
| Araguari | = | Rio de Janeiro | False |
| Araguari | = | Angra dos Reis | False |
| Araguari | = | Belo Horizonte | False |
| Araguari | = | Belo Horizonte | False |
| Araguari | = | Uberlândia | False |
| Araguari | = | Uberlândia | False |
| Araguari | = | | NULL |

FALSE OR NULL => NULL

*lembre que NULL é **desconhecido**. Ou seja, desconhecemos o resultado da operação: se o *desconhecido* fosse TRUE, temos que

FALSE OR TRUE => TRUE;

se o *desconhecido* fosse FALSE, temos que

FALSE OR FALSE => FALSE

Assim, desconhecemos o resultado da operação, ou seja, o resultado é NULL

Comando SELECT

Consultas aninhadas

- ▶ Se um único atributo com uma única tupla é retornado da consulta externa, então podemos usar o operador =

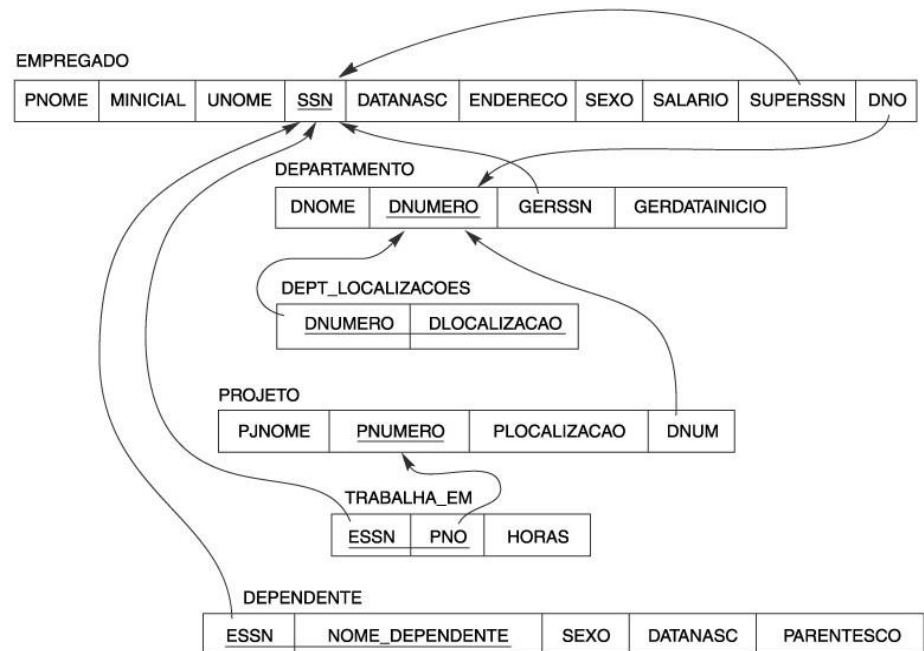
/ Listar o funcionário com o maior salário */*

SELECT pnome,unome,salario

FROM empregado

WHERE salario = (
SELECT max(salario)

FROM empregado)



Consultas aninhadas

EXISTS e NOT EXISTS

- ▶ **EXISTS (subconsulta)**
 - ▶ retorna TRUE se existir ao menos uma tupla no resultado da subconsulta
- ▶ **NOT EXISTS (subconsulta)**
 - ▶ retorna TRUE se subconsulta retornar um conjunto vazio (zero tuplas)
- ▶ **Exemplo:**

```
SELECT col1
FROM tab1
WHERE EXISTS (SELECT 1
              FROM tab2
              WHERE col2 = tab1.col2);
```

Consultas aninhadas

EXISTS e NOT EXISTS

- ▶ O otimizador do SGBD pode executar a consulta apenas até determinar que tem ao menos uma tupla como resultado
- ▶ Como o resultado depende apenas de se há tuplas no resultado, uma convenção comum é escrever as cláusulas exists na forma:
 - ▶ ... EXISTS(SELECT | FROM...WHERE ...)

Exemplo

```
SELECT * FROM produto;
```

| cod_produto | nome | valor |
|-------------|---------------|-------|
| 1 | Queijo | 15.00 |
| 2 | Goiabada | 8.00 |
| 3 | Doce de leite | 7.00 |

```
SELECT * FROM cliente;
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

```
SELECT * FROM compra;
```

| cod_cliente | cod_produto | datahora |
|-------------|-------------|---------------------|
| 1 | 1 | 2016-10-20 00:00:00 |
| 1 | 2 | 2016-10-20 00:00:00 |
| 1 | 3 | 2016-10-20 00:00:00 |
| 2 | 1 | 2016-10-20 00:00:00 |
| 2 | 2 | 2016-10-21 00:00:00 |
| 2 | 2 | 2016-10-22 00:00:00 |

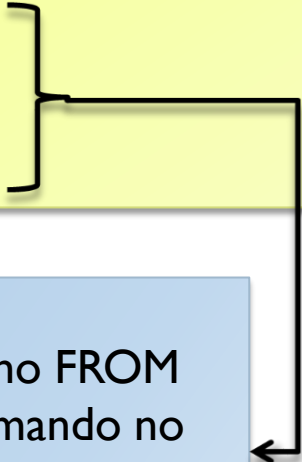
-- Mostrar os clientes que fizeram compras

```
SELECT *  
FROM cliente  
WHERE EXISTS  
    ( SELECT 1  
      FROM compra  
      WHERE compra.cod_cliente =  
            cliente.cod_cliente)
```

Exemplo

-- *Mostrar os clientes que fizeram compras*

```
SELECT *  
FROM cliente  
WHERE EXISTS  
    ( SELECT 1  
      FROM compra  
      WHERE compra.cod_cliente = cliente.cod_cliente)
```



Subconsulta com uma tabela no FROM
(tabela compra) mas com comando no
WHERE que usa uma tabela da consulta
externa



Exemplo

-- *Mostrar os clientes que fizeram compras*

```
SELECT *  
FROM cliente  
WHERE EXISTS  
  ( SELECT 1  
    FROM compra  
    WHERE compra.cod_cliente = cliente.cod_cliente)
```

Para cada tupla da consulta externa, a consulta interna é feita

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

Exemplo

-- *Mostrar os clientes que fizeram compras*

```
SELECT *  
FROM cliente  
WHERE EXISTS  
    ( SELECT 1  
      FROM compra  
      WHERE compra.cod_cliente = cliente.cod_cliente)
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

Como temos 3 clientes, a subconsulta é executada 3 vezes

Exemplo

-- *Mostrar os clientes que fizeram compras*

```
SELECT *  
FROM cliente  
WHERE EXISTS  
    ( SELECT 1  
      FROM compra  
      WHERE compra.cod_cliente = cliente.cod_cliente)
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

Para cada execução da subconsulta
o EXISTS verifica se a subconsulta
retornou algum resultado

Exemplo

```
SELECT * FROM produto;
```

| cod_produto | nome | valor |
|-------------|---------------|-------|
| 1 | Queijo | 15.00 |
| 2 | Goiabada | 8.00 |
| 3 | Doce de leite | 7.00 |

```
SELECT * FROM cliente;
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

```
SELECT * FROM compra;
```

| cod_cliente | cod_produto | datahora |
|-------------|-------------|---------------------|
| 1 | 1 | 2016-10-20 00:00:00 |
| 1 | 2 | 2016-10-20 00:00:00 |
| 1 | 3 | 2016-10-20 00:00:00 |
| 2 | 1 | 2016-10-20 00:00:00 |
| 2 | 2 | 2016-10-21 00:00:00 |
| 2 | 2 | 2016-10-22 00:00:00 |

-- Mostrar os clientes que **NÃO** fizeram compras

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
    ( SELECT 1  
      FROM compra  
      WHERE compra.cod_cliente =  
            cliente.cod_cliente)
```



Exemplo

```
SELECT * FROM produto;
```

| cod_produto | nome | valor |
|-------------|---------------|-------|
| 1 | Queijo | 15.00 |
| 2 | Goiabada | 8.00 |
| 3 | Doce de leite | 7.00 |

-- *Mostrar os clientes que
compraram **todos** os produtos*

```
SELECT * FROM cliente;
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

```
SELECT * FROM compra;
```

| cod_cliente | cod_produto | datahora |
|-------------|-------------|---------------------|
| 1 | 1 | 2016-10-20 00:00:00 |
| 1 | 2 | 2016-10-20 00:00:00 |
| 1 | 3 | 2016-10-20 00:00:00 |
| 2 | 1 | 2016-10-20 00:00:00 |
| 2 | 2 | 2016-10-21 00:00:00 |
| 2 | 2 | 2016-10-22 00:00:00 |



Exemplo

```
SELECT * FROM produto;
```

| cod_produto | nome | valor |
|-------------|---------------|-------|
| 1 | Queijo | 15.00 |
| 2 | Goiabada | 8.00 |
| 3 | Doce de leite | 7.00 |

```
SELECT * FROM cliente;
```

| cod_cliente | nome | data_nasc |
|-------------|--------|------------|
| 1 | Maria | 2000-01-05 |
| 2 | Ana | 1998-05-03 |
| 3 | Carlos | 1990-02-02 |

```
SELECT * FROM compra;
```

| cod_cliente | cod_produto | data |
|-------------|-------------|-----------|
| 1 | 1 | 2016-10-2 |
| 1 | 2 | 2016-10-2 |
| 1 | 3 | 2016-10-2 |
| 2 | 1 | 2016-10-2 |
| 2 | 2 | 2016-10-2 |
| 2 | 2 | 2016-10-2 |

-- *Mostrar os clientes que compraram **todos** os produtos*

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
  
    (SELECT cod_produto  
     FROM produto  
  
     EXCEPT  
  
     SELECT cod_produto  
     FROM compra  
     WHERE compra.cod_cliente =  
           cliente.cod_cliente  
    )
```


-- *Mostrar os clientes que compraram **todos** os produtos*

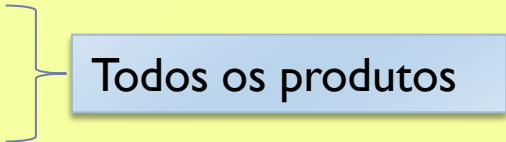
```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
  
    (SELECT cod_produto  
     FROM produto  
  
     EXCEPT  
  
     SELECT cod_produto  
     FROM compra  
     WHERE compra.cod_cliente = cliente.cod_cliente  
    )
```

Subconsulta
correlacionada: ela é
executada para cada
cliente existente na
base



-- *Mostrar os clientes que compraram **todos** os produtos*

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
  
  (SELECT cod_produto  
   FROM produto  
  
   EXCEPT  
  
   SELECT cod_produto  
   FROM compra  
   WHERE compra.cod_cliente = cliente.cod_cliente  
  )
```



Todos os produtos



-- *Mostrar os clientes que compraram **todos** os produtos*

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
  
  (SELECT cod_produto  
   FROM produto  
  
   EXCEPT  
  
   SELECT cod_produto  
   FROM compra  
   WHERE compra.cod_cliente = cliente.cod_cliente  
  )
```

Produtos para o
cliente “atual”



-- *Mostrar os clientes que compraram **todos** os produtos*

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS  
  
  (SELECT cod_produto  
   FROM produto  
  
   EXCEPT  
   )  
  SELECT cod_produto  
  FROM compra  
  WHERE compra.cod_cliente = cliente.cod_cliente  
  )
```

Ao subtrair o conjunto de todos os produtos dos produtos que o cliente comprou, teremos uma resposta vazia caso ele tenha comprado todos



-- *Mostrar os clientes que compraram **todos** os produtos*

```
SELECT *  
FROM cliente  
WHERE NOT EXISTS }  
  
    (SELECT cod_produto  
     FROM produto  
  
     EXCEPT  
  
     SELECT cod_produto  
     FROM compra  
     WHERE compra.cod_cliente = cliente.cod_cliente  
    )
```

NOT EXISTS retornará **TRUE** caso a subconsulta seja vazia, ou seja, caso o cliente tenha comprado todos os produtos



Comando SELECT

Consultas aninhadas correlacionadas

- ▶ Consultas aninhadas correlacionadas ocorrem quando o resultado da subconsulta (consulta interna) muda de acordo com a tupla que está sendo avaliada na consulta externa.
- ▶ Em outras palavras, sempre que uma condição na cláusula WHERE de uma consulta aninhada referenciar algum atributo de uma relação declarada na consulta externa.
- ▶ Cuidado: isso pode ser muito lento, pois a subconsulta é reavaliada para cada linha da consulta externa. Se houver forma de evitar isso, sua consulta pode ser mais rápida



Comando SELECT

Itens do FROM

- ▶ É possível realizar consultas sobre os resultados obtidos em outras consultas. Isso pode ser feito adicionando a consulta na cláusula FROM
 - ▶ Relembrando a sintaxe

onde item_do_from pode ser um entre:

```
[ ONLY ] nome_da_tabela [ * ] [ [ AS ] alias [ ( alias_de_coluna [, ...] ) ] ]  
( seleção ) [ AS ] alias [ ( alias_de_coluna [, ...] ) ]  
nome_da_função ( [ argumento [, ...] ] ) [ AS ] alias [ ( alias_de_coluna [, ...]  
| definição_de_coluna [, ...] ) ]  
nome_da_função ( [ argumento [, ...] ] ) AS ( definição_de_coluna [, ...] )  
item_do_from [ NATURAL ] tipo_de_junção item_do_from [ ON condição_de_junção  
| USING ( coluna_de_junção [, ...] ) ]
```



Comando SELECT

Itens do FROM

► Exemplo

/ Listar os estados com 3 filiais (sem usar having)*/*

SELECT *

FROM (

SELECT estado, count(*) as nfiliais

FROM filiais

GROUP BY estado) **AS** t

WHERE t.nfiliais = 3



Modelo Relacional (Empresa)

- ▶ Departamento(Dnome, **Dnumero**, Cpf_gerente, Data_inicio_ger)
- ▶ Funcionario(Pnome, Minicial, Unome, **Cpf**, Data_nasc, Endereco, Sexo, Salario, Cpf_supervisor, Dnr)
- ▶ Localizacao_Dep(**Dnumero**, **Dlocal**)
- ▶ Projeto(Proj_nome, **Proj_numero**, Proj_local, Dnum)
- ▶ Trabalha_Em(**Fcpf**, **Pnr**, Horas)
- ▶ Dependente(**Fcpf**, **Nome dependente**, Sexo, Data_nasc, Parentesco)

Legenda:

Chave primária

Chave estrangeira

Chave primária e estrangeira

Referências

- ▶ Slides adaptados da aula da Profa. Josiane M. Bueno (*in memoriam*)
- ▶ Slides Prof. Humberto Razente
- ▶ Slides Profa. Sandra de Amo

