O operador *EXISTS*

- Exemplo 1:

```
SELECT nomeanimal FROM animais a
WHERE
 EXISTS (Select * From especies e
          Where e.codespecie=a.codespecie
          And e.nome='Hipopótamo')
```

  - Relacionamento entre tabelas ocorre dentro da subconsulta
  - Quando a subconsulta retorna pelo menos 1 tupla, o EXISTS retorna verdadeiro e habilta o retorna da tupla respectiva na expressão principal
  - Neste exemplo estamos retornando nomes de animais da espécie Hipopótamo

O operador *EXISTS*

Exemplo 1:

```
SELECT nomeanimal FROM animais a
WHERE
 EXISTS (Select * From especies e
          Where e.codespecie=a.codespecie
          And e.nome='Hipopótamo')
```

cod-animal	nome-animal	cod-especie
1	Glória	1
2	Vermelho	1
3	Jaja	1

cod-espécie	nome-especie	expectativa-vida
1	Hipopótamo	10
2	Coelho	3

– Neste exemplo, para cada animal EXISTE uma espécie chamada Hipopótamo que possui código igual ao código de espécie de cada animal.

- Logo, o resultado é:

nomeanimal
Glória
Vermelho
Jaja

O operador *EXISTS*

Exemplo 2:

```
SELECT nomeanimal FROM animais a
WHERE
 EXISTS (Select * From especies e
          Where e.codespecie=a.codespecie
          And e.nome='Coelho')
```

cod-animal	nome-animal	cod-especie
1	Glória	1
2	Vermelho	1
3	Jaja	1

cod-espécie	nome-especie	expectativa-vida
1	Hipopótamo	10
2	Coelho	3

– Neste exemplo, para cada animal NÃO EXISTE uma espécie chamada Coelho que possui código igual ao código de espécie de cada animal.

- Logo, nenhum animal é retornado na resposta.

O operador *NOT EXISTS*

Exemplo 3:

```
SELECT nomeespecie FROM especies e1
WHERE
 NOT EXISTS (Select * From especies e2
             Where e2.expectativavida>e1.expectativavida);
```

- Neste exemplo, para cada espécie verifica se NÃO EXISTE uma outra espécie com expectativa maior que a dele:

  - Para e1.codespecie=1 a subconsulta não acha um e2 com expectativa > 10. Logo, como a subconsulta não retorna nada (0=FALSE), o NOT do (NOT EXISTS) aplicará !(FALSE) = TRUE. Isto é, fará e1 (Hipopótamo) retornar.

	cod-espécie	nome-especie	expectativa-vida
e1->	1	Hipopótamo	10
e2->	2	Coelho	3

Resultado:

nomeespecie
Hipopótamo

O operador *NOT EXISTS*

Exemplo 3:

```
SELECT nomeespecie FROM especies e1
WHERE
 NOT EXISTS (Select * From especies e2
             Where e2.expectativavida>e1.expectativavida);
```

- Neste exemplo, para cada espécie verifica se NÃO EXISTE uma outra espécie com expectativa maior que a dele:

  - Para e1.codespecie=2 a subconsulta acha um e2 com expectativa > 3. Logo, como a subconsulta retorna algo (>0=TRUE), o NOT do (NOT EXISTS) aplicará !(TRUE) = FALSE. Isto é, fará e1 (Coelho) não retornar.

	cod-espécie	nome-especie	expectativa-vida
e2->	1	Hipopótamo	10
e1->	2	Coelho	3

Resultado:

nomeespecie
Hipopótamo

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
            NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal);
```

- Neste exemplo, procura-se funcionários que tenham consultado **todos os animais**. Isto é, funcionários para os quais NÃO EXISTE Nenhum animal que ele NÃO tenha consultado.
  - **NOT com NOT = TRUE**

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

c->

cod-animal	matricula	
1	111	TRUE
2	111	
1	222	
3	333	
3	111	
2	333	

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
     NOT EXISTS(Select * From consultas c Where
          c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

!TRUE

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

!TRUE

c->

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

TRUE

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
           NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal	
1	Glória	!TRUE
2	Vermelho	!TRUE
3	Jaja	

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal	
1	Glória	!TRUE
2	Vermelho	!TRUE
3	Jaja	

c->

cod-animal	matricula	
1	111	
2	111	
1	222	
3	333	
3	111	TRUE
2	333	

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
           NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal	
1	Glória	!TRUE
2	Vermelho	!TRUE
3	Jaja	!TRUE

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
            NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

!FALSE

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

c->

cod-animal	matricula	
1	111	
2	111	
1	222	TRUE
3	333	
3	111	
2	333	

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
           NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

!TRUE

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal	
1	Glória	!TRUE
2	Vermelho	
3	Jaja	

	cod-animal	matricula	
c->	1	111	FALSE
c->	2	111	FALSE
c->	1	222	FALSE
c->	3	333	FALSE
c->	3	111	FALSE
c->	2	333	FALSE

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
           NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória !TRUE
2	Vermelho !FALSE
3	Jaja

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
            NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

!TRUE

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
        NOT EXISTS(Select * From consultas c Where
        c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

	cod-animal	matricula	
c->	1	111	FALSE
c->	2	111	FALSE
c->	1	222	FALSE
c->	3	333	FALSE
c->	3	111	FALSE
c->	2	333	FALSE

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
     NOT EXISTS(Select * From consultas c Where
          c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

a->

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

!FALSE

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

O operador *NOT EXISTS*

Exemplo 4: Encadeando múltiplos NOT EXISTS

```
SELECT nome FROM funcionarios f WHERE
NOT EXISTS(Select * From animais a Where
           NOT EXISTS(Select * From consultas c Where
                c.matricula=f.matricula And c.codanimal=a.codanimal)
```

f->

matricula	nome
111	Vera
222	Lúcia
333	Rita

!TRUE

cod-animal	nome-animal
1	Glória
2	Vermelho
3	Jaja

cod-animal	matricula
1	111
2	111
1	222
3	333
3	111
2	333

Resultado:

nome
Vera

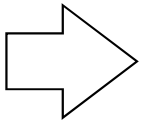
Subconsulta na Cláusula FROM

- Gera uma tabela derivada a partir de uma ou mais tabelas, para uso na consulta externa
  - otimização: filtra linhas e colunas de uma tabela que são desejadas pela consulta externa

```
select lista_atributos
from (consulta_SQL) as nome_tabela_derivada
```

- Mapeamento para a álgebra relacional

select a_1
from (select x
from t_1 where d > 5)
as t_2 join t_3
on $t_3.c = t_2.x$



$$\pi_{a1} (t_3 \ \theta \ X \ \rho_{t2}(\pi_x (\sigma_{d > 5} (t_1))))$$

$$\theta = t_3.c = t_2.x$$

Exemplos

Álgebra

$\pi_{\text{Médicos.codm, ..., nroa, hora}}$ (

(Médicos θ X

θ = Médicos.codm = C.codm

ρ_{C} ($\pi_{\text{codm, hora}}$ ($\sigma_{\text{data = '06/11/13'}}$ (Consultas)))))

$\pi_{\text{Amb.nroa, andar, capacidade}}$ (

ρ_{Amb} ($\pi_{\text{nroa, andar}}$ (Ambulatórios)) θ X

θ = Amb.nroa = M_ort.nroa

ρ_{MFlo} (π_{nroa} ($\sigma_{\text{cidade = 'Fpolis'}}$ (Médicos))))

SQL

```
select Medicos.*, C.hora
from Medicos join
 (select codm, hora
  from Consultas
  where data = '06/11/13')
  as C
on Médicos.codm = C.codm

select Amb.*
from (select nroa, andar from
ambulatorios) as Amb join
 (select nroa from Medicos
  where cidade = 'Fpolis')
  as MFlo
on Amb.nroa = MFlo.nroa
```

Atualização com Consulta

- Comandos de atualização podem incluir comandos de consulta
  - necessário toda vez que a atualização deve testar relacionamentos entre tabelas
- Exemplo 1

```
delete from Consultas
where hora > '17:00:00'
and codm in (select codm
             from Médicos
             where nome = 'Maria')
```

Atualização com Consulta

- Exemplo 2

```
update Médicos
set nroa = NULL
where not exists
  (select * from Médicos m
   where m.codm <> Médicos.codm
   and m.nroa = Médicos.nroa)
```

- Exemplo3

```
update Ambulatórios
set capacidade = capacidade +
      (select capacidade
       from Ambulatórios where nroa = 4)
where nroa = 2
```

Atualização com Consulta

- Exemplo 4 (supondo MedNovos(código, nome, especialidade))

```
insert into MedNovos
   select codm, nome, especialidade
   from Médicos
   where idade < 21;
```

Ordenação de Resultados

- Cláusula ORDER BY

```
select lista_atributos
from lista_tabelas
[where condição]
[order by nome_atributo 1 [desc] {[,
  nome_atributo n [desc]]} ]
```

- Exemplos

```
select *                    select salário, nome
from Pacientes              from Funcionários
order by nome               order by salário desc, nome
```

Ordenação de Resultados

- É possível determinar a quantidade de valores ordenados a retornar

```
select ...
 limit valor1 [,valor2]
```

- Exemplos

retorna as 5 primeiras tuplas

```
select *
from Pacientes
order by nome
limit 5
```

retorna tuplas 6 a 15

```
select salário, nome
from Funcionários
order by salário desc,
         nome
limit 5,10
```

Definição de Grupos

- Cláusula GROUP BY

```
select lista_atributos
from lista_tabelas
[where condição]
[group by lista_atributos_agrupamento
    [having condição_para_agrupamento] ]
```

- GROUP BY
  - define grupos para combinações de valores dos atributos definidos em *lista_atributos_agrupamento*
  - apenas atributos definidos em *lista_atributos_agrupamento* podem aparecer no resultado da consulta
  - geralmente o resultado da consulta possui uma função de agregação

Definição de Grupos

- Exemplo

```
select especialidade, count(*)
from Médicos
group by especialidade
```

especialidade	*Count*
ortopedia	2
pediatira	1
neurologia	1
traumatologia	3

especialidade				“grupos”		
ortopedia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	1	João	40	1000010000	Fpolis	1
	4	Carlos	28	1100011000	Joinville	
pediatria	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	3	Pedro	51	1100010000	Fpolis	2
neurologia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	5	Márcia	33	1100011100	Biguaçu	3
traumatologia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	2	Maria	42	1000011000	Blumenau	2
	6	Joana	37	1111110000	Fpolis	3
	7	Mauro	53	1111000011	Blumenau	2

Definição de Grupos

- Cláusula HAVING
  - define condições para que grupos sejam formados
    - condições só podem ser definidas sobre atributos do agrupamento ou serem funções de agregação
  - existe somente associada à cláusula GROUP BY
- Exemplos

```
select especialidade, count(*)
from Médicos
group by especialidade
having count(*) > 1
```