

DCC011

Introdução a Banco de Dados

SQL select avançado

Mirella M. Moro

Departamento de Ciência da Computação

Universidade Federal de Minas Gerais

mirella@dcc.ufmg.br

Revisão: consultas básicas

- Formato básico do comando SELECT:

SELECT <lista de atributos>
FROM <lista de tabelas>
[**WHERE** <condição>;]

- **SELECT** BDATE, ADDRESS
FROM EMPLOYEE
WHERE FNAME='John' **AND** MINIT='B' **AND**
LNAME='Smith';

EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)
superssn REFERENCIA EMPLOYEE
dno REFERENCIA DEPARTMENT
DEPARTMENT(dnum, dname, mgrssn, mgrinitialdate)
mgrssn REFERENCIA EMPLOYEE.ssn
PROJECT(pnumber, pname, plocation, dnum)
dnum REFERENCIA DEPARTMENT
DEPT_LOCATIONS(dnumber, dlocation)
dnumber REFERENCIA DEPARTMENT

$\pi_{Bdate, Address} \sigma_{Fname='John' \text{ AND } Minit='B' \text{ AND } Lname='Smith'} (EMPLOYEE)$

Revisão: consultas básicas

- **SELECT** SSN, LNAME, SALARY
FROM EMPLOYEE;
→ PROJETA COLUNAS
→ DE TABELA
- **SELECT ***
FROM EMPLOYEE;
→ PROJETA TODAS COLUNAS
- **SELECT DISTINCT** SALARY
FROM EMPLOYEE;
→ SQL **NÃO** ELIMINA DUPLICATAS POR DEFAULT
- **SELECT** LNAME, DNAME
FROM EMPLOYEE, DEPARTMENT;
→ PRODUTO CARTESIANO
- **SELECT** PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE PLOCATION='Stafford' **AND**
DNUM=DNUMBER AND MGRSSN=SSN;
→ CONDIÇÃO DE SELEÇÃO
→ JUNÇÃO NO WHERE
- **SELECT** PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM (PROJECT **JOIN** DEPARTMENT **ON** DNUM=DNUMBER)
JOIN EMPLOYEE **ON** MGRSSN=SSN
WHERE PLOCATION='Stafford';
→ JUNÇÃO NO FROM
→ CONDIÇÃO DE SELEÇÃO

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

CURSOS

Cr	Nome	Depto	Coord
CC	Ciência da Computação	DCC	RG
MC	Matemática Computacional	DCC	TN
SI	Sistemas de Informação	DCC	CDJ
ECA	Engenharia de Controle e Automação	ENG	XYZ

MATRICULAS

Matr	Disc	T	Sem
1	DCC011	Z	20162
1	DCC851	A	20162
1	DCC834	A	20161
2	DCC011	Z	20161
...

CONSULTAS ANINHADAS E CORRELACIONADAS

Até agora, apenas uma cláusula **SELECT** por consulta
MAS podemos ter mais de uma

select Matr, Nome

from ALUNOS

where Cr **IN**

(**select** Cr

from CURSOS

where Depto = 'DCC')

ALUNOS				CURSOS			
Matr	Nome	Sexo	Cr	Cr	Nome	Depto	Coord
1	A	F	CC	CC	Ciência da Computação	DCC	RG
2	B	M	CC	MC	Matemática Computacional	DCC	TN
3	C	M	CC	SI	Sistemas de Informação	DCC	CDJ
4	D	F	MC	ECA	Engenharia de Controle e Automação	ENG	XYZ
5	E	M	MC				
6	F	M	SI				
7	G	F	SI				
8	H	F	SI				
9	I	M	SI				
10	J	M	ECA				

MATRICULAS			
Matr	Disc	T	Sem
1	DCC011	Z	20162
1	DCC851	A	20162
1	DCC834	A	20161
2	DCC011	Z	20161
...

Processamentos diferentes

Equivalência no resultado??

```
select Matr, Nome
from ALUNOS
where Cr IN
(select Cr
 from CURSOS
 where Depto = 'DCC')
```

1. Resultado 2º *select* na memória
2. Processa 1º *select-from-where*

Chamam-se consulta **interna** (ou **subconsulta**) e consulta **externa** respectivamente

```
select Matr, Nome
from ALUNOS A, CURSOS C
where A.Cr=C.Cr
      AND Depto= 'DCC'
```

1. Produto cartesiano na memória (*from*)
2. Processa *where*
3. Projeta *select* para linhas *where true*

CONSULTAS ANINHADAS E CORRELACIONADAS

```
select Matr, Nome  
from ALUNOS natural join NOTASFINAIS  
where Nota > ANY  
(select Nota from NOTASFINAIS  
  where Matr=1)
```

NOTASFINAIS

Matr	Disc	Sem	Nota
1	DCC011	20162	75
1	DCC851	20162	78
1	DCC834	20161	70
2	DCC011	20161	85
2	DCC851	20171	88
2	DCC834	20171	92
3	DCC011	20161	71
3	DCC834	20162	60
...

CONSULTAS ANINHADAS E CORRELACIONADAS

```
select Matr, Nome
from ALUNOS natural join NOTASFINAIS
where Nota > ALL
(select Nota from NOTASFINAIS
 where Matr=1)
```

```
select Matr, Nome
from ALUNOS natural join NOTASFINAIS
where Nota > -- SEM O ALL
(select Nota from NOTASFINAIS where Matr=1)
```

```
select Matr, Nome
from ALUNOS natural join NOTASFINAIS
where Nota > -- COM MAX
(select max(Nota) from NOTASFINAIS where Matr=1)
```

NOTASFINAIS

Matr	Disc	Sem	Nota
1	DCC011	20162	75
1	DCC851	20162	78
1	DCC834	20161	70
2	DCC011	20161	85
2	DCC851	20171	88
2	DCC834	20171	92
3	DCC011	20161	71
3	DCC834	20162	60
...

CONSULTAS ANINHADAS E CORRELACIONADAS → consulta interna acessa externa

```
select Nome
from CURSOS C
where EXISTS
(select * from ALUNOS A
 where A.Cr=C.Cr
 AND Sexo='F');
```

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

Cr	Nome	Depto	Coord
CC	Ciência da Computação	DCC	RG
MC	Matemática Computacional	DCC	TN
SI	Sistemas de Informação	DCC	CDJ
ECA	Engenharia de Controle e Automação	ENG	XYZ

Matr	Disc	T	Sem
1	DCC011	Z	20162
1	DCC851	A	20162
1	DCC834	A	20161
2	DCC011	Z	20161
...

```
select Nome from CURSOS C
where NOT EXISTS
(select * from ALUNOS A
 where A.Cr=C.Cr AND Sexo='F');
```

CONSULTAS ANINHADAS E CORRELACIONADAS → consulta interna acessa externa

```
select Nome
from CURSOS C
where EXISTS
(select * from ALUNOS A
 where A.Cr=C.Cr
 AND Sexo='F');
```

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

Cr	Nome	Depto	Coord
CC	Ciência da Computação	DCC	RG
MC	Matemática Computacional	DCC	TN
SI	Sistemas de Informação	DCC	CDJ
ECA	Engenharia de Controle e Automação	ENG	XYZ

Matr	Disc	T	Sem
1	DCC011	Z	20162
1	DCC851	A	20162
1	DCC834	A	20161
2	DCC011	Z	20161
...

1. Carrega a tabela do FROM externo na memória
2. Para cada linha desta tabela, verifica se o resultado da consulta interna existe (ou seja, não é vazio e não é NULL)
3. Se resultado interno existe, retorna a projeção do SELECT externo

Ou seja

1. Para cada linha da tabela CURSOS
2. Se existe um aluno cujo A.Cr seja igual ao seu Cr e Sexo seja F
3. Então retorna o Nome do curso

Processamentos diferentes

Equivalência no resultado??

```
select Nome
from CURSOS C
where EXISTS
(select * from ALUNOS A
 where A.Cr=C.Cr
 AND Sexo='F');
```

1. Para cada linha da consulta externa
2. Processa consulta interna
3. Se resultado não vazio, retorna a projeção externa

```
select C.Nome
from CURSOS C, ALUNOS A
where A.Cr=C.Cr
 AND Sexo='F';
```

1. Produto cartesiano na memória (*from*)
2. Processa *where*
3. Projeta *select* para linhas *where true*

OU SEJA, vai retornar várias vezes os mesmos nomes de cursos!!!! Uma vez para cada menina matriculada

CLÁUSULA GROUP BY

Como saber quantas pessoas estão em cada curso?!

Vamos pensar na lógica

1. Temos de contar os alunos
2. Temos de contar separado por curso

O resultado seria este:

Cr	NroAlunos
CC	3
MC	2
SI	4
ECA	1

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

CLÁUSULA GROUP BY

1.select count(*) from ALUNOS

-- retorna o número total de alunos (10)

2.select count(*) from ALUNOS where Cr='CC'

-- retorna o número total de alunos apenas de um curso (3)

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

CLÁUSULA GROUP BY

select Cr, **count**(*) as NroAlunos **from** ALUNOS
group by Cr

Cr	NroAlunos
CC	3
MC	2
SI	4
ECA	1

IMPORTANTE

Coluna do group by
está no select,
por quê?

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

**select Cr, count(*) as NroAlunos from ALUNOS
group by Cr**

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA


Cr	NroAlunos
CC	3
MC	2
SI	4
ECA	1

select Sexo, **count**(*) as NroAlunos **from** ALUNOS
group by Sexo

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

Sexo	NroAlunos
F	4
M	6



select Sexo, **count**(*) as NroAlunos **from** ALUNOS
group by Sexo

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I		SI
10	J		ECA

Sexo	NroAlunos
F	4
M	4
null	2

select Cr, **Sexo**, **count(*)** as NroAlunos **from** ALUNOS
group by Cr, Sexo

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

Cr	Sexo	NroAlunos
CC	F	1
CC	M	2
MC	F	1
MC	M	1
SI	F	2
SI	M	2
ECA	M	1

select Cr, **count**(*) as NroAlunos **from** ALUNOS
where Sexo = "M" **group by** Cr

ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA

Cr	NroAlunos
CC	2
MC	1
SI	2
ECA	1

1. Define as linhas **–from**
2. Filtra as linhas **–where**
3. Cria os grupos **–group by**
4. Apresenta um resultado por grupo **–select**

select Cr, **count**(*) as NroAlunos **from** ALUNOS
where Sexo = "F" **group by** Cr


ALUNOS

Matr	Nome	Sexo	Cr
1	A	F	CC
2	B	M	CC
3	C	M	CC
4	D	F	MC
5	E	M	MC
6	F	M	SI
7	G	F	SI
8	H	F	SI
9	I	M	SI
10	J	M	ECA


Cr	NroAlunos
CC	1
MC	1
SI	2

ECA não entrou no resultado,
por quê?

CLÁUSULA GROUP BY

Como saber quantas pessoas estão em cada curso 
Apenas cursos com mais de 3 matrículas???

CLÁUSULA GROUP BY

Como saber quantas pessoas estão em cada curso 
Apenas cursos com mais de 3 matrículas???

```
select Cr, count(*) as NroAlunos from ALUNOS  
group by Cr  
having NroAlunos > 3
```

Ou seja, **having** adiciona uma condição para o grupo entrar no resultado da consulta!

Ainda de outra forma: apenas grupos com **having true** entram no resultado

NULL representa valores UNKNOWN

(pra quem acha que sabe SQL)

- Uma operação aritmética que envolve NULL, retorna NULL.
 - coluna minus NULL → NULL
(e não zero)
- Comparação booleana com NULL retorna UNKNOWN
(e não true/false)
 - NULL = NULL → UNKNOWN
 - valor {<>, >, <, >=, <=} NULL → UNKNOWN
- Testar se um valor é NULL
 - where valor **IS NULL**
- Testar se um valor não é NULL
 - where valor **IS NOT NULL**
- SQL SELECT retorna valores para consultas cujo WHERE seja TRUE
- SQL GROUP retorna grupos para consultas cujo HAVING seja TRUE
- A função de agregação COUNT(*) conta todas as tuplas
- Já COUNT(coluna) conta as tuplas cujo valor para coluna não seja NULL
- Outras funções de agregação ignoram valores NULL

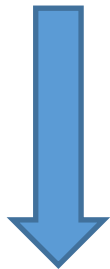
FONTE: <http://www-cs-students.stanford.edu/~wlam/compsci/sqlnulls> [@10/2017]
pesquisado em **Date&Darwen** *A Guide to the SQL Standard*. Fourth edition,
Addison-Wesley, Reading, Massachusetts, 1997. (ISBN 0-201-96426-0)

Ou seja

EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)
superssn REFERENCIA EMPLOYEE
dno REFERENCIA DEPARTMENT

Retorne os dados dos empregados que não possuem supervisor.
Duas possíveis soluções:

select * from EMPLOYEE
where **SUPERSSN IS NULL**



select * from EMPLOYEE
where **NULL IN**
(select SUPERSSN from
EMPLOYEE)



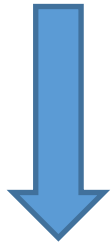
FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS		SEX	SALARY	SUPERSSN	DNO
JAMES	E	BORG	888665555	1937-11-10	450 STONE, H	STON, TX	M	55000	null	1

Empty set

RESULTADO VAZIO é diferente de RESULTADO NULL

EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)
superssn REFERENCIA EMPLOYEE
dno REFERENCIA DEPARTMENT

```
select SUPERSSN  
from EMPLOYEE  
where FNAME = "JAMES"
```



JAMES não tem supervisor!!
A consulta é válida e retorna NULL

SUPERSSN
null

```
select * from EMPLOYEE  
where NULL IN  
(select SUPERSSN from  
EMPLOYEE)
```



JAMES não tem supervisor!!
Mas a consulta não é válida porque
NULL IN retorna UNKNOWN

Empty set
.

Mais sobre NULL

Resultado **vazio**: não retorna linhas

Resultado **null**: retorna linha(s) com valor(es) null

```
CREATE TABLE R (a INTEGER);
```

```
select MAX(a) from R;
```

Qual o resultado considerando: R é vazia, R tem apenas uma linha com NULL, R contém NULL e valores inteiros?

- Se R é vazia: consulta retorna NULL.
- Se R tem apenas uma linha com NULL: NULL.
- Se R contém valores NULL e não-NULL, os NULL são ignorados, e MAX() retorna o maior inteiro.

Mais sobre NULL

Resultado **vazio**: não retorna linhas

Resultado **null**: retorna linha(s) com valor(es) null

```
CREATE TABLE R (a INTEGER);
```

```
select DISTINCT a from R
```

```
  where a >= ALL (select * from R);
```

Qual o resultado considerando: R é vazia, R contém NULL e valores inteiros?

- Tem lógica, **mas falha no SQL**.
- Se R é vazia: retorna vazio. A condição $\geq \text{ALL}$ é verdadeira para subconsulta vazia, mas não existe valor em a para executar o teste.
- Se R contém um valor NULL, a consulta retorna vazio porque o teste $a \geq \text{ALL}(\dots)$ retorna UNKNOWN para qualquer NULL, ou máximo valor inteiro não NULL de a se a subconsulta inclui um valor NULL.

Mais sobre NULL

Resultado **vazio**: não retorna linhas

Resultado **null**: retorna linha(s) com valor(es) null

CREATE TABLE R (a INTEGER);

(select DISTINCT * from R) **EXCEPT**

(select R.a from R, R as S where R.a < S.a);

Qual o resultado considerando: R é vazia, R contém NULL e valores inteiros?

- Se R é vazia, a consulta retorna vazio.
- Se R contém um valor NULL, a consulta retorna NULL, além do valor máximo inteiro de R. A consulta interna nunca inclui NULL, então NULL nunca é subtraído de R.

NA PRÁTICA (1)

PostgreSQL 7.2.2, Oracle 9i

- Where SQL mandates a behavior for a query above, PostgreSQL/Oracle complies.
- **MAX() of NULLs only**: returns NULL (consistent with ignoring NULLs, then computing the MAX() of an empty remainder).

FONTE: <http://www-cs-students.stanford.edu/~wlam/compsci/sqlnulls> [@10/2017]
pesquisado em **Date&Darwen** *A Guide to the SQL Standard*. Fourth edition, Addison-Wesley, Reading, Massachusetts, 1997. (ISBN 0-201-96426-0)

NA PRÁTICA (2)

P_Id	PName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

**SELECT PName, UnitPrice * (UnitsInStock + UnitsOnOrder)
FROM Products;**

→ O resultado é NULL se qualquer valor UnitsOnOrder é NULL

SGBDs resolve este problema de maneiras diferentes
Veja nos slides a seguir

FONTE: https://www.w3schools.com/sql/sql_isnull.asp [@10/2017]

NA PRÁTICA (2)

P_Id	PName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

MySQL

- IFNULL(): permite retornar um valor alternativo caso uma expressão seja NULL

```
SELECT PName, UnitPrice * (UnitsInStock +  
                                IFNULL(UnitsOnOrder, 0))  
FROM Products;
```

- COALESCE(): retorna o primeiro valor não NULL de uma lista

```
SELECT PName, UnitPrice * (UnitsInStock +  
                                COALESCE(UnitsOnOrder, 0))  
FROM Products;
```

NA PRÁTICA (2)

P_Id	PName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

SQL SERVER

- ISNULL(): permite retornar um valor alternativo caso uma expressão seja NULL

```
SELECT PName, UnitPrice * (UnitsInStock +  
                                ISNULL(UnitsOnOrder, 0))
```

```
FROM Products
```

ORACLE

- NVL: permite o mesmo

```
SELECT PName, UnitPrice * (UnitsInStock +  
                                NVL(UnitsOnOrder, 0))
```

```
FROM Products
```


Mais exemplos do livro

CONSULTAS ANINHADAS

Realiza *fetch* valores existentes

(armazena resultado da consulta interna na memória e depois processa a consulta externa)

```
SELECT FNAME, LNAME, ADDRESS  
FROM EMPLOYEE  
WHERE DNO IN (SELECT DNUMBER  
                FROM DEPARTMENT  
                WHERE DNAME='Research');
```

Nota: equivalente porque o resultado é o mesmo, embora o processamento seja diferente

é equivalente à consulta

```
SELECT FNAME, LNAME, ADDRESS  
FROM EMPLOYEE, DEPARTMENT  
WHERE DNO=DNUMBER AND DNAME='Research';
```

```
EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)  
    superssn REFERENCIA EMPLOYEE  
    dno REFERENCIA DEPARTMENT  
DEPARTAMENT (dnum, dname, mgrssn, mgrinitialdate)  
    mgrssn REFERENCIA EMPLOYEE.ssn
```

COMPARAÇÃO DE CONJUNTOS: IN

EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)
superssn REFERENCIA EMPLOYEE
dno REFERENCIA DEPARTMENT
DEPARTMENT (dnum, dname, mgrssn, mgrinitialdate)
mgrssn REFERENCIA EMPLOYEE.ssn
PROJECT (pnumber, pname, plocation, dnumber)
dnumber REFERENCIA DEPARTMENT
WORKS_ON (essn, pno, hours)
essn REFERENCIA EMPLOYEE
pno REFERENCIA PROJECT

(Q4A)
SELECT DISTINCT PNUMBER
FROM PROJECT
WHERE PNUMBER IN (SELECT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM = DNUMBER AND
MGRSSN = SSN AND
LNAME = 'Smith')
OR
PNUMBER IN (SELECT PNO
FROM WORKS_ON, EMPLOYEE
WHERE ESSN = SSN AND
LNAME = 'Smith');

A primeira consulta seleciona números de projetos que tem Smith como gerente; a segunda consulta seleciona número dos projetos que tem Smith como empregado.

COMPARAÇÃO DE CONJUNTOS: IN

```
SELECT DISTINCT ESSN  
FROM WORKS_ON  
WHERE (PNO, HOURS) IN (SELECT PNO, HOURS  
                           FROM WORKS_ON  
                           WHERE ESSN='123456789');
```

O esquema deve ser o mesmo: antes do IN e do SELECT interno
Neste caso, o “par” {pno, hours} deve ser o mesmo para a consulta
retornar o resultado esperado

```
EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)  
  superssn REFERENCIA EMPLOYEE  
  dno REFERENCIA DEPARTMENT  
WORKS_ON(essn, pno, hours)  
  essn REFERENCIA EMPLOYEE  
  pno REFERENCIA PROJECT
```

COMPARAÇÃO DE CONJUNTOS: ALL

```
SELECT LNAME, FNAME FROM EMPLOYEE
WHERE SALARY > ALL (SELECT SALARY
                     FROM EMPLOYEE
                     WHERE DNO=5);
```

Equivalentes??????

```
SELECT LNAME, FNAME FROM EMPLOYEE
WHERE SALARY > (SELECT MAX (SALARY) FROM EMPLOYEE
                WHERE DNO=5);
```

```
SELECT LNAME, FNAME FROM EMPLOYEE
WHERE SALARY > (SELECT SALARY FROM EMPLOYEE
                WHERE DNO=5);
```

EMPLOYEE(ssn, fname, lname, address, bdate, superssn, dno)
 superssn REFERENCIA EMPLOYEE
 dno REFERENCIA DEPARTMENT
WORKS_ON(essn, pno, hours)
 essn REFERENCIA EMPLOYEE
 pno REFERENCIA PROJECT

CONSULTAS CORRELACIONADAS

FUNÇÃO EXISTS / NOT EXISTS

(Q16B)

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE EXISTS (SELECT *
              FROM DEPENDENT
              WHERE E.SSN=ESSN AND
                    E.SEX=SEX AND
                    E.FNAME=DEPENDENT_NAME);
```

(Q6)

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS (SELECT *
                  FROM DEPENDENT
                  WHERE SSN=ESSN);
```

As consultas interna e externa são processadas em paralelo. EXISTS é uma função condicional da cláusula WHERE: para cada linha da consulta externa, verifica se a interna retorna algum resultado válido.

CONSULTAS CORRELACIONADAS

OPERADOR CONTAINS

SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE ((**SELECT** PNO
 FROM WORKS_ON
 WHERE SSN=ESSN)
 CONTAINS
 (**SELECT** PNUMBER
 FROM PROJECT
 WHERE DNUM=5));

A segunda consulta aninhada retorna os números dos projetos controlados pelo departamento 5. Para cada tupla de empregado, a primeira consulta aninhada retorna os números dos projetos nos quais o empregado trabalha; se esses contêm todos os projetos controlados pelo depto 5, a tupla do empregado é selecionada e o seu nome retornado.

Esta consulta SQL corresponde a uma operação de divisão na álgebra relacional. Porém, CONTAINS não é parte do SQL padrão, e temos de utilizar EXISTS para simulá-lo.

FUNÇÕES DE AGRUPAMENTO

- Aplicar funções de agregação a subgrupos de tuplas em uma relação
- Exemplo: média de salário em cada departamento
- **SELECT DNO, COUNT(*), AVG(SALARY)**
FROM EMPLOYEE
GROUP BY DNO;

(a)

PNOME	MINICIAL	LNOME	SSN	• • •	SALARIO	SUPERSSN	DNO
John	B	Smith	123456789	• • •	30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453		25000	333445555	5
Alicia	J	Zelaya	999887777	• • •	25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	null	1

DNO	COUNT (*)	AVG (SALARIO)
5	4	33250
4	3	31000
1	1	55000

Resultado da Q24

Agrupamento das tuplas EMPREGADO por meio do valor de DNO

Agrupamento

```
SELECT DNO, COUNT(*), AVG(SALARY)  
FROM EMPLOYEE  
GROUP BY DNO;
```

Se um dno = NULL?!

Cria-se um grupo separado para todas as tuplas nas quais o atributo é NULL

Agrupamento

```
SELECT Pnumber, Pname, COUNT(*)  
FROM PROJECT, WORKS_ON  
WHERE Pnumber=Pno  
GROUP BY Pnumber, Pname;
```

Neste caso, GROUP BY é aplicado após a junção de projeto e trabalha_para

Agrupamento condicional

- Agrupa as tuplas, dos grupos resultantes, queremos os que satisfazem uma condição
- Agrupamento com a cláusula HAVING
- Somente os grupos que satisfazem a condição especificada em HAVING são retornados

```
SELECT PNUMBER, PNAME, COUNT(*)  
FROM PROJECT, WORKS_ON  
WHERE PNUMBER=PNO  
GROUP BY PNUMBER, PNAME  
HAVING COUNT(*) > 2;
```

WHERE → tuplas

HAVING → grupos de tuplas

(b)

PNAME	PNUMERO		ESSN	PNO	HORAS
ProdutoX	1		123456789	1	32,5
ProdutoX	1		453453453	1	20,0
ProdutoY	2		123456789	2	7,5
ProdutoY	2		453453453	2	20,0
ProdutoY	2		333445555	2	10,0
ProdutoZ	3		666884444	3	40,0
ProdutoZ	3		333445555	3	10,0
Automacao	10	. . .	333445555	10	10,0
Automacao	10		999887777	10	10,0
Automacao	10		987987987	10	35,0
Reorganizacao	20		333445555	20	10,0
Reorganizacao	20		987654321	20	15,0
Reorganizacao	20		888665555	20	null
NovosBeneficios	30		987987987	30	5,0
NovosBeneficios	30		987654321	30	20,0
NovosBeneficios	30		999887777	30	30,0

Esses grupos não são selecionados por HAVING, condição da Q26

Depois da aplicação da cláusula WHERE, mas antes da aplicação da cláusula HAVING

```
SELECT PNUMBER, PNAME, COUNT(*)  
FROM PROJECT, WORKS_ON  
WHERE PNUMBER=PNO  
GROUP BY PNUMBER, PNAME  
HAVING COUNT(*) > 2;
```

PNAME	PNUMERO		ESSN	PNO	HORAS	
ProdutoY	2		123456789	2	7,5	
ProdutoY	2		453453453	2	20,0	
ProdutoY	2		333445555	2	10,0	
Automacao	10	• • •	333445555	10	10,0	
Automacao	10		999887777	10	10,0	
Automacao	10		987987987	10	35,0	
Reorganizacao	20		333445555	20	10,0	
Reorganizacao	20		987654321	20	15,0	
Reorganizacao	20		888665555	20	null	
NovosBeneficios	30		987987987	30	5,0	
NovosBeneficios	30		987654321	30	20,0	
NovosBeneficios	30		999887777	30	30,0	

PNAME	COUNT (*)
ProdutoY	3
Automacao	3
Reorganizacao	3
NovosBeneficios	3

Resultado da Q26
(PNUMERO não apresentado)

Depois da aplicação da condição da cláusula HAVING

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT(*) > 2;
```

5. Revisão

```
SELECT <atributos e funções agregação>  
  FROM <lista de tabelas>  
[ WHERE <condições> ]  
[ GROUP BY <atributo(s) de agrupamento>  
  [ HAVING <condição no grupo> ]]  
[ ORDER BY <atributos> ];
```

Revisão

- **SELECT** <atributos e funções agregação>
lista os atributos ou funções a serem recuperados
- **FROM** <lista de tabelas>
todas as tabelas necessárias para a consulta, incluindo tabelas em junção
- **[WHERE <condições>]**
condições para selecionar as tuplas dessas tabelas, incluindo condições de junção se necessário

Revisão

- [GROUP BY <atributo(s) de agrupamento>]
especifica os atributos para formar os grupos
- [HAVING <condição no grupo>]
especifica uma condição que o grupo deve satisfazer para pertencer ao resultado
- [ORDER BY <atributos>];
especifica a ordem na qual o resultado deve aparecer

Considerações finais

- As consultas são avaliadas conceitualmente na seguinte ordem:
 1. FROM, identifica as tabelas/junções
 2. WHERE
 3. GROUP BY
 4. HAVING
 5. ORDER BY
- Se a consulta não tem group by, having e order by
 1. Para cada combinação de linhas (uma de cada relação especificada em FROM)
 2. Avalia a cláusula WHERE
 3. SE é true, adiciona os atributos de SELECT da combinação de linhas no resultado

Tudo junto reunido

Para cada departamento que tem mais de 5 empregados, retorne o número do departamento e o número de seus empregados que ganham mais de 40.000.

```
EMPREGADO (ssn, pnome, minicial, unome, ...,  
           salario, superssn, dno)  
  superssn REFERENCIA EMPREGADO  
  dno REFERENCIA DEPARTAMENTO  
DEPARTAMENTO (dnumero, dnome, gerssn,  
             gerdatainicio)  
  gerssn REFERENCIA EMPREGADO.ssn
```

Tudo junto reunido

Para cada departamento que tem mais de 5 empregados, retorne o número do departamento e o número de seus empregados que ganham mais de 40.000.

```
SELECT dnumero, COUNT(*)  
FROM DEPARTAMENTO, EMPREGADO  
WHERE dnumero=dno AND salario>40000  
      AND dno IN (SELECT dno  
                  FROM EMPREGADO  
                  GROUP BY dno  
                  HAVING COUNT (*) > 5)  
GROUP BY dnumero;
```

Exercícios

Pesquisador (codP, nome)

tabela com dados dos pesquisadores que têm produção científica

Artigo (codA, titulo, veiculo, ano)

tabela com os dados dos artigos publicados: título do artigo, veículo (nome do periódico ou do evento), ano de publicação

Referencia (codArt, codRef)

tabela que contém referências bibliográficas - cada linha contém o artigo que faz a referência e o artigo que é referenciado

Autoria (codA, codP, posicao)

tabela que liga um artigo com seus vários pesquisadores autores -- a coluna *posição* contém a ordem do autor dentro do artigo (1, 2, ...)

Pesquisador (codP, nome)

Artigo (codA, titulo, veiculo, ano)

Referencia (codArt, codRef)

codArt referencia Artigo, codRef referencia Artigo

Autoria (codA, codP, posicao)

codA referencia Artigo, codP referencia Pesquisador

1. Obter a quantidade de artigos publicados
2. Obter a quantidade de artigos publicados por ano
3. Obter os nomes dos pesquisadores que, em 2006, publicaram artigos em veículo intitulado “VLDB Journal”, constando como primeiro autor.
4. Para cada pesquisador, retornar seu nome e a quantidade de artigos publicados
5. Para cada artigo, retornar seu título e o número de autores
6. Obter os nomes dos pesquisadores que *não* foram primeiro autor de artigos.
7. Para cada pesquisador, retornar seu nome e o número de coautores (sem duplicatas) – PONTO EXTRA