

Linguagem SQL - DML

Base de Dados - 2017/18

Carlos Costa

SQL DML - Introdução

- DML - Data Manipulation Language
- Os comandos SQL DML permitem:
 - Inserir, eliminar e atualizar dados
 - Efetuar consultas:
 - Simples
 - Avançadas

SQL DML

INSERT, DELETE e UPDATE

3

Inserção - INSERT INTO

- Utilizado para inserir um novo tuplo numa relação.
 - Sintaxe 1: Não se indicam as colunas, tendo os valores inseridos de respeitar a ordem de criação dos atributos. Podemos utilizar os termos NULL ou DEFAULT:

```
INSERT INTO tablename VALUES (v1,v2,...,vn);
```

```
INSERT INTO EMPLOYEE VALUES
('Richard', 'K', 'Marini', '653298653', NULL, '98
Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);
```

- Sintaxe 2: Indicamos as colunas em que queremos inserir os dados. As restantes ficam com o seu valor nulo ou por defeito (caso tenha sido definido):

```
INSERT INTO tablename (A1,A4,A8,...,An) VALUES (v1,v4,v8,...,vn);
```

```
INSERT INTO EMPLOYEE (Dno, Fname, Lname, Ssn) VALUES
(4, 'Richard', 'Marini', '653298653');
```



Eliminação - DELETE

- Utilizado para remover um ou mais tuplos de uma relação.

```
DELETE FROM tablename WHERE match_condition;
```

-- remoção (potencial) de um tuplo:

```
DELETE FROM EMPLOYEE WHERE Ssn='123456789';
```

-- remoção (potencial) de n tuplos:

```
DELETE FROM EMPLOYEE WHERE Dno = 5;
```

-- ou

```
DELETE FROM EMPLOYEE WHERE Dno > 5 AND Dno < 8;
```

-- remoção de todos os tuplos da relação:

```
DELETE FROM EMPLOYEE;
```

Só afecta uma relação. No entanto, a ação pode propagar-se a outras relações devido às definições de integridade referencial (on delete cascade).



Actualização - UPDATE

- Utilizado para atualizar um ou mais tuplos de uma relação.

```
UPDATE tablename SET A1=v1,...,An=vn WHERE match_condition;
```

-- atualiza um tuplo:

```
UPDATE PROJECT  
SET Plocation = 'Bellaire', Dnum = 5  
WHERE Pnumber=10;
```

-- atualização (potencial) de n tuplos:

```
UPDATE EMPLOYEE  
SET Salary = Salary * 1.1  
WHERE Dno = 5;
```

Só afecta uma relação. No entanto, a ação pode propagar-se a outras relações devido às definições de integridade referencial (on update cascade). 6

SQL DML

Consultas Simples

7

Operações com Conjuntos

- A linguagem SQL é baseada em operações de conjuntos e de álgebra relacional.
- No entanto, existem particularidades:
 - modificações e extensões
- SQL define formas de lidar com tuplos duplicados
 - Especifica quantas cópias dos tuplos aparecem no resultado.
 - Existem comandos para eliminar duplicados
 - Versões Multiconjunto de operadores (AR)
 - i.e. as relações podem ser multiconjuntos

8

Projeção - SELECT FROM

- **SELECT FROM**
 - Permite selecionar um conjunto de atributos (colunas) de uma ou mais tabelas.

$\Pi_{\langle \text{attribute_list} \rangle} (R1)$

```
-- Forma Básica:  
SELECT <attribute_list> FROM <table_list>;
```

SELECT * FROM EMPLOYEE; -- Todas as colunas

SELECT Fname, Ssn FROM EMPLOYEE; -- Duas colunas

| EMPLOYEE | | | | | | | |
|----------|-------|---------|-----------|------------|--------------------------|-----|-----|
| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Dno |
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 5 |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 5 |
| Alicia | J | Zelaya | 999887777 | 1968-01-10 | 3321 Castle, Spring, TX | F | 4 |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 4 |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 5 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 1 |

| Fname | Ssn |
|----------|-----------|
| John | 123456789 |
| Franklin | 333445555 |
| Alicia | 999887777 |
| Jennifer | 987654321 |
| Ramesh | 666884444 |
| Joyce | 453453453 |
| Ahmad | 987987987 |
| James | 888665555 |

9

SELECT ALL vs DISTINCT

- Podemos selecionar todos os tuplos ou eliminar os duplicados.
 - Tendo em atenção que, ao selecionarmos só algumas colunas da tabela, o resultado pode não ser um conjunto (set) mas um multiconjunto.

-- Todos os tuplos (por defeito):
SELECT All <attribute_list> FROM <table_list>;

-- Eliminar tuplos repetidos:
SELECT DISTINCT <attribute_list> FROM <table_list>;

SELECT ALL Salary FROM EMPLOYEE;

SELECT DISTINCT Salary FROM EMPLOYEE;

| Salary |
|--------|
| 30000 |
| 40000 |
| 25000 |
| 43000 |
| 38000 |
| 25000 |
| 25000 |
| 55000 |

| Salary |
|--------|
| 30000 |
| 40000 |
| 25000 |
| 43000 |
| 38000 |
| 55000 |

10

DISTINCT não pode ser aplicado a cada atributo individualmente. Deve aparecer depois do SELECT e aplicar-se ao tuplo.

Seleção - WHERE

- WHERE permite selecionar um subconjunto de tuplos da(s) tabela(s) de acordo com uma expressão condicional.

$$\Pi_{\text{attribute_list}}(\sigma_{\text{condition}}(R1))$$

```
SELECT <attribute_list> FROM <table_list> WHERE <condition>;
```

```
SELECT Bdate, Address FROM EMPLOYEE
WHERE Fname='John' AND Minit='B' AND Lname='Smith';
```

A condição pode conter operadores de comparação ($=$, $<$, \leq , $>$, \geq , \neq) e ser composta usando AND, OR e NOT.

| EMPLOYEE | | | | | | | | |
|----------|-------|---------|-----------|------------|--------------------------|-----|-----|--|
| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Dno | |
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 5 | |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 5 | |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 4 | |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 4 | |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 5 | |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 5 | |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 4 | |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 1 | |

| Bdate | Address |
|------------|-------------------------|
| 1965-01-09 | 731Fondren, Houston, TX |

11

Renomeação - Relação, Atributo e Aritmética

- Podemos renomear:
 - relações e atributos;
 - resultado de uma operação aritmética.

-- Renomear

-- Renomear Tabela*

```
SELECT E.Fname, E.Ssn FROM EMPLOYEE AS E;  $\rho_{R2}(R1)$ 
```

ou

```
SELECT E.Fname AS Fn, E.Ssn AS Ssname FROM EMPLOYEE AS E;
```

-- Renomear Atributo

```
SELECT Dno AS DepNumber FROM EMPLOYEE;  $\rho_{B1, \dots, Bn}(R1)$   $\rho_{R2(B1, \dots, Bn)}(R1)$ 
```

-- Renomear Resultado de Operação Aritmética**

```
SELECT Salary * 0.35 AS SalaryTaxes FROM EMPLOYEE;
```

* ver mais à frente a importância de renomear tabelas em operações de junção.
** qual o resultado de não renomear? Depende de SGBD. SQL Server não dá nome à coluna!!!

Reunião, Intersecção e Diferença

- Requisitos:

- as duas relações têm de ter o mesmo número de atributos.
- o domínio de cada atributo deve ser compatível.

- Operadores SQL:

- UNION, INTERSECT e EXCEPT
- devem ser colocados entre duas queries.
- tuplos duplicados são eliminados.

$R1 \cup R2$

$R1 \cap R2$

$R1 - R2$

- Para manter os tuplos duplicados devemos utilizar as suas versões multiconjunto.
- UNION ALL, EXCEPT ALL* e INTERSECT ALL*

13

* Não disponível em SQL SERVER

UNION - Exemplo

- Quais os projetos (número) que têm um funcionário **ou** um gestor do departamento que controla o projeto com o último nome Smith?

```

SELECT FROM .....
UNION (ALL)
SELECT FROM .....
(SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND Lname='Smith' )
UNION
(SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Essn=Ssn AND Lname='Smith' );

```

14



Produto Cartesiano

- Podemos utilizar mais do que uma relação na instrução SELECT FROM.
- O resultado é o produto cartesiano dos dois conjuntos.

R1 X R2 X .. X RN

```
SELECT * FROM table1, table2, ..., tableN;
```

-- Exemplo de Produto Cartesiano
`SELECT * FROM EMPLOYEE, DEPARTMENT;`

-- Exemplo de Produto Cartesiano só com dois atributos
-- >> Pode ser visto com Prod. Cartesiano seguido de Projeção
`SELECT Ssn, Dname FROM EMPLOYEE, DEPARTMENT;`

15



Junção de Relações - WHERE

- O Produto Cartesiano tem pouco interesse prático...
- No entanto, a associação do operador WHERE permite a junção de relações.

```
SELECT <attribute_list> FROM <table_list> WHERE <join_condition>;
```

-- Exemplo de “*select-project-join query*”

```
SELECT Fname, Lname, Address
FROM   EMPLOYEE, DEPARTMENT
WHERE  Dname='Research' AND Dnumber=Dno;
```

ANSI SQL 89

Join Condition

EMPLOYEE

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| John | B | Smith | 123456789 | 1985-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 |
| Alicia | J | Zelaya | 989887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 28000 | 987654321 | 4 |
| Jennifer | S | Wallace | 987654321 | 1941-08-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 |
| Ramesh | K | Narayan | 666884444 | 1982-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 28000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

| Fname | Lname | Address |
|----------|---------|--------------------------|
| John | Smith | 731 Fondren, Houston, TX |
| Franklin | Wong | 638 Voss, Houston, TX |
| Ramesh | Narayan | 975 Fire Oak, Humble, TX |
| Joyce | English | 5631 Rice, Houston, TX |

DEPARTMENT

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|----------------|---------|-----------|----------------|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

Junção de 3 Relações - Exemplo

- Caso com três relações e duas *join conditions*:

```
/* Questão: Para cada projeto localizado em 'Stafford', queremos saber o seu número, o número do departamento que o controla e último nome, endereço e data de nascimento do gestor desse departamento. */
```

| <pre>SELECT Pnumber, Dnum, Lname, Address, Bdate FROM EMPLOYEE, DEPARTMENT, PROJECT WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND Plocation='Stafford';</pre> | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Pnumber</th> <th>Dnum</th> <th>Lname</th> <th>Address</th> <th>Bdate</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>4</td> <td>Wallace</td> <td>291Berry, Bellaire, TX</td> <td>1941-06-20</td> </tr> <tr> <td>30</td> <td>4</td> <td>Wallace</td> <td>291Berry, Bellaire, TX</td> <td>1941-06-20</td> </tr> </tbody> </table> | Pnumber | Dnum | Lname | Address | Bdate | 10 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 | 30 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 |
|---|--|---------|------------------------|------------|---------|-------|----|---|---------|------------------------|------------|----|---|---------|------------------------|------------|
| Pnumber | Dnum | Lname | Address | Bdate | | | | | | | | | | | | |
| 10 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 | | | | | | | | | | | | |
| 30 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 | | | | | | | | | | | | |

Join Condition 1 Join Condition 2

| EMPLOYEE | | | | | | | | | |
|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

| Pnumber | Dnum | Lname | Address | Bdate |
|---------|------|---------|------------------------|------------|
| 10 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 |
| 30 | 4 | Wallace | 291Berry, Bellaire, TX | 1941-06-20 |

| PROJECT | | | |
|-----------------|---------|-----------|------|
| Pname | Pnumber | Plocation | Dnum |
| ProductX | 1 | Bellaire | 5 |
| ProductY | 2 | Sugarland | 5 |
| ProductZ | 3 | Houston | 5 |
| Computerization | 10 | Stafford | 4 |
| Reorganization | 20 | Houston | 1 |
| Newbenefits | 30 | Stafford | 4 |

| DEPARTMENT | | | |
|----------------|---------|-----------|----------------|
| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

Junção - Ambiguidade de Nomes de Atributos

- Quando existem nomes de atributos iguais em distintas relações da junção, podemos utilizar o *full qualified name (fqn)*:

relation_name.attribute

```
/* Exemplo: Vamos pegar num dos exemplos anteriores e imaginar que o atributo Dno de EMPLOYEE se chamava Dnumber... */
```

| | |
|--|--|
| <pre>SELECT Fname, Lname, Address FROM EMPLOYEE, DEPARTMENT WHERE Dname='Research' AND EMPLOYEE.Dnumber=DEPARTMENT.Dnumber;</pre> | |
|--|--|

Podemos também utilizar o fqn em situações em que não há ambiguidade de nomes.

18



Junção - Ambiguidade + Renomeação

- Há situações em que ambiguidade de nomes de atributos resulta de termos uma relação recursiva.
- Nesta situação temos de renomeação as relações (*alias*).

```
/* Exemplo: Para cada Funcionário, pretendemos obter o seu
primeiro e último nome, assim como do seu supervisor. */
```

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM   EMPLOYEE AS E, EMPLOYEE AS S
WHERE  E.Super_ssn=S.Ssn;
```

Muitas vezes a renomeação envolvendo várias relações ajuda a melhorar a legibilidade da instrução.

19



Queries - Comparaçāo de Strings

- Operador LIKE permite comparar *Strings*
- Podemos utilizar wildcards.
 - % - significa zero ou mais caracteres.
 - _ - significa um qualquer carácter.

Exemplos:

```
/* Obter o primeiro e último nome dos funcionários cujo endereço contém a
substring 'Houston,TX'. */
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  Address LIKE '%Houston,TX%';
```

```
/* Obter o primeiro e último nome dos funcionários nascidos nos anos 50 */
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  Bdate LIKE '_ _ 5 _ _ _ _ _';
```

Queries - Comparações de Strings

- Podemos pesquisar os próprios wildcards na string.
 - Para isso utilizamos um carácter especial a preceder o wildcard
 - Devemos definir esse carácter com a instrução ESCAPE

LIKE ... ESCAPE

```
/* Nome dos funcionários cujo endereço contém a substring 'Houston%,TX'. */
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston@%,TX%' ESCAPE '@';
```

- Alguns SGBD permitem utilizar outros Wildcards.

| Description | SQL Wildcard | MS-DOS Wildcard | Example |
|---|--------------|-----------------|--|
| Any number (zero or more) of arbitrary characters | % | * | 'Able' LIKE 'A%' |
| One arbitrary character | _ | ? | 'Able' LIKE 'Ab_' |
| One of the enclosed characters | [] | n/a | 'a' LIKE '[a-g]' 'a' LIKE '[abcdefghijklm]' |
| Match not in range of characters | [^] | n/a | 'a' LIKE '[^ w-z]' 'a' LIKE '[^ wxyz] ' |

SQL SERVER

21

Queries - Operadores Aritméticos e BETWEEN

- Operações Aritméticas:
 - Operadores: adição (+), subtração (-), multiplicação (*), divisão (/)
 - Operandos: valores numéricos ou atributos com domínio numérico.
- BETWEEN
 - Verificar se um atributo está entre uma gama de valores.

Exemplos:

```
/* Obter o salário, com um aumento de 10%, de todos os trabalhadores do projeto GalaxyS. */
```

```
SELECT E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND P.Pname='GalaxyS';

/* Funcionários do departamento nº 5 com salário entre 3k e 4k */
SELECT * FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Queries - Ordenação de Resultados

- Podemos ordenar os resultados segundo uma ou mais colunas.
- Sintaxe: **ORDER BY A₁, ..., A_k**
 - A₁, ..., A_k - atributos a ordenar.
 - 1,2,...,k - também podemos usar o número da coluna
- Podemos definir se é ascendente (ASC) ou descendente (DESC).
 - Por omissão as colunas são ordenadas ascendentemente.

Exemplo:

```
/* Lista de funcionários e projetos em que trabalham, ordenado por
departamento e, dentro deste, pelo último nome (descendente) e depois o
primeiro */

SELECT      D.Dname, E.Lname, E.Fname, P.Pname
FROM        DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE       D.Dnumber= E.Dno AND E.Ssn= W.Essn AND W.Pno= P.Pnumber
ORDER BY    D.Dname, E.Lname DESC, E.Fname;
/* ... ORDER BY 1, 2 DESC, 3; */
```

SQL DML

Consultas Avançadas

Tratamento dos NULL

- NULL
 - significa um valor desconhecido ou que não existe.
- SQL tem várias regras para lidar com os valores null.
- O resultado de uma expressão aritmética com *null* é *null*: *5+null* é *null*
- Temos possibilidade de verificar se determinado atributo é nulo: IS NULL
- Por norma, as funções de agregação ignoram o null.

25

NULL - Lógica de 3 Valores

- Quando se faz uma comparação lógica temos duas possibilidades de retorno: TRUE, FALSE
- SQL - comparação com NULL retorna UNKNOWN.
 - *12 < null*, *null <> null*, *null = null*, etc.
- Assim temos uma lógica de 3 valores em SQL:

| AND | TRUE | FALSE | UNKNOWN |
|---------|---------|---------|---------|
| TRUE | TRUE | FALSE | UNKNOWN |
| FALSE | FALSE | FALSE | FALSE |
| UNKNOWN | UNKNOWN | FALSE | UNKNOWN |
| OR | TRUE | FALSE | UNKNOWN |
| TRUE | TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE | UNKNOWN |
| UNKNOWN | TRUE | UNKNOWN | UNKNOWN |
| NOT | | | |
| TRUE | FALSE | | |
| FALSE | TRUE | | |
| UNKNOWN | UNKNOWN | | |

26

deti

NULL Lógica 3 Valores - Exemplo

| EMPLOYEE | | | | | | | | | | |
|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|--|
| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno | |
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 | |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 | |
| Alicia | J | Zeloya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 | |
| Jennifer | S | Wallace | 987654321 | 1941-08-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 | |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 | |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 | |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 | |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | NULL | NULL | 1 | |

Exemplos:

```
/* Exemplo 1 */
SELECT Fname, Salary
FROM EMPLOYEE
WHERE Salary > 40000;
```

NULL > 40000
UNKNOWN

| Fname | Salary |
|----------|--------|
| Jennifer | 43000 |


```
/* Exemplo 2 */
SELECT Fname, Salary
FROM EMPLOYEE
WHERE Salary > 40000 OR Fname='James';
```

UNKNOWN OR TRUE

| Fname | Salary |
|----------|--------|
| Jennifer | 43000 |
| James | NULL |

27

deti

IS (NOT) NULL - Exemplo

- **IS NULL**: selecionar tuplos com determinado atributo a NULL;
- **IS NOT NULL**: selecionar tuplos com determinado atributo diferente de NULL;

Exemplos:

```
-- IS NOT NULL
SELECT * FROM EMPLOYEE
WHERE Super_ssn IS NOT NULL;
```



```
-- IS NULL
SELECT * FROM EMPLOYEE
WHERE Super_ssn IS NULL;
```

| EMPLOYEE | | | | | | | | | | |
|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|--|
| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno | |
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 | |
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 | |
| Alicia | J | Zeloya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 | |
| Jennifer | S | Wallace | 987654321 | 1941-08-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 | |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 | |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 | |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 | |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 | |

28

Junções - JOIN ON

- WHERE
 - Já vimos que o produto cartesiano associado ao operador “where” permite juntar várias relações. (ANSI SQL 89)

- ANSI SQL 92: JOIN ON

- Permite especificar simultaneamente as tabelas a juntar e a condição de junção.

R \bowtie_C S

```
SELECT ... FROM(.. [INNER] JOIN .. ON ..) ...;
-- [INNER] é opcional
```

-- exemplo de Equi-join:

```
SELECT Fname, Lname, Address
FROM (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE Dname='Research';
```

.9

NATURAL JOIN

- Junção Natural - os atributos de junção têm todos o mesmo nome nas duas relações.
- Os atributos repetidos são removidos.
- Podemos renomear os atributos de uma relação para permitir a junção natural.

R \bowtie S

```
SELECT ... FROM(.. NATURAL JOIN ..) WHERE <condition>;
```

-- exemplo de Natural Join com renomeação:

```
SELECT Fname, Lname, Address
FROM (EMPLOYEE NATURAL JOIN
      (DEPARTMENT AS DEPT (Dname, Dno, Mssn, Msdate)))
WHERE Dname='Research';
```

Não disponível em
SQL Server!

30

 deti

OUTER JOIN

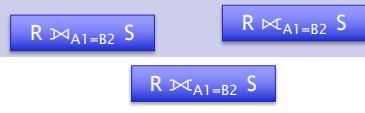
- As junções externas podem ser à esquerda, à direita ou totais (LEFT, RIGHT, FULL).

```
SELECT .. FROM (... LEFT|RIGHT|FULL [OUTER] JOIN ...) ...;
```

```
/* exemplo de Outer Join com renomeação das relações e atributos */

SELECT E.Lname AS Employee_name, S.Lname AS Supervisor_name
FROM   (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S
        ON E.Super_ssn=S.Ssn);

-- RIGHT OUTER JOIN
-- FULL OUTER JOIN
```



Nota: Em Oracle utiliza-se o operador (+) à frente do atributo na cláusula WHERE.

31

 deti

JOIN - Encadeamento

- Podemos ter várias operações JOIN encadeadas envolvendo 3..N relações.
 - uma das relações da junção resulta de outra operação de junção.

```
SELECT .. FROM (... JOIN .. JOIN .. JOIN ...) ...;
```

```
/* Exemplo do slide 17: Para cada projeto localizado em
 'Stafford', queremos saber o seu número, o número do
 departamento que o controla e último nome, endereço e data de
 nascimento do gestor desse departamento. */
-- Nota: Neste caso as join conditions estão à frente do ON

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM   ((PROJECT JOIN DEPARTMENT ON Dnum=Dnumber)
        JOIN EMPLOYEE ON Mgr_ssn=Ssn)
WHERE  Plocation='Stafford';
```

32

Agregações

- Funções de agregação introduzidas em álgebra relacional.
- Funções de Agregação
 - Exemplos*: COUNT, SUM, MAX, MIN, AVG
 - Em geral, não são utilizados os tuplos com valor NULL no atributo na função.
- Efetuar agregação por atributos
 - GROUP BY <grouping attributes>
- Efetuar seleção sobre dados agrupados
 - HAVING <condition>

33

* Existem outras funções de agregação específicas do SGBD

Funções de Agregação - Exemplo

Exemplos... sem agrupamento de atributo(s)

```
/* Exemplo 1: relativamente aos salários dos funcionários, obter o valor total, o máximo, o mínimo e o valor médio */
SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM EMPLOYEE;

/* Exemplo 2: Nº de funcionários do departamento 'Research' */
SELECT COUNT (*)
FROM EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER
WHERE DNAME='Research';

/* Exemplo 3: Nº de vencimentos distintos */
SELECT COUNT (DISTINCT Salary)
FROM EMPLOYEE;
```

Nota1: O operador COUNT(A1) conta o número de valores não NULL do atributo A1.
O operador COUNT(*) conta o número de linhas.

Nota2: Min, Max, Count(...) e Count(*) podem ser utilizadas com qualquer tipo de dados. SUM e AVG só podem ser aplicadas a campos numéricos.

34

Agregação (GROUP BY) - Exemplo

Exemplos... agregação de atributo(s)

```
/* Exemplo 1: para cada departamento, obter o seu número, o
número de funcionários e a sua média salarial */
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
GROUP BY Dno;
```

Os “grouping attributes” devem aparecer na cláusula SELECT
Exemplo: Dno

| Fname | Minit | Lname | San | ... | Salary | Super_ssn | 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 (Salary) |
|-----|-----------|--------------|
| 5 | 4 | 33250 |
| 4 | 3 | 31000 |
| 1 | 1 | 55000 |

Result of Q24

```
/* Exemplo 2: agregação com junção de duas relações */
SELECT Pnumber, Pname, COUNT(*)
FROM PROJECT JOIN WORKS_ON ON Pnumber=Pno
GROUP BY Pnumber, Pname;
```

Nota: Se existirem valores NULL nos “grouping attribute”, então é criado um grupo com todos os tuplos contendo NULL nesses atributos.

35

Agregação (GROUP BY.. HAVING) - Exemplo

Exemplo... agregação de atributo(s) com seleção

```
/* Exemplo 1: Para cada projeto, com mais de dois funcionários,
obter o seu nome e nº de funcionários que trabalham no projeto
*/
SELECT Pname, COUNT(*)
FROM PROJECT join WORKS_ON
ON Pnumber=Pno
GROUP BY Pname
HAVING COUNT(*) > 2;
```

| Pname | Count (*) |
|-----------------|-----------|
| ProductY | 3 |
| Computerization | 3 |
| Reorganization | 3 |
| Newbenefits | 3 |

Junção

| Pname | Pnumber | ... | Essn | Pno | Hours |
|-----------------|---------|-----|-----------|-----|-------|
| ProductX | 1 | ... | 123456789 | 1 | 32.5 |
| ProductX | 1 | ... | 453453453 | 1 | 20.0 |
| ProductY | 2 | ... | 123456789 | 2 | 7.5 |
| ProductY | 2 | ... | 453453453 | 2 | 20.0 |
| ProductY | 2 | ... | 333445555 | 2 | 10.0 |
| ProductZ | 3 | ... | 666884444 | 3 | 40.0 |
| ProductZ | 3 | ... | 333445555 | 3 | 10.0 |
| Computerization | 10 | ... | 333445555 | 10 | 10.0 |
| Computerization | 10 | ... | 999887777 | 10 | 10.0 |
| Computerization | 10 | ... | 987987987 | 10 | 35.0 |
| Reorganization | 20 | ... | 333445555 | 20 | 10.0 |
| Reorganization | 20 | ... | 987654321 | 20 | 15.0 |
| Reorganization | 20 | ... | 888665555 | 20 | NULL |
| Newbenefits | 30 | ... | 987987987 | 30 | 5.0 |
| Newbenefits | 30 | ... | 987654321 | 30 | 20.0 |
| Newbenefits | 30 | ... | 999887777 | 30 | 30.0 |

Nota1: A condição da cláusula WHERE é aplicada antes da criação dos grupos. A condição do HAVING é executada depois da criação dos grupos.

Nota2: Na cláusula HAVING só podemos ter atributos que aparecem em GROUP BY ou funções de agregação.

36

Agregação - Resumo

SQL

```
SELECT      A1,..,An, FAgri1,..Fagrh
FROM        R1,R2,..,Rm
WHERE       <condition_W>
GROUP BY   A1,..,An
HAVING     <condition_H>;
```

Expressão equivalente em álgebra relacional

$$\Pi_{A1,..,An, FAgri1,..Fagrh} (\sigma_{<\text{condition_H}>} (\pi_{A1,..,An} \delta_{FAgri1,..,Fagrh} (\sigma_{<\text{condition_W}>} (R1 \times .. \times Rm))))$$

Importante para se perceber
a ordem das operações

37

SubConsultas (SubQueries)

- É possível usar o resultado de uma query, i.e. uma relação, noutra query.
 - Nested Queries
- Subconsultas podem aparecer na cláusula:
 - FROM - entendidas como cálculo de relações auxiliares.
 - WHERE - efetuar testes de pertença a conjuntos, comparações entre conjuntos, calcular a cardinalidade de conjuntos, etc.

38

Cláusula FROM - Subquery como Tabela

- Podemos utilizar o resultado de uma subquery como uma tabela na cláusula FROM, dando-lhe um nome (alias).

Exemplo... agregação de atributo(s) com seleção

```
/* Exemplo 1: Obter uma lista de funcionários com mais de dois dependentes */
SELECT      Fname, Minit, Lname, Ssn
FROM        Employee JOIN ( SELECT      Essn
                           FROM        DEPENDENT
                           GROUP BY    Essn
                           HAVING      count(Essn)>2) AS Dep
                           ON Ssn=Dep.Essn;
```

39

Operador IN - Pertença a Conjunto

- WHERE A1,...,An IN (SELECT B1,...,Bn FROM ...)
 - Permite selecionar os tuplos em que os atributos indicados (A1,...,An) existem na subconsulta.
 - B1,...,Bn são os atributos retornados pela subconsulta
- A1,...,An e B1,...,Bn
 - têm de ter o mesmo número atributos e domínios compatíveis.
- NOT IN
 - permite obter o resultado inverso.

40

Operador IN - Exemplo

Exemplos...

```

/* Exemplo 1: Obter o nome de todos os funcionários que não têm
dependentes */
SELECT      Fname, Minit, Lname
FROM        EMPLOYEE
WHERE       Ssn NOT IN (SELECT Essn FROM DEPENDENT);

/* Exemplo 2: Obter o Ssn de todos os funcionários que trabalham
no mesmo projeto, e o mesmo número de horas, que o funcionário
com o Ssn = '123456789'*/
SELECT      DISTINCT Essn
FROM        WORKS_ON
WHERE       (Pno, Hours) IN ( SELECT   Pno, Hours
                             FROM     WORKS_ON
                             WHERE    Essn='123456789');

/* Exemplo 3: Obter o Ssn de todos os funcionários que trabalham
no projeto nº 1, 2 ou 3 */
SELECT      DISTINCT Essn
FROM        WORKS_ON
WHERE       Pno IN (1, 2, 3);

```

SQL Server não suporta múltiplas colunas!

Comparação de Conjuntos

- Existem **operadores** que pode ser utilizados para **comparar** um **valor simples** (tipicamente um atributo) **com** um **set** ou **multiset** (tipicamente uma subquery).
- ANY (= CASE)**
 - Permite selecionar os resultados cujos atributos indicados sejam iguais (=), maiores (>), menores(<) ou diferentes (<>) do que pelo menos um tuplo da subquery.
 - =ANY é o mesmo que IN
- ALL**
 - Também pode ser combinada com os operadores iguais (=), maiores (>), menores(<) ou diferentes (<>).

42

 deti

ANY e ALL - Exemplos

Exemplos...

```

/* Exemplo 1: Obter o nome dos funcionários cujo salário é
maior do que o salário de todos os trabalhadores do departamento
5 */
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       Salary > ALL ( SELECT  Salary
                           FROM    EMPLOYEE
                           WHERE   Dno=5);

/* Exemplo 2: Obter o nome dos funcionários cujo salário é
maior do que o salário de algum trabalhador do departamento 5 */
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       Salary > ANY ( SELECT  Salary
                           FROM    EMPLOYEE
                           WHERE   Dno=5);

```

13

 deti

Teste de Relações Vazias - EXISTS

- O operador EXISTS retorna
 - TRUE, se subconsulta não é vazia.
 - FALSE, se subconsulta é vazia.
- Existe a possibilidade de utilizar o NOT EXISTS

SQL - (NOT) EXISTS

```

/* Exemplo 1: Nomes dos funcionários que não têm dependentes */
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       NOT EXISTS ( SELECT  *
                           FROM    DEPENDENT
                           WHERE   Ssn=Essn );

```

44

Existem Tuplos Duplicados? - UNIQUE

- Unique permite verificar se o resultado de uma subconsulta possui tuplos duplicados.
- Permite verificar se determinado resultado (relação) é um conjunto ou um multiconjunto.

SQL – (NOT) EXISTS

```
/* Exemplo 1: Nomes dos funcionários que gerem um departamento.
(supondo que o mesmo funcionário pode gerir mais do que um
departamento...) */
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       UNIQUE      ( SELECT    Mgr_ssn
                           FROM      DEPARTMENT
                           WHERE     Ssn=Mgr_ssn );
```

Não disponível em
SQL Server!

45

SubConsultas Não Correlacionadas

- A subquery (query interior) não depende de dados lhe são fornecidos pela query exterior.
 - Nestes casos, a query interior é executada uma única vez e o resultado é utilizado no SELECT exterior.

SubConsulta Correlacionada

```
/* Exemplo 1: Nome dos funcionário que são gestores de
departamento */

SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       Ssn IN      (           SELECT    Mgr_ssn
                           FROM      DEPARTMENT
                           WHERE     Mgr_ssn IS NOT NULL);
```



46

SubConsultas Correlacionadas

- A subquery (query interior) depende de dados lhe são fornecidos pela query exterior.
 - Nestes casos, a query interior é executada uma vez para cada resultado do SELECT exterior.

SubConsulta Correlacionada

```
/* Exemplo 1: Nome dos funcionários que tem um dependente com o
primeiro nome e sexo igual ao próprio funcionário */
```

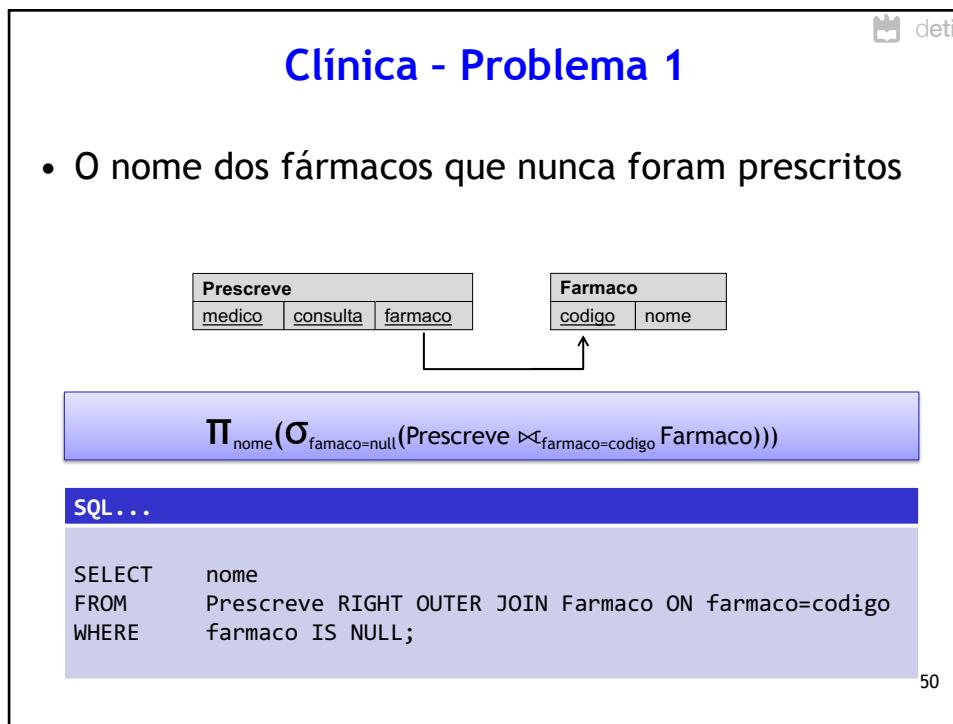
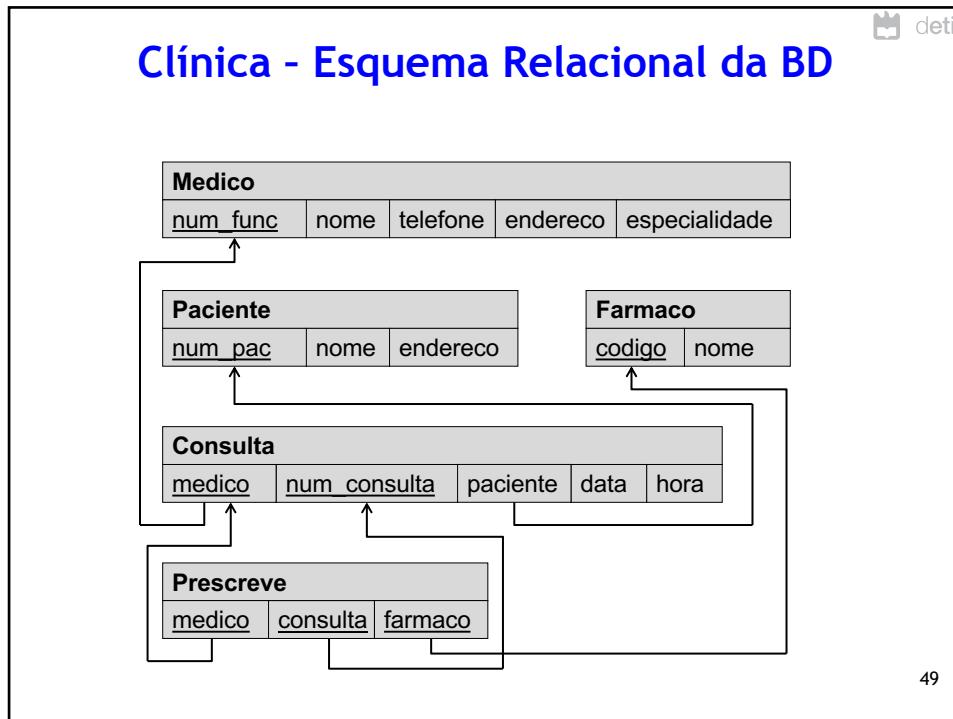
```
SELECT      E.Fname, E.Lname
FROM        EMPLOYEE AS E
WHERE       E.Ssn IN (      SELECT      Essn
                           FROM        DEPENDENT AS D
                           WHERE       E.Fname=D.Dependent_name
                                      AND E.Sex=D.Sex );
```

47

SQL DML - Caso de Estudo

Clínica
(Conversão das Queries AR para SQL)

48



Clínica - Problema 2

- O número de fármacos prescritos em cada consulta

| Prescreve | | | Farmaco | |
|-----------|----------|---------|---------|------|
| medico | consulta | farmaco | codigo | nome |



$\Pi_{\text{medico, consulta, num_farm}=\text{count}(\text{farmaco})} (\text{Prescreve})$

SQL...

```
SELECT      medico, consulta, count(farmaco) AS num_farm
FROM        Prescreve
GROUP BY    medico, consulta;
```

51

Clínica - Problema 3

- Para cada médico, a quantidade média de fármacos receitados por consulta

| Prescreve | | | Farmaco | |
|-----------|----------|---------|---------|------|
| medico | consulta | farmaco | codigo | nome |



$\Pi_{\text{medico, avg_farmaco}=\text{avg}(\text{num_farm})} (\Pi_{\text{medico, consulta, num_farm}=\text{count}(\text{farmaco})} (\text{Prescreve}))$

SQL...

```
SELECT      medico, avg(num_farm) AS avg_farmaco
FROM        (SELECT      medico, consulta, count(farmaco) AS num_farm
            FROM        Prescreve
            GROUP BY    medico, consulta) AS T
GROUP BY    medico;
```

52

Clínica - Problema 4

- O nome de todos os fármacos prescritos, incluindo a quantidade, para o paciente número 35312161

```

temp ← πmedico, num_consulta(σpaciente=35312161 (Consulta))
temp2 ← πfarmaco, quantidade=count(farmaco)(temp ⋈medico=medico AND num_consulta=consulta Prescreve)
πnome, quantidade(temp2 ⋈farmaco=codigo Farmaco)

```

SQL...

```

SELECT nome, quantidade
FROM Farmaco JOIN (SELECT farmaco, count(farmaco) AS quantidade
                     FROM Prescreve AS P
                     JOIN (SELECT medico, num_consulta
                           FROM Consulta
                           WHERE paciente=35312161) AS T
                           ON (P.medico=T.medico AND num_consulta=consulta) AS T2
                     GROUP BY farmaco)
                     ON farmaco=codigo;

```

Clínica - Problema 5

- O nome dos fármacos que já foram prescritos por todos os médicos da clínica

```

temp ← ρcodigo, num_func(πfarmaco, medico (Prescreve)) ÷ πnum_func(Medico)

```

SQL... Uma Implementação Alternativa da Query:

```

SELECT      farmaco, count(DISTINCT medico) as num_medicos
FROM        Prescreve
GROUP BY    farmaco
HAVING      count(DISTINCT medico)=(SELECT count(*) from Medico);

```

A Seguir?

| Data Operations – Relational Algebra | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-------|-------|---|---|---|---|---|---|---|-------|---|---|---|---|--|-------|-------|---|---|---|---|---|---|---|---|
| <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>r</th><th>A B</th></tr> <tr><td>α</td><td>1</td></tr> <tr><td>α</td><td>2</td></tr> <tr><td>β</td><td>1</td></tr> </table> ↔ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>s</th><th>A B</th></tr> <tr><td>α</td><td>2</td></tr> <tr><td>β</td><td>3</td></tr> </table> | r | A B | α | 1 | α | 2 | β | 1 | s | A B | α | 2 | β | 3 | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>r ∪ s</th><th>A B</th></tr> <tr><td>α</td><td>1</td></tr> <tr><td>α</td><td>2</td></tr> <tr><td>β</td><td>1</td></tr> <tr><td>β</td><td>3</td></tr> </table> | r ∪ s | A B | α | 1 | α | 2 | β | 1 | β | 3 |
| r | A B | | | | | | | | | | | | | | | | | | | | | | | | |
| α | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| α | 2 | | | | | | | | | | | | | | | | | | | | | | | | |
| β | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| s | A B | | | | | | | | | | | | | | | | | | | | | | | | |
| α | 2 | | | | | | | | | | | | | | | | | | | | | | | | |
| β | 3 | | | | | | | | | | | | | | | | | | | | | | | | |
| r ∪ s | A B | | | | | | | | | | | | | | | | | | | | | | | | |
| α | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| α | 2 | | | | | | | | | | | | | | | | | | | | | | | | |
| β | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| β | 3 | | | | | | | | | | | | | | | | | | | | | | | | |
| $\Pi_{\text{nome}}(\sigma_{\text{farmaco}=\text{null}}(\text{Prescreve} \bowtie_{\text{farmaco}=\text{codigo}} \text{Farmaco}))$ | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|--|
| SQL – Data Manipulation | | |
| SQL query: <pre>SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_ON WHERE Pnumber=Pno GROUP BY Pnumber, Pname;</pre> | SQL query: <pre>INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno) VALUES ('Robert', 'Hatcher', '980760540', 2);</pre> | |

| | | |
|---|--|--|
| SQL – Describe Database Schema | | |
| CREATE TABLE DEPARTMENT <pre>(Dname VARCHAR(15) NOT NULL, Dnumber INT NOT NULL, Mgr_ssn CHAR(9) NOT NULL, Mgr_start_date DATE,</pre> <p>PRIMARY KEY (Dname), UNIQUE (Dname), FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn);</p> | SQL View: <pre>CREATE VIEW EMPLOYEE_DEPS AS SELECT Fname, Lname, Ssn, Dno FROM EMPLOYEE WHERE Dno=5 WITH CHECK OPTION;</pre> | |

Resumo

- SQL DML
- Inserir, eliminar e atualizar dados
- Efectuar pesquisas:
 - Simples
 - Avançadas
- Caso de Estudo

56