

Linguagem SQL - DML

Base de Dados - 2018/19

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;`

*Estava a alterar o
tuplo do projeto 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).

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 **DISTINCT**
 - Versões Multiconjunto de operadores (AR)
 - i.e. as relações podem ser multiconjuntos

Em (AR) termos
a condição de conjunto

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;
```

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

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Salary
30000
40000
25000
43000
38000
55000

10

Seleção - WHERE

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

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

```
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.

Not equal!

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	838 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.

Vamos poder ter Querys dentro de Querys, isto é útil...

tbl1 para para ...

Em ambos posso ter sub-queries

Sub - Query...

*select * from Employee join (Select...) as ...*

-- Renomear

-- Renomear Tabela*

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

ou

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

-- Renomear Atributo

```
SELECT Dno AS DepNumber FROM EMPLOYEE;
```

-- 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!!!

Nova tabela (resultado)

Pode vir a ambiguidade (Se não editemos nome)

Pode vir a ambiguidade (Se não editemos nome)

$\rho_{R2(R1)}$

$\rho_{B1,\dots,Bn}(R1)$

 deti

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

* Não disponível em SQL SERVER

13

 deti

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' );

```

Automaticamente retira os duplicados

14

deti

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

15

deti

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"

seleção... não faz sentido

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

Não utiliza!

ANSI SQL 89

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

Join Condition

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	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	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	28000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

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. */

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

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

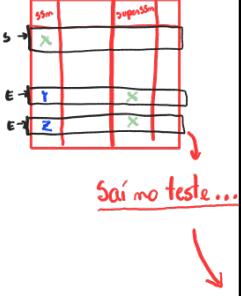
/* Exemplo: Vamos pegar num dos exemplos anteriores e imaginar que o atributo Dno de EMPLOYEE se chamava Dnumber... */

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

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;
```

A ordem em que o SQL executa pode ser diferente

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.

Muito interessante para busca de pesquisas!

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%';
      ↑
      No meio...
```

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

10

Queries - Comparações de Strings

deti

- 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 '@';
```

Nós é que definimos a seq. de Escape

literal

- 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

deti

- 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; → Orden!
/* ... ORDER BY 1, 2 DESC, 3; */
```

SQL DML

Consultas Avançadas

Tratamento dos NULL

- NULL *→ Temos de ter alguns cuidados...*
 - 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

Com booleano fica
UNKNOWN
!

NULL - Lógica de 3 Valores

- Quando se faz uma comparação lógica temos duas possibilidades de retorno: TRUE, FALSE
- SQL - comparação com NULL retorna UNKNOWN.
 - $12 < \text{null}$, $\text{null} <> \text{null}$, $\text{null} = \text{null}$, 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

Pergunta de Teste
Lógica de três valores
TRUE, FALSE, UNKNOWN

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	1941-06-20	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	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	1941-06-20	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	25000	987654321	4	

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
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 utilizar sempre a partir de agora...
 - Permite especificar simultaneamente as tabelas a juntar e a condição de junção.

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

-- exemplo de Equi-join:
`SELECT Fname, Lname, Address
 FROM (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
 WHERE Dname='Research';`

R \bowtie S

Vai ter pelo menos 1 coluna repetida!

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

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);  
  "Não preciso!"
```

-- RIGHT OUTER JOIN
-- FULL OUTER JOIN

R $\bowtie_{A1=B2}$ S

R $\bowtie_{A1=B2}$ S

R $\bowtie_{A1=B2}$ S

31

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

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';
```

12

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> *← Cláusula de seleção depois de agregação !!!*

* Existem outras funções de agregação específicas do SGBD

33

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 (*) Contabiliza os NULL
FROM EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER
WHERE DNAME='Research';

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

salary
10K
20K
null
null
5K
10K

COUNT(*) $\Rightarrow 6$ *com NULL*
 COUNT(Salary) $\Rightarrow 4$ *Sem NULL*
 COUNT(DISTINCT salary) $\Rightarrow 3$ *Diferentes*
sem NULL

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

Sai SEMPRE !!!

① *Tuplo (Dno, sex, avg(salary))* ↓
 • **SELECT Dno, Sex, COUNT(*), AVG(Salary)**
 Não podemos!
 • Não está na Agregação

② **GROUP BY Dno, Sex**
Tuplo (Dno, Sex, avg(salary))
 ✓ Não é muito útil
 mas é possível!

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)③
FROM EMPLOYEE④
GROUP BY Dno;
```

Tem de ser igual!!!

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

Fname	Minit	Lname	Sal	... Salary	Super_ss	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

No site para testar do Professor funciona, mas está errado!!

SELECT e.tname, COUNT(*)
FROM A GROUP BY e.tname, age

! → **contendo...** → **Valido**

Sai no teste

SEL.T Pname, COUNT(*) AS mp
FROM PROJECT JOIN WORKS_ON
ON Pnumber = Pno
GROUPING BY Pname
HAVING mp > 2

→ **Posso encapsular em outra query ...**

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;
```

Seleção por dados agregados...

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

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

Junção

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

 deti

SQL

```

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

```

A mais!
 Pode agrupar por mais colunas que projetos!
 Sai SEMPRE!

A última...

Como o nome só é atribuído no final, isto dá erro!

Expressão equivalente em álgebra relacional

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

Importante para se perceber a ordem das operações

37



SubConsultas (SubQueries)

 deti

- 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;
    
```

Sub-Query

Fonte de dados virtual

Precisa de nome ...

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);

```

Com a diferença ...
que não tem dependentes...

```

/* 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');

```

SQL Server não suporta múltiplas colunas!

```

/* 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);

```

conjunto explícito

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);

Todos maiores que os salários dos que
    os do Dep. 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 );

```

fSem dependentes...

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

Importante



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.

• Não correlacionada:
→ não à dependência
com a Query 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);
```

Não tem dependência!

Eficiente!

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 );
```

*Sub-Query Correlacionada
(dependência com a Query Exterior)*

47

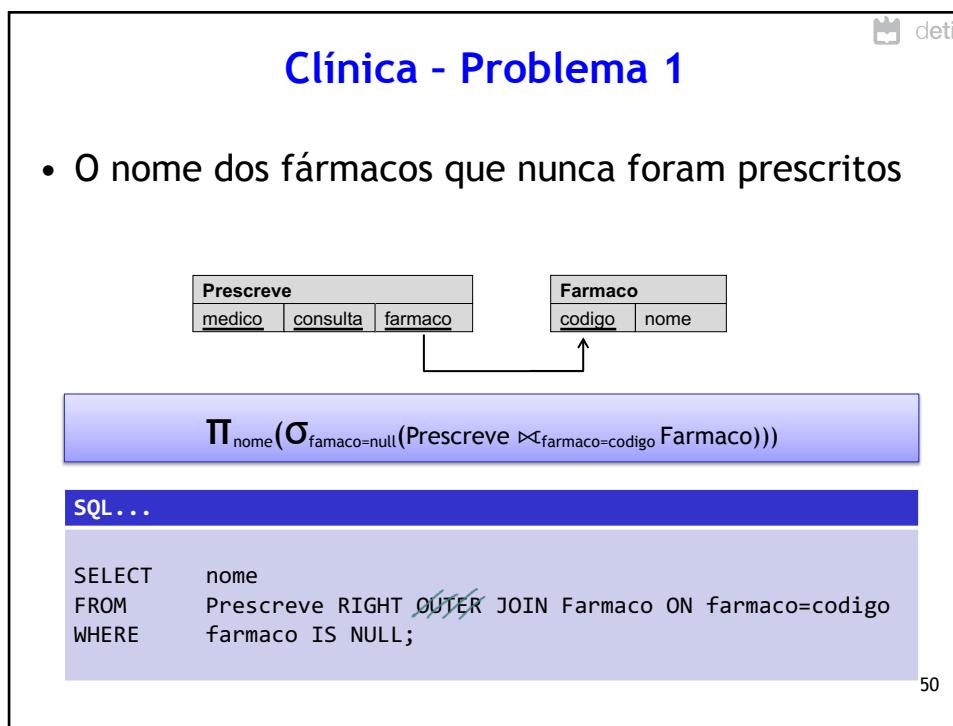
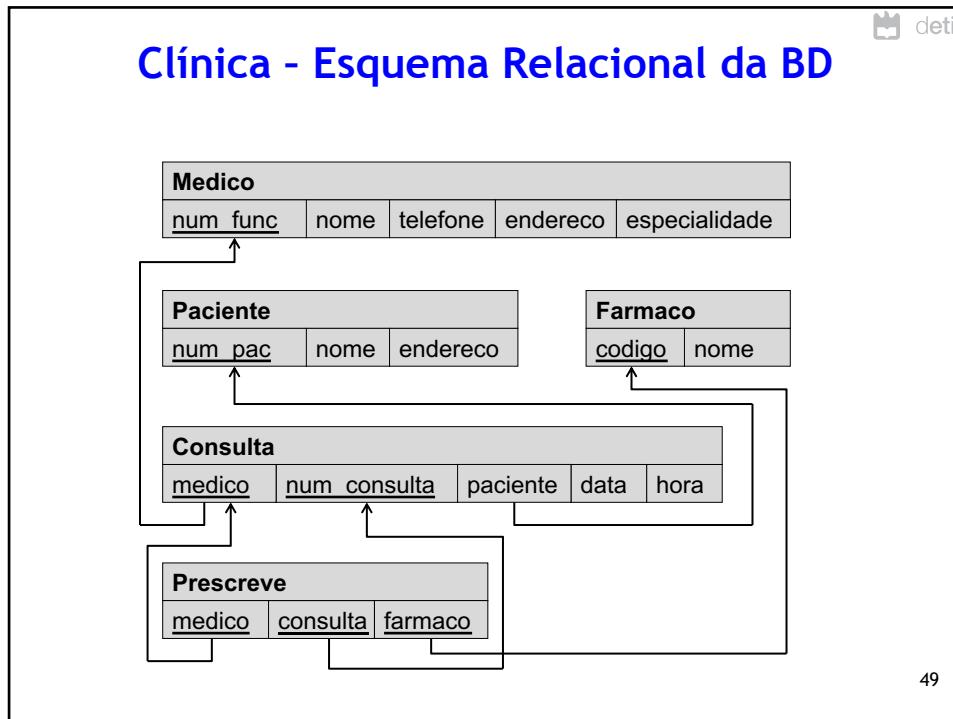
→ O SGBD implementa isto iterativamente!
→ Para cada tuplo da Query exterior executa a Query interior
✓ Pouco eficiente!

• Optimizar BD dá muito €!

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
----------------------	--------------------

↑

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

Agregação

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
----------------------	--------------------

↑

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

Agregação

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

```

    graph TD
        Paciente["Paciente  
num_pac | nome | endereço"]
        Consulta["Consulta  
medico | num_consulta | paciente | data | hora"]
        Prescreve["Prescreve  
medico | consulta | farmaco"]
        Fármaco["Fármaco  
codigo | nome"]

        Paciente --> Consulta
        Consulta --> Prescreve
        Prescreve --> Fármaco
    
```

$$\begin{aligned} \text{temp} &\leftarrow \Pi_{\text{medico}, \text{num_consulta}} (\sigma_{\text{paciente}=35312161} (\text{Consulta})) \\ \text{temp2} &\leftarrow \Pi_{\text{farmaco}, \text{quantidade}=\text{count}(\text{farmaco})} (\text{temp} \bowtie_{\text{medico}=\text{medico} \text{ AND } \text{num_consulta}=\text{consulta}} \text{Prescreve}) \\ &\Pi_{\text{nome}, \text{quantidade}} (\text{temp2} \bowtie_{\text{farmaco}=\text{codigo}} \text{Farmaco}) \end{aligned}$$

SQL...

```

SELECT nome, quantidade
FROM Farmaco JOIN (SELECT farmaco, count(farmaco) AS quantidade
                     FROM Prescreve AS T1
                     JOIN (SELECT medico, num_consulta
                           FROM Consulta
                           WHERE paciente=35312161) AS T
                           ON (T1.medico=T.medico AND T.num_consulta=T1.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

```

    graph TD
        Médico["Médico  
num_func | nome | telefone | endereço | especialidade"]
        Consulta["Consulta  
medico | num_consulta | paciente | data | hora"]
        Prescreve["Prescreve  
medico | consulta | farmaco"]
        Fármaco["Fármaco  
codigo | nome"]

        Médico --> Consulta
        Consulta --> Prescreve
        Prescreve --> Fármaco
    
```

$$\begin{aligned} \text{temp} &\leftarrow \rho_{\text{codigo}, \text{num_func}} (\Pi_{\text{farmaco}, \text{medico}} (\text{Prescreve})) \div \Pi_{\text{num_func}} (\text{Medico}) \\ &\Pi_{\text{nome}} (\text{temp} \bowtie \text{Farmaco}) \end{aligned}$$

÷ não existe em SQL

SQL... Uma Implementação Alternativa da Query:

```

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

na query total

