Gestão de Banco de Dados

Engenharia de Software

Aula 10: Composição de Relações (Join) - Exercícios

Anrafel Fernandes Pereira

http://about.me/Anrafel

Modificações no Banco de Dados

As instruções para modificar a instância de banco de dados serão demonstradas, basicamente, sobre o seguinte esquema:

Modelo Relacional (modelo lógico):

cargo

CdCargo	NmCargo	VrSalario

depto

CdDepto	NmDepto	Ramal

funcionario

NrMatric	NmFunc	DtAdm	Sexo	CdCargo	CdDepto

Modificações no Banco de Dados

Inserção (*insert*)

Para inserir dados em uma relação podemos especificar uma tupla a ser inserida ou escrever uma consulta cujo resultado é um conjunto de tuplas a inserir. Obviamente, os valores dos atributos para as tuplas a inserir devem pertencer ao domínio desses atributos. Similarmente, tuplas a inserir devem possuir a ordem correta (mesmo esquema).

```
insert into NomeDaRelação
  values (ValorDoAtributo1, ValorDoAtributo2, ...,
    ValorDoAtributoN)
```

Modificações no Banco de Dados

Inserção (*insert*)

```
Inserindo uma tupla.
insert into cargo
  values(1, 'Programador Analista', 2500.00)

insert into cargo
  values(2, 'DBA', 4700.00)

insert into cargo
  values(3, 'Suporte', 800.00)
```

```
select * from cargo
```

CDCARGO	NMCARGO	VRSALARIO
1	Programador Analista	2.500,00
2	DBA	4.700,00
3	Suporte	800,00

Modificações no Banco de Dados

Inserção (*insert*)

```
Inserindo uma tupla e usando apenas alguns atributos.
insert into cargo(cdcargo, nmcargo)
  values(1, 'Programador Analista')

Seria equivalente a seguinte instrução.
insert into cargo
  values(1, 'Programador Analista', null)
```

```
Inserindo uma tupla e usando apenas alguns atributos.
insert into cargo(cdcargo, vrsalario)
  values(2, 5000.00)

Seria equivalente a seguinte instrução.
insert into cargo
  values(2, null, 5000.00)
```

```
select * from cargo
```

CDCARGO	NMCARGO	VRSALARIO
1	Programador Analista	<null>
2	<null>	5.000,00

Modificações no Banco de Dados

Remoção (*delete*)

Um pedido de remoção de dados é expresso muitas vezes do mesmo modo que uma consulta. Pode-se remover somente tuplas inteiras; não é possivel, por exemplo, excluir valores de um atributo em particular.

```
delete from r
  where P
```

em que P representa um predicado e r, uma relação. O comando **delete** encontra primeiro todas as tuplas t em r para as quais $P(t)$ é verdadeira e então remove-as de r. A cláusula **where** pode ser omitida nos casos de remoção de todas as tuplas de P.

Modificações no Banco de Dados

Remoção (*delet*)

```
Remove todos os funcionários com CdCargo = 1.
delete from func
  where CdCargo = 1

Remove todos os funcionários com CdCargo = 1 e
com CdDepto = 1.
delete from func
  where (CdCargo = 1) and (CdDepto = 1)

Remove todos os funcionários.
delete from func
```

*O pedido **delete** por conter um **select** aninhado, como por exemplo, para remover todos os funcionários com o valor do atributo Salary maior que a média do próprio atributo.*

```
delete from employee
where salary > (select avg(salary) from employee)
```

Modificações no Banco de Dados

Atualizações (*update*)

```
Aumenta 10% os salário de todos os cargos.
update cargo
set vrSalario = vrSalario * 1.10

Aumenta em R$ 50,00 os salário inferiores a R$ 1.000,00.
update cargo
set vrSalario = vrSalario + 50.00
where vrSalario < 1000.00

Modifica o nome e o salário do CdCargo = 1.
update cargo
set nmCargo = 'Programador Analista Senior',
    vrSalario = 4500.00
where cdCargo = 1

Aumenta 5% os salário dos cargos com salário abaixo da média.
update cargo
set vrSalario = vrSalario * 1.05
where vrSalario < (select avg(vrSalario) from cargo)
```

Operações de Conjuntos

Os operadores SQL-92 **union**, **intersect** e **except** operam relações e correspondem às operações de união ($\cup$), interseção ($\cap$) e diferença ($-$) da álgebra relacional, e portanto, as relações participantes devem ser compatíveis, ou seja, apresentar o mesmo conjunto de atributos (ou esquema).

A SQL-89 possui diversas restrições para o uso de **union**, **intersect** e **except**.

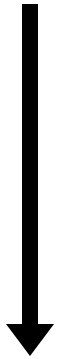
Certos produtos não oferecem suporte para essas operações.

A Operação de União (∪)

Una todas as tuplas da relação “employee” para as quais o valor do atributo “dept_no” seja igual a 120 com as tuplas da relação “employee” cujo o valor do atributo “dept_no” seja igual a 600.

```
select full_name, salary, dept_no from employee
where dept_no = 120
union
select full_name, salary, dept_no from employee
where dept_no = 600
```

FULL_NAME	SALARY	DEPT_NO
Bennet, Ann	22.935,00	120
Brown, Kelly	27.000,00	600
Nelson'', Robert	105.900,00	600
Reeves, Roger	33.620,63	120
Stansbury, Willie	39.224,06	120



$$A \leftarrow \pi_{\text{full\_name, salary, hire\_date}} (\sigma_{\text{dept\_no = 120}}(\text{employee}))$$
$$B \leftarrow \pi_{\text{full\_name, salary, hire\_date}} (\sigma_{\text{dept\_no = 600}}(\text{employee}))$$
$$A \cup B$$

Valores Nulos

O valor *null* indica a ausência de informação sobre o valor de um atributo. Sendo assim, pode-se usar a palavra-chave *null* como predicado para testar a existência de valores nulos.

```
select * from customer
inner join sales
on customer.cust_no = sales.cust_no
where phone_no is null
```

CUST_NO	CUSTOMER	CONTACT_FIRST	CONTACT_LAST	PHONE_NO
1.007	Mrs. Beauvais	<null>	Mrs. Beauvais	<null>

O predicado *not null* testa a ausência de valores nulos.

Qual a diferença nas duas consultas?

```sql
SELECT * FROM clientes c, enderecos e WHERE c.id = e.id_cliente;
```

```sql
SELECT * FROM clientes c JOIN enderecos e ON c.id = e.id_cliente;
```

Qual a diferença nas duas consultas?

```
SELECT * FROM clientes c, enderecos e WHERE c.id = e.id_cliente;
```

```
SELECT * FROM clientes c JOIN enderecos e ON c.id = e.id_cliente;
```

Algebricamente as consultas são idênticas e tem a mesma performance.

A primeira está no padrão **ANSI 89**, já a segunda está no padrão **ANSI 92**.

Composição de Relações

Além de fornecer o mecanismo básico do produto cartesiano para a composição das tuplas de uma relação disponível nas primeiras versões da SQL, a SQL-92 também oferece diversos outros mecanismos para composição de relações como as junções condicionais e as junções naturais, assim como várias formas de junções externas.

- Junção interna (ou junção condicional): **inner join**

- Junção externa à esquerda: **left outer join**
- Junção externa à direita: **right outer join**
- Junção externa total: **full outer join**

Junção Interna

A conexão interna inicialmente faz a mesma coisa que a conexão cruzada, porém aplica restrições que podem ser de igualdade ou desigualdade, isso faz com que algumas linhas sejam eliminadas do resultado.

INNER JOIN, NATURAL JOIN e STRAIGHT_JOIN

Digamos que eu necessite fazer uma busca de todos os programadores e suas respectivas empresas.

Como podemos resolver isso?

Junção Interna

```
SELECT * FROM programadores p INNER
JOIN empresas e ON e.id_empresa = p.id_empresa
```

Na consulta acima, fizemos uma junção entre as duas tabelas que guardam as informações que precisamos e fizemos uma restrição (ON) comparando a chave da empresa existente nas duas tabelas.

Junção Interna

id_programador	id_empresa	nome	id_empresa	nome
1	1	Paulinha	1	Empresa 1
1	1	Paulinha	2	Empresa 2
1	1	Paulinha	3	Empresa 3
1	1	Paulinha	4	Empresa 4
1	1	Paulinha	5	Empresa 5
2	2	Pedro	1	Empresa 1
2	2	Pedro	2	Empresa 2
2	2	Pedro	3	Empresa 3
2	2	Pedro	4	Empresa 4
2	2	Pedro	5	Empresa 5
3	3	Márcia	1	Empresa 1
3	3	Márcia	2	Empresa 2
3	3	Márcia	3	Empresa 3
3	3	Márcia	4	Empresa 4
3	3	Márcia	5	Empresa 5

Somente as linhas em vermelho aparecerão no resultado.

4	4	Filipi	1	Empresa 1
4	4	Filipi	2	Empresa 2
4	4	Filipi	3	Empresa 3
4	4	Filipi	4	Empresa 4
4	4	Filipi	5	Empresa 5
5	4	Pinter	1	Empresa 1
5	4	Pinter	2	Empresa 2
5	4	Pinter	3	Empresa 3
5	4	Pinter	4	Empresa 4
5	4	Pinter	5	Empresa 5
6	1	Gabriel	1	Empresa 1
6	1	Gabriel	2	Empresa 2
6	1	Gabriel	3	Empresa 3
6	1	Gabriel	4	Empresa 4
6	1	Gabriel	5	Empresa 5

Junção Interna

A restrição (ON) não impede que utilizemos as outras opções da sintaxe do SELECT, por exemplo, o WHERE.

Inclusive poderíamos obter o mesmo resultado com a consulta abaixo:

```
SELECT * FROM programadores p, empresas e
WHERE e.id_empresa = p.id_empresa;
```

Junção Interna

Vamos para uma segunda situação:

O diretor da empresa está precisando de uma lista com todos os programadores que programam em pelo menos uma linguagem e saber quais são estas.

Só que eu tenho um problema:

Existe um relacionamento n:n que originou a tabela programadores_linguagens, como resolvo?

Junção Interna

```
SELECT p.nome, l.nome
FROM programadores p
INNER JOIN programadores_linguagens pl
ON pl.id_programador = p.id_programador
INNER JOIN linguagens l
ON l.id_linguagens = pl.id_linguagens;
```

nome	nome
Paulinha	ASP
Paulinha	PHP
Pedro	Java
Márcia	Ruby
Márcia	.NET
Filipi	.NET

Junção Interna

Reparem que só apareceram os programadores que estão relacionados à alguma linguagem de programação.

Mas, e se fosse necessário a presença de todos os programadores na listagem?

To be continued...

Junção Interna (Outro exemplo)

Relaciona (ou junta) através do atributo "cust_no" cada tupla da relação "customer" com as suas tuplas correspondentes na relação "sales". Cada tupla resultante dessa primeira relação é juntada com a tupla correspondente na relação "employee" através do predicado "**on** s.sales_rep = e.emp_no".

```
select c.cust_no, customer,
  po_number, ship_date, total_value, sales_rep,
  full_name
from customer c

inner join sales s on c.cust_no = s.cust_no

inner join employee e on s.sales_rep = e.emp_no
```

Junção Interna (Outro exemplo)

CUST_NO	CUSTOMER	PO_NUMBER	SHIP_DATE	TOTAL_VALUE	SALES_REP	FULL_NAME
1.001	Signature Design	V9324200	09.08.1993 00:00	560.000,00	72	Sutherland, Claudia
1.001	Signature Design	V9324320	16.08.1993 00:00	0,00	127	Yanowski, Michael
1.001	Signature Design	V9320630		60.000,00	127	Yanowski, Michael
1.001	Signature Design	V9420099		3.399,15	127	Yanowski, Michael
1.001	Signature Design	V9427029	10.02.1994 00:00	422.210,97	127	Yanowski, Michael
1.002	Dallas Technologies	V9333005	03.03.1993 00:00	600,50	11	Weston, K. J.
1.002	Dallas Technologies	V9333006	02.05.1993 00:00	20.000,00	11	Weston, K. J.
1.002	Dallas Technologies	V9336100	01.01.1994 00:00	14.850,00	11	Weston, K. J.

Junção Interna (Continuação)

O NATURAL JOIN faz exatamente a mesma coisa que o INNER JOIN em questão de resultado, porém, com suas particularidades:

NATURAL JOIN: com ele você não precisa identificar quais colunas serão comparadas, pois ele fará a comparação entre campos com mesmo nome.

```
SELECT * FROM programadores NATURAL JOIN empresas;
```

Junção Interna

Algumas Observações

- Posso substituir o **ON** por **USING** quando o nome nas duas tabelas for idêntico.

```
SELECT * FROM programadores p INNER JOIN empresas
e USING(id_empresa);
```

- O uso do **INNER** é opcional.

```
SELECT * FROM programadores p JOIN empresas
e USING (id_empresa);
```

Junção Externa

As conexões externas servem para efetuar junções entre tabelas sem que necessariamente exista entre elas uma combinação exata. O **LEFT** e o **RIGHT OUTER JOIN** são os componentes desse tipo de conexão. Não é obrigado o uso do termo **OUTER**, se você encontrar apenas **LEFT JOIN**, por exemplo, funcionará da mesma maneira.

O **OUTER JOIN** pode ser utilizado quando você quiser retornar uma lista de todos os programadores, mesmo que estes não estejam relacionados a nenhuma linguagem de programação, por exemplo.

A diferença do LEFT para o RIGHT está apenas na identificação de qual tabela da junção irá retornar todos os dados. O mais comum é o LEFT, pois normalmente colocamos a tabela mais importante primeiro. Inclusive a ordem das tabelas e das cláusulas de restrições na consulta, em alguns bancos, altera o desempenho.

Junção Externa

A diferença do **LEFT** para o **RIGHT** está apenas na identificação de qual tabela da junção irá retornar todos os dados.

O mais comum é o **LEFT**, pois normalmente colocamos a tabela mais importante primeiro. Inclusive a ordem das tabelas e das cláusulas de restrições na consulta, em alguns bancos, altera o desempenho.

Junção Externa à Esquerda (left outer join)

As operações de "junção externa" são uma extensão da operação de junção interna (inner join) para tratar informações omitidas.

empregado

NOME_EMP	RUA	CIDADE
Alzemiro	Rua A	Pato Branco
Arnilda	Rua B	Mariópolis
Juca	Rua C	Vitorino
Maria	Rua D	Verê

empregado_depto

NOME_EMP	DEPTO	SALARIO
Arnilda	X	1.500,00
Juca	X	1.300,00
Maria	Y	5.300,00
Tadeu	Z	1.500,00

```
select * from empregado e
inner join empregado_depto d
on e.nome_emp = d.nome_emp
```

NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00

Junção Interna

Os empregados "Alzemiro" e "Tadeu" não participam da relação resultado porque não possuem valores nas duas relações envolvidas.

Junção Externa à Esquerda (left outer join)

```
select * from empregado e
left outer join empregado_depto d
on e.nome_emp = d.nome_emp
```

NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Alzemiro	Rua A	Pato Branco	<null>	<null>	<null>
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00

Junção Externa à Esquerda

Acrescenta a relação resultado "todas" as tuplas da relação à esquerda que não encontram par entre as tuplas da relação à direita, preenchendo com valores ***nulo*** todos os outros atributos da relação a direita.

Junção Externa à Direita (right outer join)

A "junção externa à direita" acrescenta a relação resultado "todas" as tuplas da relação à direita que não encontram par entre as tuplas da relação à esquerda, preenchendo com valores *nulo* todos os outros atributos da relação a direita.

empregado

NOME_EMP	RUA	CIDADE
Alzemiro	Rua A	Pato Branco
Arnilda	Rua B	Mariópolis
Juca	Rua C	Vitorino
Maria	Rua D	Verê

empregado_depto

NOME_EMP	DEPTO	SALARIO
Arnilda	X	1.500,00
Juca	X	1.300,00
Maria	Y	5.300,00
Tadeu	Z	1.500,00

```
select * from empregado e
inner join empregado_depto d
on e.nome_emp = d.nome_emp
```

NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00

Junção Interna

Os empregados "Alzemiro" e "Tadeu" não participam da relação resultado porque não possuem valores nas duas relações envolvidas.

Junção Externa à Direita (right outer join)

```
select * from empregado e
right outer join empregado_depto d
on e.nome_emp = d.nome_emp
```

NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00
<null>	<null>	<null>	Tadeu	Z	1.500,00

Junção Externa Total (full outer join)

A "junção externa total" acrescenta a relação resultado as tuplas da relação à esquerda que não encontram par entre as tuplas da relação à direita, assim como as tuplas da relação à direita que não encontram par entre as tuplas da relação à esquerda.

empregado

NOME_EMP	RUA	CIDADE
Alzemiro	Rua A	Pato Branco
Arnilda	Rua B	Mariópolis
Juca	Rua C	Vitorino
Maria	Rua D	Verê

empregado_depto

NOME_EMP	DEPTO	SALARIO
Arnilda	X	1.500,00
Juca	X	1.300,00
Maria	Y	5.300,00
Tadeu	Z	1.500,00

```
select * from empregado e
inner join empregado_depto d
on e.nome_emp = d.nome_emp
```

NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00

Junção Interna

Os empregados "Alzemiro" e "Tadeu" não participam da relação resultado porque não possuem valores nas duas relações envolvidas.

Junção Externa Total (full outer join)

```
select * from empregado e
full outer join empregado_depto d
on e.nome_emp = d.nome_emp
```

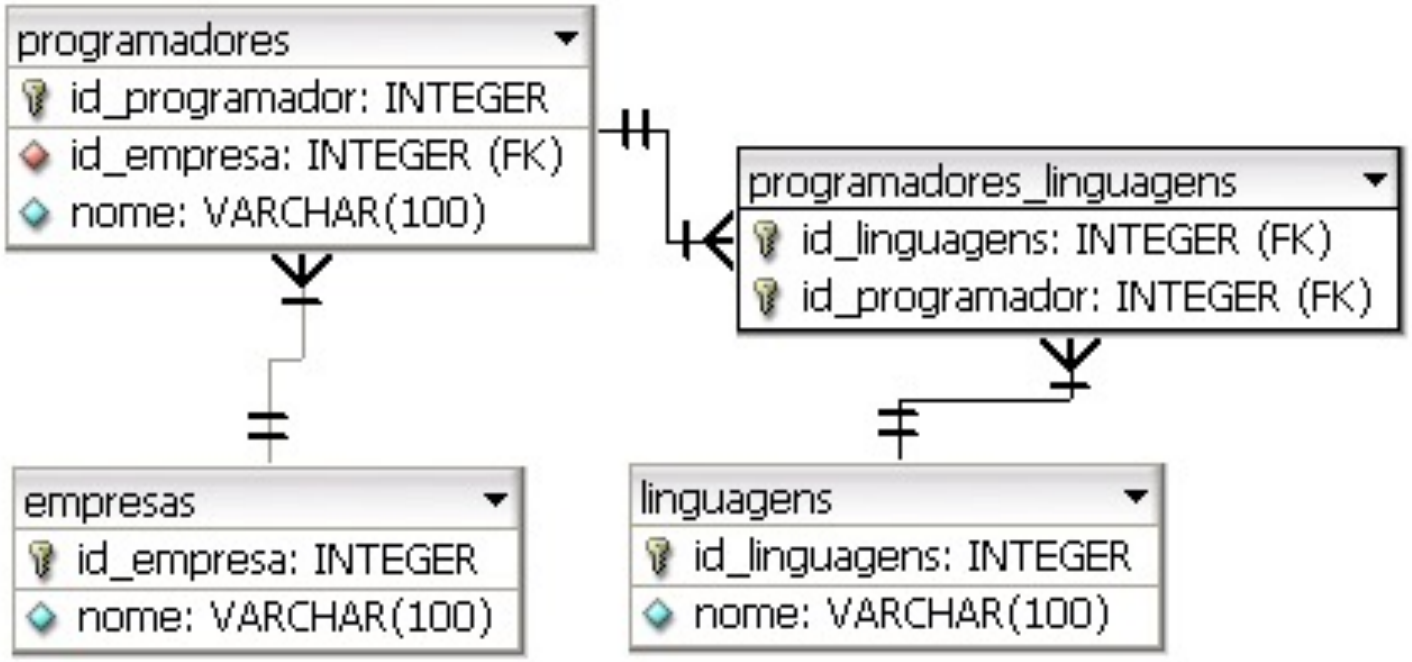
NOME_EMP	RUA	CIDADE	NOME_EMP1	DEPTO	SALARIO
Arnilda	Rua B	Mariópolis	Arnilda	X	1.500,00
Juca	Rua C	Vitorino	Juca	X	1.300,00
Maria	Rua D	Verê	Maria	Y	5.300,00
<null>	<null>	<null>	Tadeu	Z	1.500,00
Alzemiro	Rua A	Pato Branco	<null>	<null>	<null>

Algumas razões para usar o padrão ANSI92 (JOIN)

- **Mais legível**, com os critérios de união separadas da cláusula WHERE, pois não se sabe a primeira vista se condições na clásula WHERE são filtros ou junções.
- **Menos provável que perca critérios de união,** na primeira consulta se não especificarmos o critério na cláusula **WHERE** o resultado será o produto cartesiano entre as tabelas clientes, endereços.
- **Evolução**, se o padrão **ANSI 92** especifica operadores de junções específicas, por que não usá-los?
- **Flexibilidade**, uma junção na cláusula **WHERE** que tenha um efeito de **INNER JOIN** e posteriormente precise ser alterado para **OUTER** pode ser bem mais complicado.

Exercícios

Dado o seguinte esquema relacional, deseja-se realizar as seguintes consultas:



Exercícios

1) Quais são os programadores que programam em PHP (Nome e empresa em que trabalham).
2) Exiba uma lista com todos os programadores e suas empresas independente se eles estão relacionados a uma empresa ou não.
3) Exiba a lista de todas as empresas e seus programadores, independente se a empresa tem ou não programadores.
4) Liste o nome e o ID de todos os programadores da empresa ABC e as linguagens em que eles tem habilidades.
5) Com suas palavras: O que você entendeu sobre junções internas e externas? Cite um exemplo diferente do exemplo citado na apresentação em que poderíamos aplicar estes tipos de junções.