TAKAI, O.K.; ITALIANO, I.C; FERREIRA, J.E. Introdução a Banco de Dados. DCC-IME-USP. Fev, 2005 (página 49-65).

**Álgebra Relacional**

Álgebra relacional é uma coleção de operações utilizadas para manipular relações. Essas operações são usadas para selecionar tuplas de uma determinada relação ou para combinar tuplas relacionadas a diversas relações com o propósito de especificar uma consulta - uma requisição de recuperação - sobre a base de dados.

As operações da álgebra relacional são normalmente divididas em dois grupos. O primeiro deles inclui um conjunto de operações da teoria de conjuntos. As operações são UNION, INTERSECTION, DIFFERENCE e CARTESIAN PRODUCT. O segundo grupo consiste de operações desenvolvidas especificamente para bases de dados relacionais, tais como: SELECT, PROJECT e JOIN entre outras.

**1. O Operador SELECT**

A operação SELECT é usada para selecionar um subconjunto de tuplas de uma relação as quais devem satisfazer uma condição de seleção. Por exemplo, a seleção de um subconjunto de tuplas da relação EMPREGADOS que trabalham para o departamento 4 ou que tenham salário maior que 3000. Cada uma dessas condições é especificada individualmente usando a operação SELECT como segue:

σ NDEP = 4 (EMPREGADO)

σ SALÁRIO > 3000 (EMPREGADO)

Em geral, a operação SELECT é denotada por:

**σ <condição de seleção> (<nome da relação>)**

Onde, o símbolo **σ** é usado para denotar o operador SELECT e a condição de seleção é uma expressão Booleana especificada sobre atributos da relação especificada.

A relação resultante da operação SELECT tem os mesmos atributos da relação especificada em <nome da relação>. A expressão booleana especificada em <condição de seleção> é construída a partir de cláusulas da forma:

**<nome de atributo> <operador de comparação> <valor constante>, ou**

**<nome de atributo> <operador de comparação> <nome de atributo>**

Onde <nome de atributo> é o nome de um atributo da <nome da relação>, <operador de comparação> é normalmente um dos operadores relacionais {=,<,>, ...} <valor constante> é um valor constante. As cláusulas podem ser utilizadas em conjunto com os operadores lógicos {AND, OR NOT} para formar uma condição de seleção composta.

Por exemplo, suponha que se deseja selecionar as tuplas de todos os empregados que ou trabalham no departamento 4 com salário superior a R$2.500,00 ou trabalham no departamento 5 e ganham mais que R$3.000,00. Neste caso, pode-se especificar a consulta da seguinte forma:

**σ (NDEP = 4 AND SALÁRIO > 2500) OR (NDEP = 5 AND SALÁRIO > 3000) (EMPREGADO)**

O operador SELECT é unário; isto é, ele é aplicado somente a uma relação. Assim, o SELECT não pode ser usado para selecionar tuplas de mais de uma relação. Observe também que a operação de seleção é aplicada individualmente para cada tupla. Assim, as condições de seleção não podem ser aplicadas a mais que uma tupla. O grau da relação resultante é a mesma que a relação original. O número de tuplas da relação resultante é sempre menor ou igual ao número de tuplas da relação original.

Note que o operador SELECT é comutativo; isto é,

**σ <cond1> (σ <cond2> (R))= σ<cond2> (σ<cond1> (R))**

Assim, uma seqüência de SELECTs pode ser aplicada em qualquer ordem. Além disso, pode-se sempre trocar operadores SELECT em cascata com a conjuntiva AND; isto é:

**σ <cond1> (σ <cond2> (...(σ <condn> (R))...))= σ <cond1> AND <cond2> AND ... AND <condn>(R)**

**2. Operações da Teoria dos Conjuntos**

O próximo grupo de operações da álgebra relacional são as operações matemáticas padrões sobre conjuntos. Elas se aplicam ao modelo relacional porque uma relação é definida como um conjunto de tuplas.

Existem várias operações da teoria de conjuntos que são utilizadas para agrupar elementos de dois conjuntos, entre elas estão: UNION, INTERSECTION e DIFFERENCE. Estas operações são binárias; isto é, elas necessitam de dois conjuntos. Quando essas operações são adaptadas para a base de dados relacional, deve-se assegurar que essas operações resultem sempre em relações válidas. Para conseguir isso, as duas relações aplicadas a qualquer uma das três operações acima devem ter o mesmo tipo de tuplas; esta condição é chamada união compatível. Duas relações R(A1, A2, ..., An) e S(B1, B2, ..., Bn) são união compatível se elas tiverem o mesmo grau n, e dom(Ai)=dom(Bi) para 1 ≤ i ≤ n. Isso significa que as duas relações têm o mesmo número de atributos e que cada par de atributos correspondentes tem o mesmo domínio.

Pode-se definir as três operações UNION, INTERSECTION e DIFFERENCE sobre duas relações que sejam união compatível R e S:

• UNION ⎯ O resultado da operação, denotado por R ∪ S, é uma relação que inclui todas as tuplas de R e todas as tuplas de S. Tuplas duplicadas são eliminadas.

• INTERSECTION ⎯ O resultado desta operação, denotado por R ∩ S, é a relação que inclui todas as tuplas que são comuns a R e S.

• DIFFERENCE ⎯ O resultado desta operação, denotado por R - S, é a relação que inclui todas as tuplas de R, mas que não estão em S.

Note que as operações UNION e INTERSECTION são operações comutativas:

**R ∪ S = S ∪ R, e R ∩ S = S ∩ R.**

Estas operações podem ser aplicadas a qualquer número de relações, e ambas são associativas:

**R ∪ (S ∪ T) = (R ∪ S) ∪ T, e R ∩ (S ∩ T) = (R ∩ S) ∩ T.**

A operação DIFFERENCE não é comutativa:

**R - S ≠ S - R.**

A operação CARTESIAN PRODUCT, denotada por χ, é também uma operação de conjunto binária, mas as relações sobre as quais são aplicadas não necessitam ser união compatível.

Esta operação é usada para combinar tuplas de duas relações tal que tuplas relacionadaspossam ser identificadas. Em geral, o resultado de **R(A1, A2, ..., An) χ S(B1, B2, ..., Bm)** é a relação Q com n + m atributos **Q(A1, A2, ..., An, B1, B2, ..., Bm)** nesta ordem. A relação resultante Q tem uma tupla para cada combinação de tuplas. Assim, se R tem nR tuplas e S tem nS tuplas, então RχS terá nR\*nS tuplas.

O CARTESIAN PRODUCT cria tuplas com atributos combinados de duas relações. Pode-se então selecionar apenas as tuplas que estejam relacionadas especificando uma condição de seleção apropriada, como foi feita no exemplo. Devido à seqüência: CARTESIAN PRODUCT seguido de SELECT, ser muito comum para se identificar tuplas relacionadas de duas relações, uma operação especial JOIN foi criada para especificar esta seqüência como uma única operação. Assim, a operação CARTESIAN PRODUCT é raramente utilizada isoladamente.

**3. A Operação JOIN**

A operação JOIN, denotada por ⌧, é usada para combinar tuplas relacionadas de relações em uma única tupla. Esta operação é muito importante para quaisquer bases de dados relacionais, pois permite processar relacionamentos entre relações. Para ilustrar a operação JOIN, suponha que se deseja recuperar os nomes dos gerentes de cada departamento. Para obter-se o nome dos gerentes, é necessário combinar cada tupla de departamento com tuplas de empregados cujo valor NSS seja igual ao valor de NSSGER na tupla departamento. Isto é feito usando a operação JOIN, então projeta-se o resultado sobre aqueles atributos necessários:

**DEPT\_GER←DEPARTAMENTO ⌧NSSGER=NSS EMPREGADO**

**RESULT←πDNOME, SNOME, PNOME (DEPT\_GER)**

O exemplo utilizado para ilustrar o CARTESIAN PRODUCT pode ser especificado usando o operador JOIN trocando as duas operações:

**EMP\_DEP←EMP\_NOMES χ DEPENDENTE**

**DEP\_ATUAL←σNSS=NSSEMP (EMP\_DEP)**

por

**DEP\_ATUAL←EMP\_NOME ⌧NSS=NSSEMP DEPENDENTE**

A forma geral da operação JOIN sobre duas relações R(A1, A2, ..., An) e S(B1, B2, ..., Bm) é:

**R ⌧<condição join> S.**

O resultado de JOIN é uma relação Q com n+m atributos Q(A1, A2, ..., An, B1, B2, ..., Bm) nesta ordem; Q tem um tupla para cada combinação de tuplas ⎯ uma de R e uma de S ⎯ onde quer que a combinação satisfaça a condição join. Esta é a principal diferença entre CARTESIAN PRODUCT e JOIN; em JOIN, apenas combinações de tuplas que satisfazem a condição join é que aparecerá no resultado, já no CARTESIAN PRODUCT, todas as combinações de tuplas são incluídas no resultado. A condição join é especificada sobre atributos de R e de S, e é avaliada para cada combinação de tuplas.

Uma condição join tem a forma:

**<condição> AND <condição> AND,...,OR, ..., AND <condição>**

onde cada condição é da forma Ai θ Bj, Ai é um atributo de R, Bj é um atributo de S, Ai e Bj têm o mesmo domínio e θ é um dos operadores de comparação { =, <, ≤, >, ≥, ≠}. Uma operação JOIN de condição especificada é denominada **THETA JOIN**. Tuplas cujos valores dos atributos join são null não aparecem no resultado.

É muito comum encontrar JOIN que envolva condições joins com apenas a comparação de igualdade. Um JOIN, onde apenas o operador de comparação = é utilizado é chamado **EQUIJOIN**. Os dois exemplos anteriores eram EQUIJOIN. Note-se, que no resultado de uma EQUIJOIN sempre terá um ou mais pares de atributos que tem valores idênticos em todas as tuplas.

Devido a esta duplicação ser desnecessária, uma nova operação chamada **NATURAL JOIN** foi criada. Denota-se o join natural por \*, que é basicamente um equijoin seguido da remoção de atributos desnecessários. Um exemplo é:

**PROJ\_DEPT← (PROJETO) \* DNUM=DNÚMERO (DEPARTAMENTO)**

Os atributos DNUM e DNUMERO são chamados atributos de união.

Na relação PROJ\_DEPT, cada tupla combina uma tupla de PROJETO com a tupla de DEPARTAMENTO que controla o projeto. Na relação resultante, manteve-se apenas o primeiro atributo de união. Devido às comparações em uma condição join de um join natural serem sempre comparações de igualdade, pode-se descartar o operador de comparação e apenas listar os atributos de união como segue:

**PROJ\_DEPT←PROJETO \* (DNUM), (DNÚMERO) DEPARTAMENTO**

Assim, a forma geral de um NATURAL JOIN é:

**Q←R \* (<lista1>),(<lista2>) S**

Neste caso, <lista1> especifica uma lista de i atributos de R e <lista2> especifica uma lista de i atributos de S. Apenas os atributos da primeira relação R ⎯ <lista1> ⎯ serão mantidos na relação resultante Q.

Se os atributos sobre os quais o join natural é especificado tiverem os mesmos nomes em ambas relações, pode-se tirar a condição join totalmente. Por exemplo, para aplicar um join natural sobre DNÚMERO de DEPARTAMENTO e LOCAIS\_DEPTO, é suficiente escrever:

**DEPT\_LOCS←DEPARTAMENTO \* LOCAIS\_DEPTO**

Esta operação é executada igualando-se todos os pares de atributos que tenham o mesmo nome nas duas relações.

Note que, se nenhuma combinação de tuplas satisfizer a condição join, então o resultado será uma relação vazia. Em geral, se R tiver nR tuplas e S tiver nS tuplas, o resultado de uma operação JOIN R ⌧ <condição join> S terá entre zero e nR\*nS tuplas. Se não existir <condição join> para satisfazer, então todas as combinações de tuplas serão incluídas. Nestes casos, JOIN torna-se um CARTESIAN PRODUCT.

**ATIVIDADE: Especifique as seguintes consultas sobre a base de dados mostrada na Figura 1.** usando operações relacionais discutidas. Mostre também os resultados de cada consulta se aplicada à base de dados da Figura 2.

a) Recuperar os nomes de empregados do departamento 5 que trabalham mais que 10 horas no projeto 'ProdutoX'.

b) Listar os nomes dos empregados que tenham um dependente com o mesmo nome (PNOME).

c) Encontrar os nomes de empregados que são diretamente supervisionados por 'Franklin Wong'.

d) Para cada projeto, listar o nome do projeto e o total de horas (de todos os empregados) gastos em cada projeto.

e) Recuperar os nomes dos empregados que trabalham em todos os projetos.

f) Recuperar os nomes dos empregados que não trabalham em quaisquer projetos.

g) Para cada departamento, recuperar o nome do departamento e a média salarial dos empregados que trabalham no departamento.

h) Recuperar a média salarial de todos os empregados femininos.

i) Encontrar os nomes e endereços de empregados que trabalham em ao menos um projeto localizado em Houston mas cujo departamento não possua localização em Houston.

j) Listar os sobrenomes dos gerentes de departamentos que não tenham dependentes.

k) Generalize a consulta i) acima para listar os nomes e endereços de empregados que trabalham em um projeto em alguma cidade , mas que o departamento não tenha nenhuma localização nessa cidade.

Figura 1. Esquema da relação

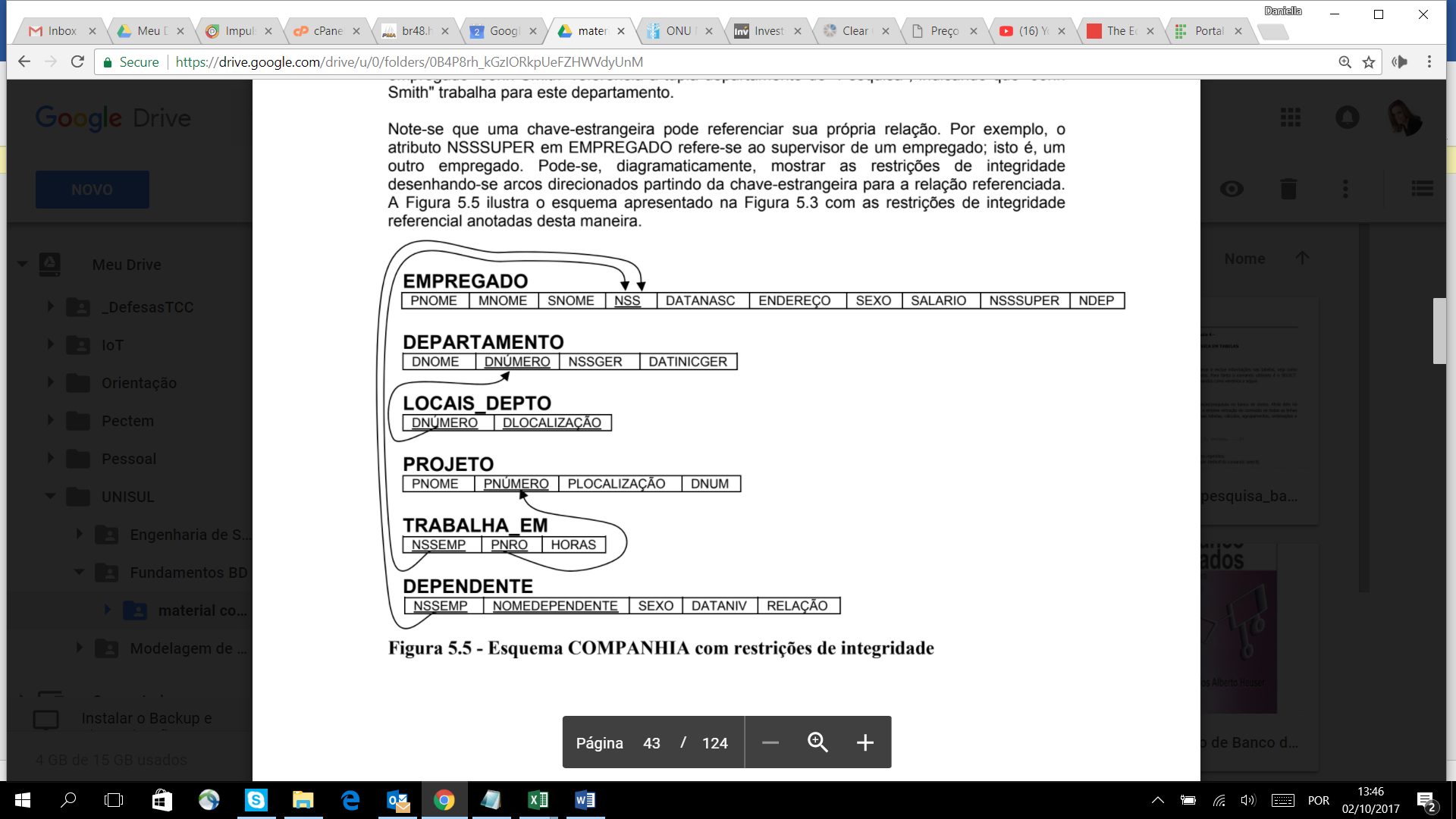


Figura 2. Dados das tabelas

