Álgebra Relacional

Linguagens de consultas relacionais

- *Linguagens de consultas*: Permitem manipulação e recuperação de dados de um BD.
- O modelo relacional suporta LCs simples e poderosas:
  - Forte fundamentação teórica baseada em lógica.
  - Permite otimizações.
- Ling. de consulta ≠ ling. de programação
  - LCs não tem a intenção de suportar cálculos complexos.
  - LCs suportam acesso fácil e eficiente a grandes conjuntos de dados.

LCs relacionais formais

Duas LCs matemáticas formam a base para as LCs "reais" (p.ex., SQL), e p/ implementação:

❶ *Álgebra relacional*: Predominantemente operacional, útil para representar planos de execução.

❷ *Cálculo Relacional* : Permite usuários descreverm *o que* querem, ao invés de *como* querem. (não-operacional, *declarativa*.)

☛ Entender álgebra e cálculo e'uma chave para entender SQL e processamento de consultas.

Preliminares

- Uma consulta é aplicada para *instâncias de relação*, e o resultado de uma consulta é também uma instância de relação.
  - *Esquemas* de consumo relações para uma consulta são fixadas (mas consultas rodarão independente de exemplos!)
  - O esquema para o *resultado* de uma propensa consulta é também fixada! Determinada por definição de construção de linguagem de consulta.

Example Instances

- Relações “Sailors” e “Reserves” para nossos exemplos.
- Usaremos positional ou named field notation, assume que nomes de campos em resultados de consulta são `herdados’ de nomes de campos em relações gastos de consulta.

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Algebra Relacional

- Operações Básicas:
  - *Selection* (σ) Seleciona em sub-conjunto de fileiras da relação.
  - *Projection* (π) Deleta colunas indesejadas da relação.
  - *Cross-product* ($\times$) Permite-nos combinar duas relações.
  - *Set-difference* ($-$) Tuplas em reln. 1, mas não em reln. 2.
  - *Union* ($\cup$) Tuplas em reln. 1 e em reln. 2.
- Operações Adicionais:
  - Intersecção, *junção*, divisão, renomear
- Desde que cada operação retorna uma relação, operações podem ser *compostas* !

Projeção

- Deleta atributos que não estão na *lista de projeção*.
- *Esquema* de resultado contem exatamente o campo na lista de projeção, com os mesmos nomes que eles tinham na (somente) relação gasto.
- Operador de projeção tem eliminar *duplicadas!* (Porque??)
  - Note: sistemas reais tipicamente não fazem eliminação duplicada a menos que o usuário explicitamente peça isso.(Porque não?)

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$

age
35.0
55.5

$$\pi_{age}(S2)$$

Seleção

- Seleciona fileiras que satisfazem *condição seleção*.
- Não duplica no resultado! (Porque?)
- *Esquema* de resultado idêntico para esquema de (somente) relação gasto.
- Relação de *Resultado* pode ser usado para outra operação de algebra relacional !

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

União, Intersecção, Diferença de Conjuntos

- Todas estas operações tomam duas relações de gastos, com os quais tem ser *union-compatible*:
  - Mesmo número de campos.
  - Campos `correspondentes' tem o mesmo tipo.
- O que é o *esquema* de resultado?

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Produto Cartesiano

- Cada fileira de S1 é combinada com cada fileira de R1.
- *Esquema resultante* tem um campo por campo S1 e R1, com nomes campos`herdados' se possível.
  - *Conflito:* Ambos S1 e R1 tem um campo chamado *sid*.

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

☛ *Operador renomear*: $\rho\,(C(1 \to sid1, 5 \to sid2), S1 \times R1)$

Junções

- *Condição Junção*: $R \bowtie_c S = \sigma_c (R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

- *Result schema* o mesmo que do produto cartesiano
- Poucas tuplas do que produto cartesiano, tem que estar disponíveis para computar mais eficientemente.
- Algumas vezes chamada um *theta-join*.

Joins

- *Equijoin*: Um caso especial de condição junção onde a condição *c* contem somente ***igualdades.***

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

$$S1 \bowtie_{sid} R1$$

- *Result schema* parecido com produto cartesiano, mas somente uma copia de campos para o qual igualdade é especificada.
- *Natural Join*: Equijoin em todos campos iguais.

Division

- Não suportado como um operador primitivo, mas útil para expressar consultas como:

  *Find sailors who have reserved **<u>all</u>** boats.*

- Seja A com dois campos, *x* e *y*; e *B* com apenas *y*:
  - $A/B = \{\langle x \rangle \mid \exists \langle x, y \rangle \in A \ \forall \langle y \rangle \in B\}$
  - i.e., ***A/B contém todas as tuplas (*sailors) tais que para <u>cada</u> tupla y (boat) em *B*, há uma tupla *xy* em *A*.**
  - *Ou*: Se o conjunto de valores *y* (boats) associados com um valor *x* (sailor) em *A* contem todos os valores *y* em *B*, o valor *x* está em *A/B*.
- Em geral *x* e *y* podem estar em qualquer listas de campos; *y* é a lista de campo em *B*, e *x* é a lista de campos de *A*.

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

A/B2

pno
p1
p2
p4

B3

sno
s1

A/B3

Expressando A/B Usando Operadores Básicos

- Divisão não é operador essencial; só uma útil taquigrafia.
  - Também é o caso de joins, mas joins são tão comuns que muito sistemas o implementam.
  - *Idéia*: Para *A/B*, calcule todos valores *x* que não são disqualificados por algum *y* em B.
  - *x* é disqualificado se ao juntar um *y* de B, obtemos uma tupla *xy* que não está em A.

Disqualified *x* values: $\pi_x((\pi_x(A) \times B) - A)$

A/B: $\pi_x(A)$ − all disqualified tuples

Encontre os nomes dos sailors que reservaram o barco # 103

- Solução 1: $\pi_{sname}((\sigma_{bid=103} \text{Re}serves) \bowtie Sailors)$

- Solução 2: $\rho\ (Temp1, \sigma_{bid=103} \text{Re}serves)$

  $\rho\ (Temp2, Temp1 \bowtie Sailors)$

  $\pi_{sname}(Temp2)$

- Solução 3: $\pi_{sname}(\sigma_{bid=103}(\text{Re}serves \bowtie Sailors))$

Encontre nomes dos sailors que reservaram um barco vermelho

- Informação sobre cor disponível somente em barcos; assim precisa de uma junção extra:

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \text{Re}serves \bowtie Sailors)$$

- Uma solução mais eficiente:

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie \text{Re}s) \bowtie Sailors)$$

☛ Um otimizador de consulta pode fazer isso a partir da primeira solução !

Encontre nomes dos sailors que reservaram um barco vermelho ou verde

- Pode identificar todos os barcos vermelhos ou verdes, então achar sailors que tenham reservado um destes barcos:

$$\rho\ (Tempboats, (\sigma_{color='red' \vee color='green'} Boats))$$

$$\pi_{sname}(Tempboats \bowtie \text{Re}serves \bowtie Sailors)$$

- Também pode-se definir Tempboats usando união (?)
- E se $\vee$ é substituido por $\wedge$ nesta consulta ?

Encontre os sailors que reservaram um barco vermelho e um barco verde

- Idéia anterior não funciona ! Deve-se identificar os sailor que reservaram barcos vermelhos, aqueles que reservaram barcos verdes e encontrar a interseção destes:

$$\rho\ (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \text{Re}serves))$$

$$\rho\ (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie \text{Re}serves))$$

$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Encontre sailors que reservaram todos os barcos

- Usando divisão, esquemas a serem "divididos" devem ser cuidadosamente escolhidos:

$$\rho\ (Tempsids, (\pi_{sid,bid} \text{Re}serves) / (\pi_{bid} Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

- P/ encontrar sailors que reservaram os barcos 'Interlake':

$$..... \quad / \pi_{bid}(\sigma_{bname='Interlake'} Boats)$$