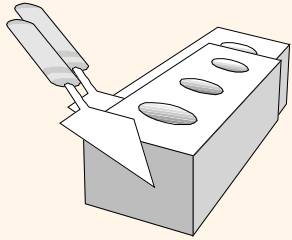
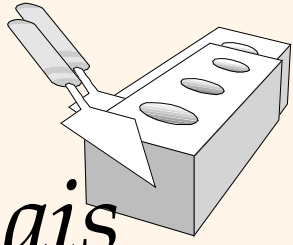


Álgebra Relacional

Capítulo 4, Parte A



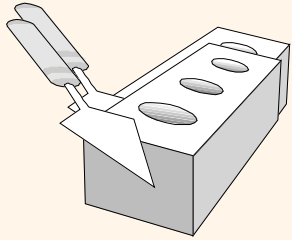
Linguagens de Consulta Relacionais

- *Linguagens de Consulta (Query Languages* – QLs*):* Possibilitam a manipulação e recuperação de dados do banco de dados.
- O modelo relacional suporta QLs simples e poderosas:
  - Fundamentação formal poderosa baseada na lógica.
  - Torna possível maior otimização.
- Linguagens de consulta **!=** linguagens de programação!
  - Não se espera que QLs sejam "Turing completas".
  - QLs não foram pensadas para uso em cálculos complexos.
  - QLs suportam acessos simples e eficientes a extensos conjuntos de dados.

Linguagens de Cons

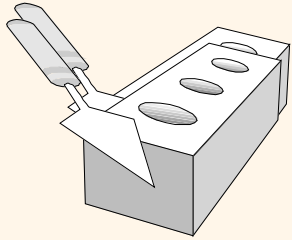
Uma linguagem é dita "Turing completa" se puder ser demonstrado que ela é computacionalmente equivalente à máquina de Turing.

- *Linguagens de Consulta (Quer*
  Possibilitam a manipulação e recuperação de dados do banco de dados.
- O modelo relacional suporta QLs simples e poderosas:
  - Fundamentação formal poderosa baseada na lógica.
  - Torna possível maior otimização.
- Linguagens de consulta **!=** linguagens de programação!
  - Não se espera que QLs sejam "Turing completas".
  - QLs não foram pensadas para uso em cálculos complexos.
  - QLs suportam acessos simples e eficientes a extensos conjuntos de dados.



Linguagens Formais de Consultas Relacionais

- Duas Linguagens de Consulta matemáticas formam as bases para linguagens "reais" (ex. SQL), e para sua implementação:
  - *Álgebra Relacional:* Mais operacional, muito útil para a representação de planos de execução.
  - *Cálculo Relacional:* Permite que o usuário descreva o que ele quer, ao invés de como deve ser computado o que ele quer. (Não operacional, declarativo).



Preliminares

- Uma *query* é aplicada a instâncias de relação, e o resultado da *query* também é uma instância de relação.
  - Os esquemas das relações de entrada para a *query* são fixos (mas a *query* irá executar independentemente da instância!).
  - O esquema do resultado de uma dada *query* também é fixo! Determinado pela definição dos construtores da linguagem *query*.
- Notação posicional vs. nome do campo:
  - Notação posicional é mais fácil para definições formais, notação com nome do campo é mais legível.
  - Ambas são usadas em SQL.

Instâncias Exemplo

- "Marinheiros" e "Reservas" – relações para nossos exemplos.
- Nós iremos usar notação posicional ou com nome do campo, assumindo que os nomes dos campos nos resultados da *query* são 'herdados' dos nomes dos campos das relações de entrada da *query*.

R1

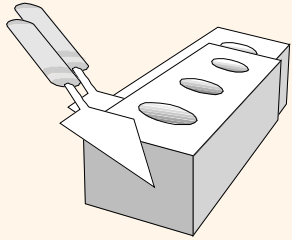
sid	bid	day
22	101	10/10/96
58	103	11/12/96

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

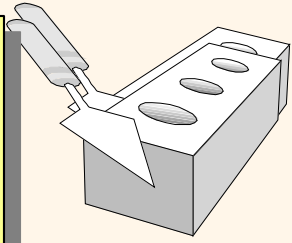


Álgebra Relacional

- Operações Básicas:
  - *Seleção* (σ) Seleciona um subconjunto de linhas da relação.
  - *Projeção* (π) Elimina colunas não desejadas da relação.
  - *Produto cartesiano* (×) Nos permite combinar duas relações.
  - *Diferença de conjuntos* (–) Tuplas da rel. 1, mas que não estão na rel. 2.
  - *União* (∪) Tuplas na rel. 1 e na rel. 2.
- Operações Adicionais:
  - Interseção, *junção*, divisão, renomeação: Não essenciais, mas (muito!) úteis.
- Uma vez que cada operação retorna uma relação, operações podem ser *compostas*! (A Álgebra é "fechada").

Álgebra Relacional

Operação fechada de um conjunto - Produz-se quando o resultado da operação pertence ao mesmo conjunto.



- Operações Básicas:
  - *Seleção* (σ) Seleciona um subconjunto de linhas da relação.
  - *Projeção* (π) Elimina colunas não desejadas da relação.
  - *Produto cartesiano* (×) Nos permite combinar duas relações.
  - *Diferença de conjuntos* (–) Tuplas da rel. 1, mas que não estão na rel. 2.
  - *União* (∪) Tuplas na rel. 1 e na rel. 2.
- Operações Adicionais:
  - Interseção, *junção*, divisão, renomeação: Não essenciais, mas (muito!) úteis.
- Uma vez que cada operação retorna uma relação, operações podem ser *compostas*! (A Álgebra é "fechada").

Projeção

- Elimina os atributos que não estão na *lista de projeção*.
- O *esquema* do resultado contém exatamente os campos da lista de projeção, com os mesmos nomes que eles tinham na (única) relação de entrada.
- O operador de Projeção tem que eliminar *duplicatas*! (Porque??)
  - Note: sistemas reais tipicamente não eliminam duplicatas a menos que o usuário explicitamente o peça. (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 linhas que satisfazem a *condição de seleção*.
- Não há duplicatas no resultado! (Porque?)
- O *esquema* do resultado é idêntico da (única) relação de entrada.
- A relação *resultante* pode ser a *entrada* para outra operação de álgebra relacional! (*Composição de Operadores*)

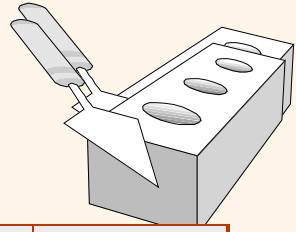
sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

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

sname	rating
yuppy	9
rusty	10

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

União, Interseção e Diferença de conjuntos



- Todas estas operações tem duas relações de entrada, que devem ser *união-compatíveis*:
  - Mesmo número de campos.
  - Campos `correspondentes' do mesmo domínio.
- Qual é o *esquema* do 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

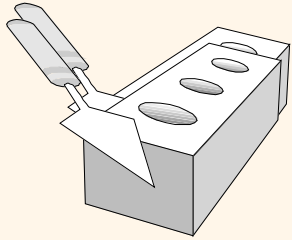
$S1 \cup S2$

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

$S1 \cap S2$

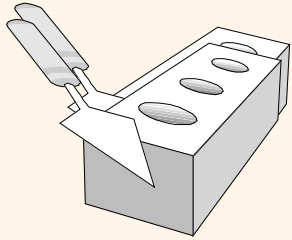


Produto Cartesiano

- Cada linha de S1 é emparelhada com cada linha de R1.
- O *esquema do resultado* tem um campo para cada campo de S1 e R1, com os nomes dos 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 de renomeação*: $\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$



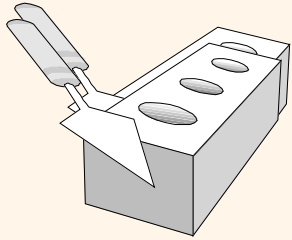
Junções

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

(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

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

- O *esquema resultante* é o mesmo do produto cartesiano.
- Menos tuplas do que o produto cartesiano, pode tornar a computação mais eficiente.
- Algumas vezes chamado *junção-theta*.



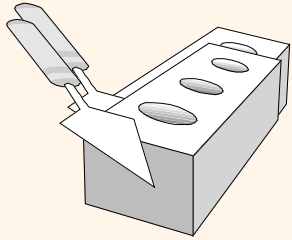
Junções

- *Equi-Junção*: Um caso especial de junção condicional onde a condição *c* contém 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

$$S1 \bowtie_{sid} R1$$

- *Esquema resultante* similar ao produto cartesiano, mas com somente uma cópia dos campos para os quais a igualdade foi especificada.
- *Junção Natural:* Equi-junção sobre *todos* os campos em comum.

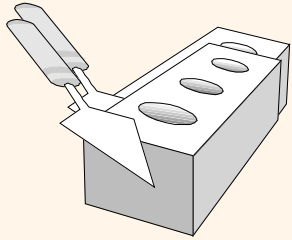


Divisão

- Não é suportado como um operador primitivo, mas é útil para expressar *queries* como:

  Encontre marinheiros que tenham reservado ***todos*** *os barcos.*

- Seja A tendo 2 campos, x e y; B tendo um só campo y:
  - $A/B = \left\{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \ \ \forall \langle y \rangle \in B \right\}$
  - i.e., A/B **contém todas tuplas x (marinheiros) tal que para *cada* tupla y (barco) em B, há uma tupla xy em A.**
  - *Ou*: Se o conjunto de valores y (barcos) associados com valores x (marinheiros) em A contiverem todos os valores y em B, o valor x está em A/B.
- Em geral, x e y podem ser quaisquer listas de campos; y é a lista de campos em B, e $x \cup y$ é a lista de campos de A.



Exemplos de Divisão 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

B

1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

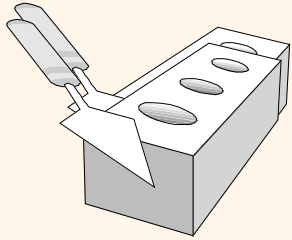
A/B2

pno
p1
p2
p4

B3

sno
s1

A/B3

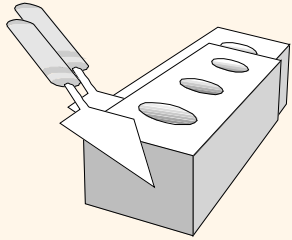


Expressando A/B usando Operadores Básicos

- Divisão não é um operador essencial; somente um atalho útil.
  - (Também é verdadeiro para junções, mas junções são tão comuns que os sistemas as implementam).
- *Idéia*: Para *A/B*, compute todos os valores *x* que não estão `desqualificados' por algum valor *y* em *B*.
  - o valor *x* está *desqualificado* se anexando o valor *y* de *B*, nós obtivermos uma tupla *xy* que não está em A.

Valores *x* desqualificados: $\pi_x((\pi_x(A) \times B) - A)$

A/B: $\pi_x(A)$ − todas as tuplas desqualificadas



Encontre os nomes dos marinheiros que reservaram o barco #103

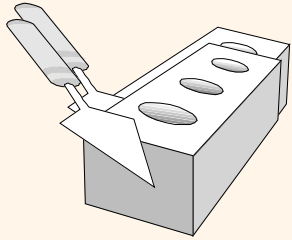
- Solução 1: $\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$

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

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

  $\pi_{sname}(Temp2)$

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



Encontre os nomes dos marinheiros que reservaram um barco vermelho

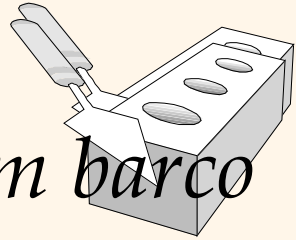
- Informação sobre a cor do barco só está disponível em Boats; então é necessária uma junção extra:

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

- Uma solução mais eficiente:

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

Um otimizador de query pode encontrar isto, dada a primeira solução!



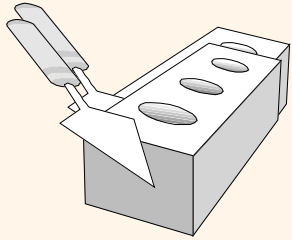
Encontre os marinheiros que reservaram um barco vermelho ou um barco verde

- Pode-se identificar todos os barcos vermelhos ou verdes, então encontrar marinheiros que reservaram um destes barcos:

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

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

- Pode-se também definir Tempboats usando-se a união! (Como?)
- O que acontece se o $\vee$ for substituído pelo $\wedge$ nesta *query*?



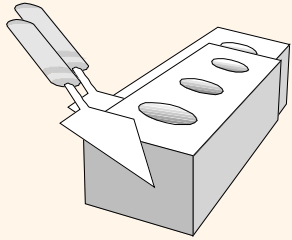
Encontre os marinheiros que reservaram um barco vermelho e um verde

- A abordagem anterior não funciona! Precisamos identificar os marinheiros que reservaram barcos vermelhos, marinheiros que reservaram barcos verdes, e então encontrar a interseção (note que *sid* é a chave para Sailors):

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

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

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



Encontrar os nomes dos marinheiros que reservaram todos os barcos

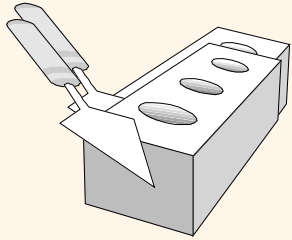
- Usando divisão; esquemas das relações de entrada para / deve ser escolhidos com cuidado:

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

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

- Para encontrar os marinheiros que reservaram todos os barcos 'Interlake':

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



Sumário

- O modelo relacional tem linguagens de consulta rigorosamente definidas que são simples e poderosas.
- A álgebra relacional é mais operacional; útil como representação interna para planos de avaliação de *query*.
- Existem muitos modos de expressão para uma dada *query*; um otimizador de *query* deve escolher a versão mais eficiente.