



# Capítulo 3

## Modelo Relacional

# Modelo Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Modelo Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Estrutura Básica

◆ Dados conjuntos  $A_1, A_2, \dots, A_n$ , uma relação  $r$  é um subconjunto de

$$\mathbf{A_1} \times \mathbf{A_2} \times \dots \times \mathbf{A_n}$$

Assim, uma relação é um conjunto de n-tuplas

$$(a_1, a_2, \dots, a_n)$$

onde  $a_i \in A_i$ , para cada  $i$  de 1 até  $n$

# Estrutura Básica

◆ Exemplo:

Se

**nome\_cliente** = {Jones, Smith, Curry, Lindsay}

**rua\_cliente** = {Main, North, Park}

**cidade\_cliente** = {Harrison, Rye, Pittsfield}

Então  $r = \{(Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield)\}$  é uma relação sobre

**nome\_cliente** × **rua\_cliente** × **cidade\_cliente**

# Esquema de Relação

- ◆ Sejam os atributos  $A_1, A_2, \dots, A_n$
- ◆  $R = (A_1, A_2, \dots, A_n)$  é dito ser um esquema de relação  
Esquema\_cliente = (nome\_cliente, rua\_cliente, cidade\_cliente)
- ◆  $r(R)$  é uma relação no esquema de relação R cliente (Esquema\_cliente)

# Instância de Relação

- ◆ Os valores correntes de uma relação (instância da relação) são especificados por uma tabela.
- ◆ Um elemento  $t$  de  $r$  é uma tupla; representada por uma linha na tabela.

nome_cliente	rua_cliente	cidade_cliente
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

**cliente**

# Observações Gerais

- ◆ Como as tabelas em essência são relações, utiliza-se os termos matemáticos **relação** e **tupla**, no lugar de **tabela** e **linhas**.
- ◆ Como uma **relação** é um conjunto de **tuplas**, podemos usar a notação matemática  $t \in r$  para denotar que a **tupla**  $t$  está na **relação**  $r$ .



# Chaves

- ◆ Seja  $K \subseteq R$
- ◆  $K$  é uma super chave de  $R$  se valores de  $K$  são suficientes para identificar uma única tupla de cada possível relação  $r(R)$ . Por “possível  $r$ ” estamos querendo denotar uma relação que poderia existir na empresa que está sendo modelada.

Exemplo:  $\{\text{cliente\_nome, rua\_nome}\}$  e  $\{\text{cliente\_nome}\}$  são ambas super chaves de **cliente**, se assumirmos que dois clientes não possam ter o mesmo nome.

# Chaves

- ◆ **K** é uma chave candidata se **K** é mínima.  
Exemplo: **{cliente\_nome}** é uma chave candidata para **cliente**, pois ela é super chave (assumindo que dois clientes não possam ter o mesmo nome) e é mínima desde que nenhum subconjunto dela é uma super chave.

# Determinando Chaves a partir dos Conjuntos E-R

- ◆ **Conjunto de entidades fortes.** A chave primária da entidade torna-se a chave primária da relação.
- ◆ **Conjunto de entidades fracas.** A chave primária da relação consiste da união da chave primária da entidade forte relacionada e o discriminador da entidade fraca.

# Determinando Chaves a partir dos Conjuntos E-R

- ◆ **Conjunto de relacionamentos.** A união da chave primária das entidades relacionadas torna-se uma super chave da relação. Para relacionamentos binários do tipo **muitos-para-muitos**, essa super chave também é a chave primária. Para relacionamentos binários do tipo **muitos-para-um**, a chave primária da entidade “muitos” torna-se a chave primária da relação. Para relacionamentos binários do tipo **um-para-um**, pode ser a chave primária de qualquer das duas entidades.

# Linguagens de Consulta

- ◆ Linguagem por meio da qual usuários solicitam informações do banco de dados.
- ◆ Categoria de linguagens:
  - **Procedural**
  - **Não-procedural**
- ◆ Linguagens “Puras” :
  - Álgebra Relacional (procedural)
  - Cálculo relacional de tupla (não-procedural)
  - Cálculo relacional de domínio (não-procedural)
- ◆ Linguagens puras (formais, sem a sintaxe agradável das linguagens comerciais) formam a base subjacente das linguagens de consultas usadas comercialmente.

# Modelo Relacional

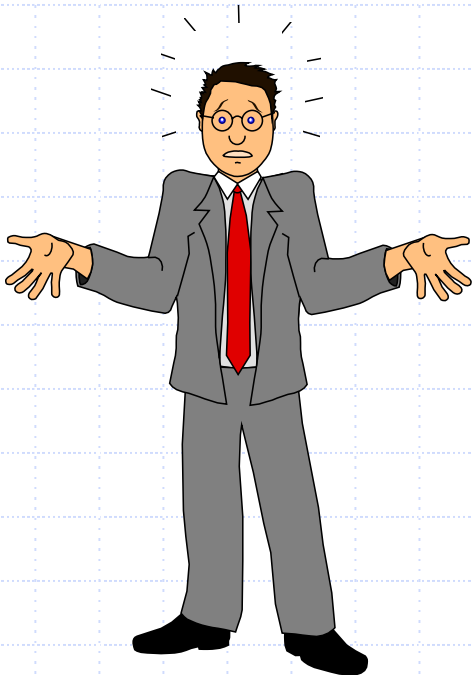
- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Álgebra Relacional

- ◆ Linguagem procedural
- ◆ Seis operadores básicos
  - seleção (  $\sigma$  )
  - projeção (  $\rightarrow$  )
  - união (  $\cup$  )
  - diferença (  $-$  )
  - Produto cartesiano (  $\times$  )
  - Rename (  $\uparrow$  )
- ◆ Os operadores tomam **uma ou mais relações** como **entrada** e produzem uma **nova relação** como **resultado**.

# Observações Gerais

- ◆ O que são operações primárias ?  
O que são operações binárias?





# Observações Gerais

◆ O que são operações primárias ?

São aquelas operações que são efetuadas em cima de uma única relação (ou seja única tabela).

Quais as operações que vocês consideram como primárias ?

- Seleção, projeção e rename

# Observações Gerais

O que são operações binárias?

As operações que operam sobre um par de relações (tabelas) são ditas operações binárias.

Quais operações podemos considerar como binárias ?

- União, Diferença e Produto cartesiano

# Observações Gerais

Os operadores tomam **uma ou mais relações** como **entrada** e produzem uma **nova relação** como **resultado**.

Está certa esta afirmação ?

Os operadores tomam **uma ou mais tabela** como **entrada** e produzem uma **nova tabela** como **resultado**.

# Operação de Seleção

- ◆ Notação:  $\sigma_P(r)$
- ◆ Definida como:  $\sigma_P(r) = \{t \mid t \in r \text{ and } P(t)\}$
- ◆ Onde P é uma fórmula do cálculo proposicional, tratando termos da seguinte forma:

<atributo> = < atributo > ou <constante>

$\neq$

$>$

$\geq$

$<$

$\leq$

"conectados por":  $\wedge$  (**and**),  $\vee$  (**or**),  $\neg$  (**not**)

# Observações Gerais

A operação de *seleção* é representada pela letra grega  $\downarrow$ , assim temos :

$\downarrow_{\text{nome\_agência} = \text{UFSC}} (\text{empréstimo})$

Onde :

- Predicado :  $\text{nome\_agência} = \text{UFSC}$
- Argumento da Relação :  $\text{empréstimo}$

# Operação de Seleção – Exemplo

◆ Relação r:

A	B	C	D
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	7
$\beta$	$\beta$	12	3
$\beta$	$\beta$	23	10

◆  $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

# Operação de Projeção

◆ Notação:

$$\Pi_{A_1, A_2, \dots, A_k}(r)$$

onde  $A_1, A_2$  são nomes de atributos e  $r$  um nome de relação.

- ◆ O resultado é definido como a relação de  $k$  colunas obtida pela remoção das colunas que não estão listadas
- ◆ Linhas duplicadas são eliminadas do resultado, visto que relações são conjuntos.

# Operação de Projeção - Exemplo

◆ Relação r:

A	B	C
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

◆  $\Pi_{A, C}(r)$

A	C
$\alpha$	1
<del><math>\alpha</math></del>	<del>1</del>
$\beta$	1
$\beta$	2

=

A	C
$\alpha$	1
$\beta$	1
$\beta$	2



# Operação União

◆ Notação:  $r \cup s$

◆ Definida como:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

◆ Para  $r \cup s$  ser válida,

1. **r, s** devem ter o mesmo grau (aridade - mesmo número de atributos).
2. Os domínios dos atributos devem ser compatíveis (ex.: a segunda coluna de *r* lida com o mesmo tipo de valores da segunda coluna de *s*).

# Operação União - Exemplo

◆ Relações  $r$ ,  $s$ :

$r$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$s$

A	B
$\alpha$	2
$\beta$	3

◆  $r \cup s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3

# Operação Diferença

◆ Notação:  $r - s$

◆ Definida como:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

◆ A operação de diferença só pode ser realizada entre relações compatíveis.

- $r$  e  $s$  devem ter o mesmo grau.
- Os domínios dos atributos de  $r$  e  $s$  devem ser compatíveis

# Operação Diferença – Exemplo

◆ Relações  $r$ ,  $s$ :

$r$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$s$

A	B
$\alpha$	2
$\beta$	3

◆  $r - s$

A	B
$\alpha$	1
$\beta$	1

# Operação Produto Cartesiano

◆ Notação:  $r \times s$

◆ Definida como:

$$r \times s = \{t \ q \mid t \in r \textbf{ and } q \in s\}$$

◆ Assuma que os atributos de  $r(R)$  e  $s(S)$  são disjuntos. (Isto é,  $R \cap S = \emptyset$ ).

◆ Se os atributos de  $r(R)$  e  $s(S)$  não são disjuntos, então uma renomeação deve ser feita.

# Operação Produto Cartesiano - Exemplo

◆ Relações  $r, s$ :

$r$

A	B
$\alpha$	1
$\beta$	2

$s$

C	D	E
$\alpha$	10	+
$\beta$	10	+
$\beta$	20	-
$\gamma$	10	-

◆  $r \times s$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	+
$\alpha$	1	$\beta$	10	+
$\alpha$	1	$\beta$	20	-
$\alpha$	1	$\gamma$	10	-
$\beta$	2	$\alpha$	10	+
$\beta$	2	$\beta$	10	+
$\beta$	2	$\beta$	20	-
$\beta$	2	$\gamma$	10	-

# Composição de Operações

- ◆ Pode-se construir expressões usando múltiplas operações
- ◆ Exemplo :  $\sigma_{A=C}(r \times s)$
- ◆  $r \times s$ 
  - Notação:  $r \bowtie s$
  - Sejam  $r$  e  $s$  relações sobre esquemas  $R$  e  $S$ , respectivamente. O resultado é uma relação com esquema  $R \cup S$  o qual é obtido considerando cada par de tuplas  $t_r$  de  $r$  e  $t_s$  de  $s$ .
  - Se  $t_r$  e  $t_s$  têm os mesmos valores em cada um dos atributos comuns (i.e.,  $R \cap S$ ), então a tupla  $t$  é adicionada ao resultado, onde
    - ◆  $t$  tem o mesmo valor como  $t_r$  em  $r$
    - ◆  $t$  tem o mesmo valor como  $t_s$  em  $s$

# Composição de Operações

Exemplo:

$$R = (A, B, C, D)$$

$$S = (E, B, D)$$

◆ Esquema resultado = (A, B, C, D, E)

◆  $r \bowtie s$  é definido como:

◆  $\pi_{r.A, r.B, r.C, r.D, s.E} \left( \sigma_{r.B = s.B \wedge r.D = s.D} (r \times s) \right)$



# Operação de Junção Natural – Exemplo

◆ Relações r,s:

*r*

A	B	C	D
$\alpha$	1	$\alpha$	a
$\beta$	2	$\gamma$	a
$\gamma$	4	$\beta$	b
$\alpha$	1	$\gamma$	a
$\delta$	2	$\beta$	b

*s*

B	D	E
1	a	$\alpha$
3	a	$\beta$
1	a	$\gamma$
2	b	$\delta$
3	b	$\epsilon$

◆  $r \bowtie s$

A	B	C	D	E
$\alpha$	1	$\alpha$	a	$\alpha$
$\alpha$	1	$\alpha$	a	$\gamma$
$\alpha$	1	$\gamma$	a	$\alpha$
$\alpha$	1	$\gamma$	a	$\gamma$
$\delta$	2	$\beta$	b	$\delta$

# Operação de Divisão

$$r \div s$$

- ◆ é interessante para consultas que incluem a frase “para todos”.
- ◆ Sejam  $r$  e  $s$  relações com esquemas  $R$  e  $S$  respectivamente, onde
  - $R = (A_1, \dots, A_m, B_1, \dots, B_n)$
  - $S = (B_1, \dots, B_n)$

O resultado de  $r \div s$  é uma relação com esquema  $R - S = (A_1, \dots, A_m)$ , tal que

$$r \div s = \{t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r)\}$$

# Operação de Divisão - Exemplo

◆ Relações  $r, s$ :

$r$		$s$
A	B	B
$\alpha$	1	1
$\alpha$	2	2
$\alpha$	3	
$\beta$	1	
$\gamma$	1	
$\delta$	1	
$\delta$	3	
$\delta$	4	
$\delta$	6	
$\varepsilon$	1	
$\varepsilon$	2	

◆  $r \div s$

A
$\alpha$
$\varepsilon$

# Outro Exemplo de Divisão

◆ Relações  $r, s$ :

$r$				
A	B	C	D	E
$\alpha$	a	$\alpha$	a	1
$\alpha$	a	$\gamma$	a	1
$\alpha$	a	$\gamma$	b	1
$\beta$	a	$\gamma$	a	1
$\beta$	a	$\gamma$	b	3
$\gamma$	a	$\gamma$	a	1
$\gamma$	a	$\gamma$	b	1
$\gamma$	a	$\beta$	b	1

$s$	
D	E
a	1
b	1

◆  $r \div s$

A	B	C
$\alpha$	a	$\gamma$
$\gamma$	a	$\gamma$

# Operação de Designação

- ◆ A operação de designação ( $\leftarrow$ ) provê uma maneira conveniente de expressar consultas complexas; escrever uma consulta como um programa seqüencial consiste de uma série de atribuições seguidas por uma expressão cujo valor é apresentado como o resultado da consulta.

# Operação de Designação

◆ A designação deve sempre ser feita a uma variável de relação temporária.

◆ Exemplo: Escrever  $\mathbf{r} \div \mathbf{s}$  como

$$\mathbf{temp1} \leftarrow \Pi_{\mathbf{R-S}}(\mathbf{r})$$

$$\mathbf{temp2} \leftarrow \Pi_{\mathbf{R-S}}((\mathbf{temp1} \times \mathbf{s}) - \Pi_{\mathbf{R-S,S}}(\mathbf{r}))$$

$$\mathbf{result} = \mathbf{temp1} - \mathbf{temp2}$$

- O resultado da expressão a direita de  $\leftarrow$  é atribuído à variável de relação à esquerda de  $\leftarrow$ .
- Pode-se usar variáveis em expressões subsequentes.

# Exemplo de Consultas

◆ Encontrar todos os clientes que tenham ao menos uma conta nas agências "Downtown" e "Uptown".

## ■ Consulta 1

$$\Pi_{\text{nome\_cliente}}(\sigma_{\text{nome\_agência} = \text{"Downtown"}}(\text{depositante} \bowtie \text{conta})) \cap \Pi_{\text{nome\_cliente}}(\sigma_{\text{nome\_agência} = \text{"Uptown"}}(\text{depositante} \bowtie \text{conta}))$$

# Exemplo de Consultas

- ◆ Achar todos os clientes que tem uma conta em todas as agências localizadas no Brooklyn.



$\Pi_{\text{cliente\_nome, agência\_nome}}$  (depositante  
conta)

$\div \Pi_{\text{agência\_nome}}$  ( $\sigma_{\text{cidade\_agência} =$   
"Brooklyn") (agência))



# Modelo Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Cálculo Relacional de Tupla

- ◆ O Cálculo Relacional de Tupla é uma linguagem não-procedural, onde cada consulta é da forma
$$\{t \mid P(t)\}$$
- ◆ Este é o conjunto de todas as tuplas  $t$  que tornam o predicado  $P$  verdadeiro.
- ◆  $t$  é uma variável de tupla;  $t[A]$  denota o valor da tupla  $t$  no atributo  $A$ .
- ◆  $t \in r$  denota que a tupla  $t$  está na relação  $r$ .
- ◆  $P$  é uma fórmula similar àquelas do cálculo de predicados.

# Fórmulas do Cálculo de Predicado

Incluem:

1. Conjunto de atributos e constantes.
2. Conjunto de operadores comparação: (e.g.,  $<$ ,  $\leq$ ,  $=$ ,  $\neq$ ,  $>$ ,  $\geq$ )
3. Conjunto de conectivos lógicos: and ( $\wedge$ ), or ( $\vee$ ), not ( $\neg$ )
4. Implicação ( $\Rightarrow$ ):  $x \Rightarrow y$ , se  $x$  é verdade, então  $y$  é verdade.

$$x \Rightarrow y \equiv \neg x \vee y$$

5. Conjunto de quantificadores:
  - $\exists t \in r (Q(t))$  "existe" uma tupla  $t$  na relação  $r$  que torna o predicado  $Q(t)$  verdadeiro.

# O Exemplo da Empresa Bancária

**agência** (nome\_agência, cidade\_agência, fundos)

**cliente** (nome\_cliente, rua\_cliente, cidade\_cliente)

**conta** (nome\_agência, número\_conta, saldo)

**empréstimo** (nome\_agência, número\_empréstimo, total)

**depositante** (nome\_cliente, número\_conta)

**devedor** (nome\_cliente, número\_empréstimo)

# Exemplo de Consultas

(1) Achar os nomes de todas as agências na relação **emprestimo**

```
select nome_agencia  
from emprestimo
```

# Exemplo de Consultas

(2) Achar os nomes de todas as agencias na relação emprestimo e remover as duplicatas

```
select distinct nome_agencia  
from emprestimo
```

# Exemplo de Consultas

(3) Achar todos os numeros de emprestimos de emprestimos feitos na agencia Perryridge com totais maiores que \$1200.

```
select numero_empestimo  
from emprestimo  
where nome_agencia = 'Perryridge'  
and total > 1200
```

# Exemplo de Consultas

(4) Achar o numero do emprestimo dos emprestimos com total entre \$90,000 e \$100,000 (isto e,  $\geq$  \$90,000 and  $\leq$  \$100,000)

```
select numero_emprestimo  
       from emprestimo
```

```
where total between 90000 and  
100000
```



# Exemplo de Consultas

(5) Encontre o nome do cliente e o numero de emprestimo de todos os clientes que possuem um emprestimo na agencia Perryridge.

```
select distinct  
nome_cliente,devedor.numero_emprest  
imo  
from devedor,emprestimo  
where devedor.numero_emprestimo =  
emprestimo.numero_emprestimo  
and nome_agencia = 'Perryridge'
```

# Exemplo de Consultas

(6) Encontre o nome e o número do empréstimo dos clientes que possuem um empréstimo na agência Perryridge; substitua o nome da coluna número\_empréstimo por número\_do\_empréstimo\_do\_devedor'.

# Exemplo de Consultas

```
select distinct nome_cliente,  
devedor.numero_emprestimo as  
  
    numero_do_emprestimo_do_dev  
edor  
from    devedor, emprestimo  
where  devedor.numero_emprestimo  
= emprestimo.numero_emprestimo  
and    nome_agencia = 'Perryridge'
```

# Exemplo de Consultas

(7) Encontre o nome dos clientes e seus numeros de emprestimo para todos os clientes que possuem um emprestimo em alguma agencia.

# Exemplo de Consultas

```
select distinct nome_cliente,  
                T.numero_emprestimo  
from          devedor as T, emprestimo as S  
where         T.numero_emprestimo  
              =S.numero_emprestimo
```

# Exemplo de Consultas

(8) Encontre o nome de todas as agencias que possuam fundos maiores que ao menos uma agencia daquelas localizadas no Brooklyn.

```
select distinct T.nome_agencia  
from agencia as T, agencia as S  
where T.fundos > S.fundos and  
S.cidade_agencia = 'Brooklyn'
```

# Exemplo de Consultas

(9) Listar em ordem alfabética os nomes de todos os clientes que tem um empréstimo na agência Perryridge:

```
select distinct nome_cliente  
from devedor, emprestimo  
where devedor.numero_emprestimo =  
        emprestimo.numero_emprestimo  
and nome_agencia = 'Perryridge'  
order by nome_cliente
```

# Exemplo de Consultas

(10) Encontre todos os clientes que possuam um empréstimo, uma conta ou ambos:

```
(select nome_cliente from depositante )  
union (select nome_cliente from devedor)
```



# Exemplo de Consultas

(11) Encontre todos os clientes que possuem ambos uma conta e um empréstimo:

```
(select nome_cliente from depositante )  
intersect (select nome_cliente from devedor )
```

# Exemplo de Consultas

(12) Encontre todos os clientes que possuem uma conta mas não possuem emprestimo;

```
(select nome_cliente from depositante)  
except      (select nome_cliente from  
devedor )
```

# Exemplo de Consultas

(13a) Encontre a média dos saldos em contas na agencia Perryridge.

```
select avg (saldo)  
from contas  
where nome_agencia = 'Perryridge'
```

# Exemplo de Consultas

(13b) Encontre o número de tuplas na relação clientes:

```
select count (*)  
from cliente
```

# Exemplo de Consultas

(14) Encontre o número de depositantes no banco:

```
select count (distinct nome_cliente)  
from depositante
```

# Exemplo de Consultas

(15) Encontre o número de depositantes em cada agência.

```
select nome_agencia, count (distinct
nome_cliente )
from depositante,conta
where depositante.numero_conta
=conta.numero_conta
group by nome_agencia
```

# Exemplo de Consultas

(16) Encontre o nome de todas as agencias onde a media do saldo das contas seja maior que \$1,200

```
select nome_agencia, avg (saldo)
from conta
group by nome_agencia
having avg (saldo) > 1200
```

# Exemplo de Consultas

(17) Encontrar todos os clientes que possuem uma conta e um empréstimo no banco.

```
select distinct nome_cliente  
from devedor  
where nome_cliente in (select nome_cliente  
                        from depositante)
```



# Exemplo de Consultas

(18) Encontre a media do balanço de contas das agencias onde a media do balanço de contas e maior que \$1200.

```
select nome_agencia, saldo_medio
  from (select nome_agencia, avg (saldo)
        from conta
       group by nome_agencia)
      as result (nome_agencia,saldo_medio)
 where saldo_medio > 1200
```

# Exemplo de Consultas

(19) Exclua todas os registros de contas da agencia Perryridge

```
delete from conta  
where nome_agencia = 'Perryridge'
```

# Exemplo de Consultas

(20) Apague os registros de todas as contas com saldos abaixo da media no banco

```
delete from conta  
where saldo < (select avg (saldo)  
                from conta)
```

# Exemplo de Consultas

(21) Adicionar uma nova tupla em conta

```
insert into conta  
values ('Perryridge', A-9732, 1200)
```

ou de forma equivalente

```
insert into conta (nome_agencia, saldo,  
numero_conta)  
values ('Perryridge', 1200, A-9732)
```

# Exemplo de Consultas

(22) Forneça aos clientes da agencia Perryridge uma caderneta de poupança de \$200 como brinde para cada emprestimo que eles tenham. O numero do emprestimo será usado como numero da caderneta de poupança

# Exemplo de Consultas

```
insert into conta  
    select nome_agencia,  
numero_emprestimo,200  
    from emprestimo  
    where nome_agencia = 'Perryridge'
```

```
insert into depositante  
    select nome_cliente,  
numero_emprestimo  
    from emprestimo, devedor  
    where nome_agencia = 'Perryridge'  
    and emprestimo.numero_conta =  
devedor.numero_conta
```

# Exemplo de Consultas

- ◆ Encontrar o **nome\_agência**, **número\_empréstimo**, e **total** para empréstimos acima de \$1200:

$$\{t \mid t \in \text{empréstimo} \wedge t[\text{total}] > 1200 \}$$

- ◆ Encontrar o **número\_empréstimo** para cada empréstimo com valor acima de \$1200:

$$\{t \mid \exists s \in \text{empréstimo} \\ (t[\text{número\_empréstimo}] = \\ s[\text{número\_empréstimo}] \wedge s[\text{total}] > 1200)\}$$

Note que uma relação no esquema [empréstimo] é definida implicitamente pela consulta.

# Exemplo de Consultas

- ◆ Encontrar os nomes de todos os clientes que têm um empréstimo, uma conta, ou ambos no banco:

$$\{t \mid \exists s \in \text{devedor} (t[\text{cliente\_nome}] = s[\text{cliente\_nome}]) \\ \vee \exists u \in \text{depositante} (t[\text{cliente\_nome}] = u[\text{cliente\_nome}])\}$$

- ◆ Encontrar os nomes de todos os clientes que têm um empréstimo e uma conta no banco:

$$\{t \mid \exists s \in \text{devedor} (t[\text{cliente\_nome}] = s[\text{cliente\_nome}]) \\ \wedge \exists u \in \text{depositante} (t[\text{cliente\_nome}] = u[\text{cliente\_nome}])\}$$



# Exemplo de Consultas

- ◆ Encontrar o nome de todos os clientes que têm um empréstimo na agência Perryridge:

$$\{t \mid \exists s \in \text{devedor} (t[\text{nome\_cliente}] = s[\text{nome\_cliente}] \wedge \exists u \in \text{empréstimo} (u[\text{nome\_agência}] = \text{"Perryridge"} \wedge u[\text{número\_empréstimo}] = s[\text{número\_empréstimo}])))\}$$

- ◆ Encontrar o nome de todos os clientes que têm um empréstimo na agência Perryridge, mas não tem conta em nenhuma agência do banco:

$$\{t \mid \exists s \in \text{devedor} (t[\text{nome\_cliente}] = s[\text{nome\_cliente}] \wedge \exists u \in \text{empréstimo} (u[\text{nome\_agência}] = \text{"Perryridge"} \wedge u[\text{número\_empréstimo}] = s[\text{número\_empréstimo}]))) \wedge \neg(\exists v) \in \text{depositante} (v[\text{nome\_cliente}] = t[\text{nome\_cliente}])\}$$

# Exemplo de Consultas

- ◆ Encontrar os nomes de todos os clientes que têm empréstimos na agência de Perryridge e nas cidades em que eles vivem:  
 $\{t \mid \exists s \in \text{empréstimo} (s[\text{nome\_agência}] = \text{"Perryridge"} \wedge \exists u \in \text{devedor} (u[\text{número\_empréstimo}] = s[\text{número\_empréstimo}] \wedge t[\text{nome\_cliente}] = u[\text{nome\_cliente}] \wedge \exists v \in \text{cliente} (u[\text{nome\_cliente}] = v[\text{nome\_cliente}] \wedge t[\text{cidade\_cliente}] = v[\text{cidade\_cliente}])))\})$

# Exemplo de Consultas

- ◆ Encontrar os nomes de todos os clientes que têm uma conta em todas as agências localizadas no Brooklyn:

$$\{t \mid \forall s \in \text{agência} (s[\text{cidade\_agência}] = \text{"Brooklyn"}) \\ \exists u \in \text{conta} \\ (s[\text{nome\_agência}] = u[\text{nome\_agência}] \\ \wedge \exists s \in \text{depositante} \\ (t[\text{nome\_cliente}] = s[\text{nome\_cliente}] \\ \wedge s[\text{número\_conta}] = u[\text{número\_conta}])))\}$$

# Expressões de Segurança

- ◆ É possível escrever expressões do cálculo de tuplas que gerem relações infinitas.
- ◆ Por exemplo,  $\{t \mid \neg t \in r\}$  resulta em uma relação infinita se o domínio de qualquer atributo da relação  $r$  for infinito.
- ◆ Este problema é tratado restringindo o conjunto de expressões possíveis à *expressões seguras*.
- ◆ Uma expressão  $\{t \mid P(t)\}$  do cálculo relacional de tuplas é segura se toda componente de  $t$  aparece em uma das relações, tuplas, ou constantes que aparecem em  $P$ .

# Expressividade das Linguagens

- ◆ O cálculo relacional de tuplas limitado por expressões de segurança é equivalente em poder de expressividade à álgebra relacional.
- ◆ Toda expressão do cálculo relacional possui expressão semelhante na álgebra relacional, e vice-versa.

# Modelo Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Cálculo Relacional de Domínio

- ◆ O cálculo relacional de domínio é uma linguagem de consultas não-procedural equivalente ao cálculo relacional de tupla.
- ◆ Essa forma usa variáveis de domínio que tomam valores do domínio de um atributo, ao invés de valores da tupla inteira.
- ◆ Cada consulta é uma expressão da forma:

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

onde

- $x_1, x_2, \dots, x_n$  representam variáveis de domínio
- $P$  representa uma fórmula similar àquelas do cálculo de predicados

# Exemplo de Consultas

- ◆ Achar o **nome\_agência**, **número\_empréstimo** e **total** para empréstimos acima de \$1200:  
$$\{ \langle b, l, a \rangle \mid \langle b, l, a \rangle \in \text{empréstimo} \wedge a > 1200 \}$$
- ◆ Achar os nomes de todos os **clientes** que possuem empréstimos acima de \$1200:  
$$\{ \langle c \rangle \mid \exists b, l, a (\langle c, l \rangle \in \text{devedor} \wedge \langle b, l, a \rangle \in \text{empréstimo} \wedge a > 1200) \}$$
- ◆ Achar os nomes de todos os clientes e o total do empréstimo dos que possuem um empréstimo na agência Perryridge:  
$$\{ \langle c, a \rangle \mid \exists l (\langle c, l \rangle \in \text{devedor} \wedge \exists b (\langle b, l, a \rangle \in \text{empréstimo} \wedge b = \text{"Perryridge"})) \}$$



# Exemplo de Consultas

- ◆ Achar os nomes de todos os clientes que têm um empréstimo, uma conta, ou ambos na agência Perryridge:

$$\{ \langle c \rangle \mid \exists l (\langle c, l \rangle \in \text{devedor} \\ \wedge \exists b, a (\langle b, l, a \rangle \in \text{empréstimo} \wedge b = \text{"Perryridge"})) \\ \vee \exists a (\langle c, a \rangle \in \text{depositante} \\ \wedge \exists b, n (\langle b, a, n \rangle \in \text{conta} \wedge b = \text{"Perryridge"})) \}$$

- ◆ Achar os nomes de todos os clientes que possuem conta em todas as agências localizadas no Brooklyn:

$$\{ \langle c \rangle \mid \forall x, y, z (\langle x, y, z \rangle \in \text{agência} \wedge y = \text{"Brooklyn"}) \\ \exists a, b (\langle x, a, b \rangle \in \text{conta} \wedge \langle c, a \rangle \in \text{depositante}) \}$$

# Expressões de Segurança

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

É segura se todas as seguintes propriedades são válidas:

1. Todos os valores que aparecem nas tuplas das expressões são valores de  $\text{dom}(P)$  (isto é, os valores aparecem ou em  $P$  ou em uma tupla de uma relação mencionada em  $P$ ).
2. Para toda sub-fórmula "existe" da forma  $\exists x(P_1(x))$ , a sub-fórmula é verdadeira se e somente se existe um valor  $x$  em  $\text{dom}(P_1)$  tal que  $P_1(x)$  é verdadeiro.
3. Para toda sub-fórmula "para todo" da forma  $\forall x(P_1(x))$ , a sub-fórmula é verdadeira se e somente se  $P_1(x)$  é verdadeiro para todos os valores  $x$  de  $\text{dom}(P_1)$ .

# Expressividade das Linguagens

São equivalentes:

- ◆ Álgebra relacional
- ◆ Cálculo relacional de tuplas restrito por expressões de segurança
- ◆ Cálculo relacional de domínio restrito por expressões de Segurança

# Modelo

## Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Operações da Álgebra relacional Estendida

Serão vistas:

- ◆ Projeção generalizada
- ◆ Junção externa (Outer Join)
- ◆ Funções Agregadas

# Projeção Generalizada

- ◆ Estende a operação de projeção para permitir que funções aritméticas sejam usadas em listas de projeções.

$$\Pi_{F_1, F_2, \dots, F_n}(E)$$

- ◆ E é uma expressão da álgebra relacional.
- ◆ Os  $F_1, F_2, \dots, F_n$  são expressões aritméticas envolvendo constantes e atributos no esquema de E.
- ◆ Dada uma relação `info_crédito(nome_cliente, limite, saldo_crédito)`, achar o quanto cada pessoa ainda pode gastar:

$$\Pi_{\text{nome\_cliente}, (\text{limite} - \text{saldo\_crédito})}(\text{info\_crédito})$$

# Junção Externa

- ◆ Uma extensão da operação de junção que evita perda de informações.
- ◆ Calcula-se a junção e então adiciona-se ao resultado da junção as tuplas de uma relação que não combinam (*match*) com as tuplas da outra relação.
- ◆ Uso de valores nulos:
  - Nulo significa que o valor é desconhecido ou não existe.
  - Todas as comparações envolvendo valores nulos são **falsas** por definição.

# Exemplo de Junção Externa

## ◆ Relação empréstimo

nome_agência	número_emprestimo	total
Downtown	L-170	3000
Redwood	L-230	4000
Perryridge	L-260	1700

## ◆ Relação devedor

nome_cliente	numero_empréstimo
Jones	L-170
Smith	L-230
Hayes	L-155



# Exemplo de Junção Externa

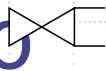
◆ empréstimo ⋈ devedor

nome_agência	número_agência	total	nome_cliente
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith

◆ empréstimo ⋈ devedor

nome_agência	número_empréstimo	total	nome_cliente	número_empréstimo
Downtown	L-170	3000	Jones	L-170
Redwood	L-230	4000	Smith	L-230
Perryridge	L-260	1700	nulo	nulo

# Exemplo de Junção Externa

◆ empréstimo  devedor

nome_agência	número_empréstimo	total	nome_cliente
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
null	L-155	null	Hayes

◆ empréstimo  devedor

nome_agência	número_empréstimo	total	nome_cliente
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
Perryridge	L-260	1700	null
null	L-155	null	Hayes

# Funções agregadas

- ◆ A operação de agregação  $G$  tem como entrada uma coleção de valores e retorna um único valor como resultado.

**avg:** média dos valores

**min:** valor mínimo

**max:** valor máximo

**sum:** soma dos valores

**count:** número de valores

Onde  $G_1, G_2, \dots, G_n \quad \mathbf{G} \quad F_1 A_1, F_2 A_2, \dots, F_m A_m \quad (\mathbf{E})$

- $E$  expressão da álgebra relacional
- $G_1, G_2, \dots, G_n$  é uma lista de atributos para agrupar
- $F_i$  é uma função de agregação
- $A_i$  é um nome de atributo

# Exemplo de Funções Agregadas

◆ Relação r:

A	B	C
$\alpha$	$\alpha$	7
$\alpha$	$\beta$	7
$\beta$	$\beta$	3
$\beta$	$\beta$	10

◆  $\text{sum}_C(r)$

sum-C

27

# Exemplo de Funções Agregadas

◆ Relação conta agrupada pelo

nome\_agência

nome_agência	número_conta	saldo
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

◆ nome\_agência

G

nome_agência	soma_saldo
Brighton	1500
Redwood	700

# Modelo

## Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Modificações no Banco de Dados

- ◆ O conteúdo do banco de dados pode ser modificado usando as seguintes operações:
  - Exclusão
  - Inserção
  - Atualização
- ◆ Todas essas operações são expressas usando o operador de designação.

# Exclusão

- ◆ A solicitação de exclusão é expressa de maneira similar a uma consulta. No entanto, ao invés de mostrar as tuplas selecionadas ao usuário, elas são excluídas do banco de dados.
- ◆ Pode-se excluir apenas tuplas inteiras; não é possível excluir valores de atributos específicos.
- ◆ Uma exclusão é expressa na álgebra relacional por:

$$r \leftarrow r - E$$

- ◆ Onde  $r$  é uma relação e  $E$  é uma consulta da álgebra relacional.



# Exemplos de Exclusão

- ◆ Excluir todos os registros de contas na agência Perryridge.

**conta**  $\leftarrow$  **conta** -  $\sigma_{\text{nome\_agência} = \text{"Perryridge"}}$  (**conta**)

- ◆ Excluir todos os registros de empréstimo com total entre 0 e 50.

**empréstimo**  $\leftarrow$  **empréstimo** -  $\sigma_{\text{total} \geq 0 \text{ and total} \leq 50}$  (**empréstimo**)

- ◆ Excluir todas as contas nas agências localizadas em Needham.

$r_1 \leftarrow \sigma_{\text{cidade\_agência} = \text{"Needham"}}$  (**conta**  $\bowtie$  **agência**)

$r_2 \leftarrow \Pi_{\text{nome\_agência, número\_conta, saldo}}$  ( $r_1$ )

$r_3 \leftarrow \Pi_{\text{nome\_cliente, número\_conta}}$  ( $r_2$  **depositante**)

**conta**  $\leftarrow$  **conta** -  $r_2$

**depositante**  $\leftarrow$  **depositante** -  $r_3$

# Inserção

- ◆ Para inserir dados em uma relação, deve-se:
  - Especificar uma tupla a ser inserida, ou
  - Escrever uma consulta cujo resultado é um conjunto de tuplas a ser inserido
- ◆ Em álgebra relacional, uma inserção é expressa por:

$$r \leftarrow r \cup E$$

- ◆ Onde  $r$  é uma relação e  $E$  é uma expressão da álgebra relacional.
- ◆ A inserção de uma única tupla é expressa especificando  $E$  como uma relação constante contendo uma tupla.

# Exemplos de Inserção

- ◆ Inserir informação no banco de dados especificando que o cliente Smith tem \$1200 na conta A-973 na agência Perryridge.

$\text{conta} \leftarrow \text{conta} \cup \{(\text{"Perryridge"}, \text{A-973}, 1200)\}$

$\text{depositante} \leftarrow \text{depositante} \cup \{(\text{"Smith"}, \text{A-973})\}$

- ◆ Incluir, a título de presente para todos os clientes de empréstimos na agência Perryridge, uma conta de poupança de poupança de \$200. Faça o número de empréstimo servir como número de conta para essas novas contas de poupança.

$r_1 \leftarrow (\sigma_{\text{nome\_agência} = \text{"Perryridge"}}(\text{devedor} \quad \text{empréstimo}))$

$\text{conta} \leftarrow \text{conta} \cup \Pi_{\text{nome\_agência}, \text{número\_empréstimo}, 200}(r_1)$

$\text{depositante} \leftarrow \text{depositante} \cup \Pi_{\text{nome\_cliente}, \cancel{\text{número\_empréstimo}}}(r_1)$

# Atualização

- ◆ Um mecanismo para mudar um valor em uma tupla sem mudar todos os valores na tupla
- ◆ Usa-se o operador de projeção generalizada para esta tarefa

$$\mathbf{r} \leftarrow \Pi_{F_1, F_2, \dots, F_n}(\mathbf{r})$$

- Cada  $F_i$  ou é o  $i$ -ésimo atributo de  $r$ , se seu valor não é modificado, ou é uma expressão para o valor do atributo a ser modificado.
- $F_i$  é uma expressão, envolvendo somente constantes e os atributos de  $r$ , os quais dão o novo valor para o atributo.

# Exemplos de Atualização

- ◆ Fazer pagamento de juros aumentando todos os saldos em 5 por cento.

**conta**  $\leftarrow \Pi_{\text{nome\_agência, número\_conta, saldo} \leftarrow \text{saldo} * 1.05}$   
**(conta)**

- ◆ Fazer pagamentos de juros de 6% para contas com saldo acima de \$10.000 e 5% para as outras contas.

**conta**  $\leftarrow \Pi_{\text{nome\_agência, número\_conta, saldo} \leftarrow \text{saldo} * 1.06} (\sigma_{\text{saldo} > 10000}$   
**(conta))**  $\cup \Pi_{\text{nome\_agência, número\_conta, saldo} \leftarrow \text{saldo} * 1.05} (\sigma_{\text{saldo} \leq}$   
**10000 (conta))**

# Modelo

## Relacional

- ◆ Estrutura dos Bancos de Dados Relacionais
- ◆ Álgebra Relacional
- ◆ Cálculo Relacional de Tuplas
- ◆ Cálculo Relacional de Domínio
- ◆ Operações de Álgebra Relacional Estendida
- ◆ Modificações no Banco de Dados
- ◆ Visões

# Visões

- ◆ Em alguns casos, não é desejável que todos os usuários possam ver o modelo lógico inteiro (isto é, as relações armazenadas de fato no banco de dados).

# Visões

- ◆ Considere um pessoa que precise conhecer o número do empréstimo de um cliente mas não precisa ver o total do empréstimo. Esta pessoa deveria ver uma relação descrita, na álgebra relacional, como

$\Pi_{\text{nome\_cliente, número\_empréstimo}}(\text{devedor empréstimo})$  ✕

- ◆ Toda relação que não é parte do modelo conceitual mas é tornada visível para um usuário como uma “relação virtual” é chamada de uma visão.



# Definição de Visões

- ◆ Uma visão é definida usando a declaração **create view** que tem a forma

**create view** v **as** <expressão de consulta>

onde < expressão de consulta > é qualquer expressão correta da álgebra relacional. O nome da visão é representado por v.

- ◆ Uma vez definida a visão, o nome da visão pode ser usado para se referir à relação virtual que a definição da visão gera.
- ◆ Definição de visão não é o mesmo que criar uma nova relação a partir da avaliação da expressão de consulta. Ao invés disso, uma definição de visão causa o armazenamento de uma expressão para ser substituída nas consultas que usam a visão.

# Exemplos de Visões

- ◆ Considere a visão (chamada de **todos\_clientes**) consistindo das agências e seus clientes.

**create view** todos\_clientes **as**

$\Pi_{\text{nome\_agência, nome\_cliente}} (\text{depositante} \bowtie \text{conta})$   
 $\cup \Pi_{\text{nome\_agência, nome\_cliente}} (\text{devedor} \bowtie \text{empréstimo})$

- ◆ Pode-se encontrar todos os clientes da agência Perryridge escrevendo:

$\Pi_{\text{nome\_cliente}} (\sigma_{\text{nome\_agência} = \text{"Perryridge"}} (\text{todos\_clientes}))$

# Atualizações Através de Visões

- ◆ Modificações no banco de dados expressas como visões devem ser traduzidas em modificações de relações reais do banco de dados.
- ◆ Considere a pessoa que precisa ver todos os dados de empréstimo na relação exceto o **total**. A visão, **agência\_empréstimo**, dada à pessoa poderia ser definida como:

# Atualizações Através de Visões

**create view** agência\_empréstimo  
**as**

$\Pi_{\text{nome\_agência, número\_empréstimo}}$  (empréstimo)

Como permite-se que o nome de uma visão apareça em referências a nomes de relações, se poderia escrever :

agência\_empréstimo  $\leftarrow$  agência\_empréstimo  
 $\cup \{("Perryridge", L-37)\}$

# Atualizações Através de Visões

- ◆ A inserção anterior deve ser representada por uma inserção na relação empréstimo do banco de dados a qual é usada a construção da visão **agência\_empréstimo**.
- ◆ Uma inserção na relação **empréstimo** requer o valor do atributo **total**. Uma inserção pode ser tratada das seguintes maneira:
  - Rejeitando a inserção e apresentando uma mensagem de erro para o usuário;
  - Inserindo a tupla ("Perryridge", L-37, nulo) na relação empréstimo.

# Visões Definidas Usando Outras Visões

- ◆ Uma visão pode ser usada na expressão de definição de outra visão.
- ◆ Uma relação de visão  $v_1$  é dita **depende diretamente** de uma relação de visão  $v_2$ , se  $v_2$  é usada na expressão que define  $v_1$ .
- ◆ Uma relação de visão  $v_1$  é dita **depende** de uma relação de visão  $v_2$ , se e somente se existe um caminho de  $v_2$  para  $v_1$  no grafo de dependência.
- ◆ Uma relação de visão  $v$  é dita ser **recursiva** se ela depende dela mesma.

# Expansão de Visões

- ◆ É uma maneira de definir o significado de visões definidas em termos de outras visões.
- ◆ Seja a visão  $v_1$  definida por uma expressão  $e_1$  que pode por sua vez conter usos de relações de visões.

# Expansão de Visões

- ◆ A expansão de visão de uma expressão repete os seguintes passos de substituição:

**repeat**

Encontrar todas as relações de visão  $v_i$  em  $e_1$

Substituir a relação de visão  $v_i$  pela expressão que define  $v_i$

**until** não existirem mais relações de visão em  $e_1$

- ◆ Desde que as definições de visão não são recursivas, esse loop deve terminar.