Capítulo 2: Modelo Relacional

Database System Concepts, 5th Ed.

Capítulo 2: Modelo Relacional

- Estrutura das Bases de Dados Relacionais
- Operações fundamentais da Álgebra Relacional
- Operações adicionais da Álgebra Relacional
- Operações estendidas da Álgebra Relacional
- Valores Nulos
- Modificação da base de dados

Exemplo de uma Relação

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Estrutura básica

- Dados os conjuntos D_1, D_2, …. D_n a **relação** *r* é um subconjunto de
  D_1 x D_2 x … x D_n
  Uma relação é um conjunto de tuplos de aridade n (a_1, a_2, …, a_n) com $a_i \in D_i$
- Exemplo: Se
  - *customer_name* = {Jones, Smith, Curry, Lindsay, …}
    /* Conjunto de todos os nomes de clientes */
  - *customer_street* = {Main, North, Park, …}
    /* Conjunto de todas as ruas de clientes*/
  - *customer_city* = {Harrison, Rye, Pittsfield, …}
    /* Conjunto de todas as cidades de clientes */

Então *r* = { (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield) }

é uma relação sobre *customer_name* x *customer_street* x *customer_city*

Tipos de Atributos

- Cada atributo de uma relação tem um nome
- Ao conjunto dos valores permitidos para cada atributo chama-se **domínio** do atributo
- Os valores dos atributos são normalmente **atómicos**; isto é, indivisíveis
  - E.g. O valor de um atributo pode ser o número de uma conta (account number), mas não pode ser um conjunto de números de conta
- O domínio é atómico se todos os seus membros são atómicos
- O valor especial *null* (nulo) é membro de todos os domínios
- O valor null complica a definição de muitas operações
  - Na apresentação inicial dos operadores vamos ignorar os casos em que o valor é null

Esquema de uma relação

- $A_1, A_2, \ldots, A_n$ *são atributos*
- $R = (A_1, A_2, \ldots, A_n)$ é o esquema da relação

  Exemplo:

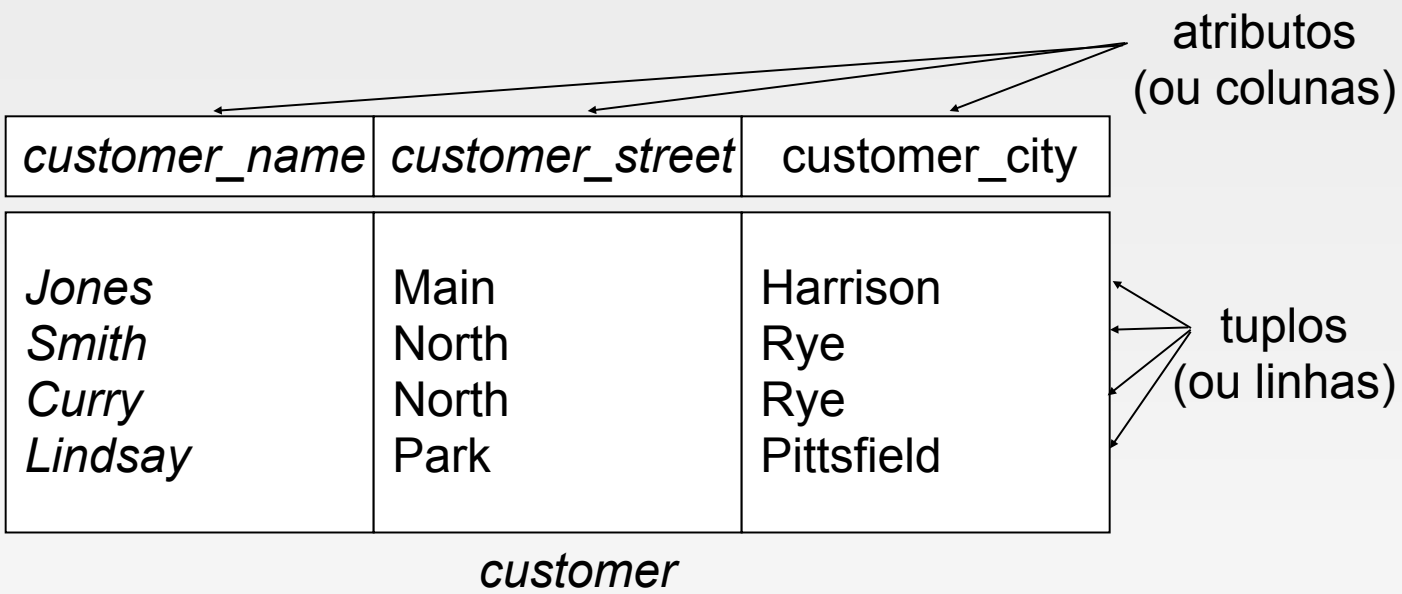
  Customer_schema = (*customer_name, customer_street, customer_city*)
- $r(R)$ denota a relação r no esquema de relação R

  Exemplo:

  customer (Customer_schema)

Instância de uma relação

- Os valores actuais (*instância da relação*) da relação são especificados numa tabela
- Um elemento *t* de *r* é um tuplo, representado numa *linha* da tabela



As relações não têm ordem

- A ordem dos tuplos não é relevante (os tuplos podem ser guardados por uma ordem arbitrária)
- Exemplo: relação conta com os tuplos sem ordem

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

Base de Dados

- Uma base de dados é um conjunto de relações
- A informação de uma empresa é separada em várias partes e cada relação guarda uma parte da informação

  account : (conta) guarda a informação sobre contas
  depositor : (depositante) guarda a informação sobre que cliente tem que conta
  customer : (cliente) guarda a informação sobre clientes

- Guardar toda a informação numa única relação como por exemplo:
  bank(*account_number, balance, customer_name*, ..)
  resulta na:
  - Repetição de informação
    - e.g.,se dois clientes têm a mesma conta (o que é que se repete?)
  - Necessidade de usar valores nulll
    - e.g., para representar um cliente que não tem conta
- No capítulo 7 vamos lidar com o desenho de esquemas relacionais usando a teorias de normalização.

A relação cliente

customer_name	*customer_street*	*customer_city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

A relação depositante

customer_name	*account_number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Chaves

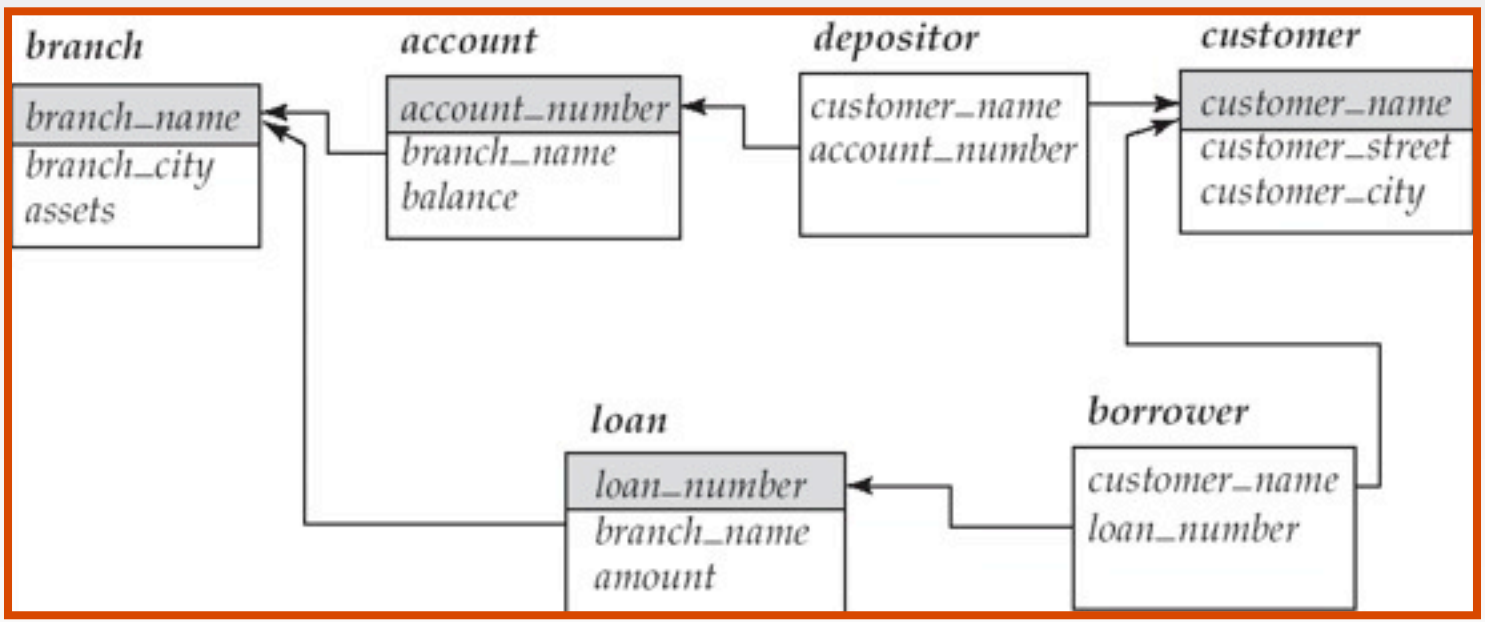
- Seja K ∈ R
- *K é uma* **super chave de** *R* se os valores de K são suficientes para identificar um tuplo único de cada relação possível r(R)
  - Com "relação possível r" queremos dizer a relação r que poderá existir na empresa que estamos a modelar.
  - Exemplo: {*customer_name, customer_street*} e
    {*customer_name*}
    são ambas super chaves de *Customer*, se não é possível que dois clientes tenham o mesmo nome
    - Na vida real, seria necessário usar um atributo *customer_id* em vez de *customer_name para identificar univocamente um cliente*, mas nos exemplos para ficarem mais pequenos assumimos que os nomes dos clientes são únicos.

Chaves (Cont.)

- *K* é uma **chave candidata** **se** *K* é minimal
  Exemplo: {*customer_name*} é uma chave candidata para *Customer*, pois é uma super chave e não tem nenhum subconjunto que seja super chave.
- **Chave primária:** uma chave candidata escolhida como a forma principal para identificar os tuplos de uma relação
  - Deve-se escolher uma atributo que nunca ou raramente muda.
  - E.g. o endereço de email é único, mas pode mudar, o Nº do BI não muda.

Chaves estrangeiras

- O esquema de uma relação pode ter uma atributo que é a chave primária de outra relação. A este atributo nesta relação chama-se **chave estrangeira**.
  - E.g. os atributos *customer_name* e *account_number* de ***depositante*** são chaves estrangeiras de *cliente* e *conta* respectivamente.
  - Só os valores que ocorrem no atributo chave primária da **relação referenciada** podem ocorrer no atributo chave estrangeira da **relação que referencia**.
- **Diagrama do esquema**



Linguagem de interrogação
Query Languages

- Linguagem em que o utilizador pede a informação à base de dados.
- Categorias das Linguagens
  - Procedimental
  - Não-procedimentall, ou declarativa
- Linguagens “Puras”:
  - Álgebra Relacional
  - Tuple relational calculus
  - Domain relational calculus
- As linguagens puras estão na origem das linguagens de interrogação implementadas e usadas nos sistemas de gestão de bases de dados

Algebra Relacional

- Linguagem procedimental
- Seis operadores básicos
  - Selecciona (select): σ
  - Projecta (project): Π
  - União (union): $\cup$
  - Diferença conjuntos (set difference): –
  - Produto cartesiano (Cartesian product): x
  - Renomear (rename): ρ
- Os operadores tem uma ou duas relações como argumentos e retornam uma nova relação como resultado.

Operação seleciona (select) – Exemplo

- Relação r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Operação seleciona (select)

- Notação: $\sigma_p(r)$
- *A p* chama-se **o predicado de selecção**
- Definido como:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \text{ e } p(t)\}$$

onde *p* é uma fórmula do cálculo de proposições que consiste em **termos** ligados com : ∧ (**e**), ∨ (**ou**), ¬ (**negação**)
Cada **termo** é:

<atributo> *op* <atributo> ou <constante>

onde *op* é: =, ≠, >, ≥. <. ≤

- Exemplo de uma selecção:

$$\sigma_{branch\_name=\text{“Perryridge”}}(account)$$

Operação projecção (project) – Exemplo

- *Relação r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Operação projecção (project)

- Notação:

$$\Pi_{A1,A2,...,Aκ}(r)$$

onde A_1, $A_{2,\ ...,}$ são nomes de atributos e r é o nome de uma relação.

- O resultado é uma relação de k colunas que se obtém eliminando as colunas que não foram listadas
- As linhas duplicadas são removidas, uma vez que as relações são conjuntos
- Exemplo: Para eliminar o atributo *branch_name* da relação conta

$$\Pi_{account\_number,\ balance}\ (account)$$

Operação União (union) – Exemplo

- Relações *r, s:*

A	*B*
α	*1*
α	*2*
β	*1*

r

A	*B*
α	*2*
β	*3*

s

- r ∪ s:

A	*B*
α	*1*
α	*2*
β	*1*
β	*3*

Operação União

- Notação: $r \cup s$
- Definição:

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

- Para $r \cup s$ ser válida
  1. *r, s* devem ter a mesma **aridade** (o mesmo número de atributos)
  2. O domínio dos atributos deve ser **compatível** (exemplo: a 2ª coluna de *r* tem os mesmo tipo de valores que a 2ª coluna de *s*)
- Exemplo: para encontrar todos os clientes que têm uma conta ou um empréstimo

  $$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$$

Operação diferença de conjuntos – Exemplo

- Relações *r*, *s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

- *r* – *s*:

A	*B*
α	1
β	1

Operação diferença de conjuntos

- Notação $r - s$
- Definição:

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

- A diferença de conjuntos só pode ser feita entre relações **compatíveis**.
  - r e s devem ter a mesma aridade
  - Os domínios dos atributos de r e s devem ser compatíveis

Operação produto cartesiano – Exemplo

- Relações *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Operação produto cartesiano

- *Notação r* x *s*
- Definição:

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

- Os atributos de r(R) e s(S) são disjuntos. (Isto é, $R \cap S = \varnothing$).
- Se os atributos de *r(R)* e *s(S)* não são disjuntos é necessário usar a operação renomear.

Composição de operações

- Podem-se construir expressões usando vários operadores
- Exemplo: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Operação renomear

- Permite dar um nome aos resultados das expressões da álgebra relacional.
- Permite a referência a uma relação com mais do que um nome
- Exemplo:

$$\rho_{X}(E)$$

  Retorna a expressão *E* com o nome *X*

- Se a expressão da álgebra relacional, E, tem aridade n, então

$$\rho_{\Xi(A1, A2, \ldots, An)}(E)$$

  Retorna o resultado da expressão *E* com o nome *X*, e com o nome dos atributos renomeados $A_1, A_2, \ldots, A_n$.

Exemplo do banco

branch (branch_name, branch_city, assets)

agencia(agencia_nome, agencia_cidade, agencia_negocios)

customer (customer_name, customer_street, customer_city)

cliente(cliente_nome, cliente_rua, cliente_cidade)

account (account_number, branch_name, balance)

conta(conta_numero, agencia_nome, saldo)

loan (loan_number, branch_name, amount)

emprestimo(emprestimo_numero, agencia_nome,valor)

depositor (customer_name, account_number)

depositante(cliente_nome, conta_numero)

borrower (customer_name, loan_number)

credito(cliente_nome, emprestimo_numero)

Exemplos de perguntas

- Encontrar todos os empréstimos maiores de $1200

$$\sigma_{amount > 1200}(loan)$$

- Encontrar os números dos empréstimos maiores de $1200

$$\Pi_{loan\_number}(\sigma_{amount > 1200}(loan))$$

- Encontrar os nomes dos cliente que têm um empréstimo uma conta ou ambos

$$\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)$$

Exemplos de Perguntas

- Encontrar os nomes de todos os clientes que têm um empréstimo na agência Perryridge.

$$\Pi_{customer\_name} (\sigma_{branch\_name=\text{“Perryridge”}}$$

$$(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

- Encontrar os nomes de todos os clientes que têm um empréstimo na agência Perryridge mas não têm nenhuma conta em nenhuma agência do banco.

$$\Pi_{customer\_name} (\sigma_{branch\_name = \text{“Perryridge”}}$$

$$(\sigma_{borrower.loan\_number = loan.loan\_number}(\text{borrower x loan})) -$$

$$\Pi_{customer\_name}(\text{depositor})$$

Exemplo de pergunta

- Encontrar todos os nomes dos clientes que têm um empréstimo na agência Perryridge.
  - Query 1

$$\pi_{customer\_name} (\sigma_{branch\_name = \text{“Perryridge”}} (\sigma_{borrower.loan\_number = loan.loan\_number} (borrower \times loan)))$$

  - Query 2

$$\pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number} ((\sigma_{branch\_name = \text{“Perryridge”}} (loan)) \times borrower))$$

Exemplo de Pergunta

- Encontrar o maior saldo das contas do banco
  - Estratégia:
    - Encontrar os saldos que não são o maior
      - Renomear a relação *account por d para que se possa compara cada conta com as outras*
    - Usar a operação diferença de conjuntos para encontrar as contas que não estão no passo anterior.
  - A query é:

$$\pi_{balance}(account) -$$

$$\pi_{account.balance} (\sigma_{account.balance < d.balance} (account \times \rho_{d} (account)))$$

Definição formal

- Uma expressão básica da álgebra relacional é:
  - Uma relação da base de dados
  - Ou uma relação constante
- Se E_1 e E_2 são expressões da álgebra relacional; as expressões seguintes também são:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P é um predicado sobre atributos de E_1
  - $\pi_s(E_1)$, S é uma lista com alguns atributos de E_1
  - $\rho_x (E_1)$, x é o nome para o resultado de E_1

Operações adicionais

As seguintes operações podem ser definidas para simplificar as perguntas mais comuns:

- Intersecção de conjuntos (Set intersection)
- Junção natural (Natural join)
- Divisão (Division)
- Afectação (Assignment)

Operação interseção de conjuntos

- Notação: $r \cap s$
- Definição:

$r \cap s = \{ t \mid t \in r \textbf{ e } t \in s \}$

- Para que a operação possa ser definida é necessário que:
  - r, s tenham a mesma aridade
  - Os atributos de r e s sejam compatíveis
- Nota: $r \cap s = r - (r - s)$

Operação intersecção de conjuntos – Exemplo

- Relações *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Operação Junção natural

- Notação: r ⋈ s
- Sejam *r* e *s* relações nos esquemas *R* e *S* respectivamente.
  Enyão, r ⋈ s é uma relação no esquema $R \cup S$ obtida da seguinte forma:
  - Considere os pares de tuplos t_r da relação *r* e t_s da relação *s*.
  - Se t_r e t_s tem o mesmo valor cada uma dos atributos de $R \cap S$, adiciona-se o tuplo *t* ao resultado, com
    - *t* tem o mesmo valor de t_r em *r*
    - *t* tem o mesmo valor de t_s em *s*
- Exemplo:

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

S = (*E, B, D*)

- Esquema resultado = (*A, B, C, D, E*)
- *r* ⋈ *s* é definido com:

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

Operação Junção natural – Exemplo

- Relações r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	∈

s

- r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Operação Divisão

- Notação: $r \div s$
- Usa-se para perguntas com a expressão “para todos”.
- Sejam *r* e *s* relações nos esquemas *R* e *S* respectivamente onde
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

O resultado de r ÷ s é uma relação no esquema

$R - S = (A_1, \ldots, A_m)$

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

tu é a concatenação dos tuplos *t* e *u* para obter um unico tuplo

Operação Divisão – Exemplo

- Relações: *r, s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
∈	6
∈	1
β	2

r

B
1
2

s

- *r* ÷ *s*:

A
α
β

Operação Divisão – Exemplo

- Relações *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

- *r* ÷ *s*:

A	*B*	*C*
α	a	γ
γ	a	γ

Operação Divisão (Cont.)

- Propriedade
  - Seja $q = r \div s$
  - Então q é a maior relação que satisfaz $q \times s \subseteq r$
- Definição usando os operadores básicos
  Sejam $r(R)$ e $s(S)$ relações, e seja $S \subseteq R$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

  - $\Pi_{R-S,S}(r)$ reordena os atributos de r
  - $\Pi_{R-S}(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ tem os tuplos t pertencentes a $\Pi_{R-S}(r)$ tal que para algum tuplo $u \in s$, $tu \notin r$.

Operação Afectação

- A operação afectação (←) permite representar as perguntas mais complexas de forma conveniente
  - Escrever uma pergunta com
    - Uma série de afectações
    - Seguida de uma expressão cujo valor é apresentado como resultado da pergunta.
  - A Afectação é sempre feita para uma variável de relação temporária.
- Exemplo: Escrever $r \div s$ como

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

  - O resultado à direita de ← é atribuido à variável de relação do lado esquerdo de ←.
  - Pode-se usar a variável em expressões subsequentes.

Exemplos de Perguntas sobre o Banco

- Encontrar os nomes de todos os cliente que têm um empréstimo e uma conta no banco

$$\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$$

- Encontrar o nome de todos os clientes que têm um empréstimo e o valor do empréstimo

$$\Pi_{customer\_name,\ loan\_number,\ amount}(borrower \bowtie loan)$$

Exemplos de Perguntas sobre o Banco

- Encontrar todos os clientes que têm pelo menos uma conta nas agências de Downtown e Uptwon.
  - pergunta 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Downtown”}}(depositor \bowtie account)) \cap \Pi_{customer\_name}(\sigma_{branch\_name = \text{“Uptown”}}(depositor \bowtie account))$$

  - pergunta 2

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \rho_{temp(branch\_name)}(\{(\text{“Downtown”}), (\text{“Uptown”})\})$$

Notem que a pergunta 2 usa uma relação constante

Exemplos de Perguntas sobre o Banco

- Encontrar todos os clientes que têm uma conta em todas as agências na cidade de Brooklyn.

$$\Pi_{customer\_name,\ branch\_name}\ (depositor \bowtie account)$$

$$\div\ \Pi_{branch\_name}\ (\sigma_{branch\_city = \text{“Brooklyn”}}\ (branch))$$

Operações da Algebra Relacional estendidas

- Projecção Generalizada (Generalized Projection)
- Funções de Agregação (Aggregate Functions)
- Junção exterior (Outer Join)

Projecção generalizada

- Estende a operação projecção permitindo o uso de funções aritméticas na lista de atributos da projecção.
  - $\Pi_{F1,F2,...,Fn}(E)$
- *E* é uma expressão da álgebra relacional
- $F_1, F_2, \ldots, F_n$ *são expressões aritméticas com constantes e atributos do esquema de E*.
- Dada a relação

credit_info(customer_name, limit, credit_balance),

info_credito(nome_cliente, limite, balanço_credito)

encontrar o valor que cada pessoa pode gastar:

$$\Pi_{customer\_name,\ limit - credit\_balance}\ (credit\_info)$$

Funções de Agregação e operações

- **Funções de agregação** recebem uma colecção de valores e retornam um único valor

  avg: valor médio
  min: valor minimo
  max: valor máximo
  sum: soma dos valores
  count: numero de valores

- **Operação agregação** na álgebra relacional

  - $_{G1,G2,...Gn}\mathcal{g}_{F1(A1),F2(A2),...,Fm(Am)}(E)$

    - *E* é uma expressão da álgebra relacional
    - $G_1, G_2 \ldots, G_n$ é a lista de atributos onde se pretende agrupar (pode ser vazia)
    - F_i *é uma função de agregação*
    - A_i *é o nome de um atributo*

Operação Agregação – Exemplo

- Relação *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

- $g_{\text{sum(c)}}(r)$

sum(*c*)
27

Operação Agregação – Exemplo

- A relação conta (*account)* agrupada por nome_agencia (*branch-name)*:

branch_name	*account_number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$branch\_name\ \mathcal{G}_{\mathbf{sum}(balance)}\ (account)$$

branch_name	**sum**(*balance*)
Perryridge	1300
Brighton	1500
Redwood	700

Funções de Agregação (Cont.)

- O resultado da agregação não tem nome
  - Pode-se dar um nome ao resultado
  - Pode-se renomear o resultado na operação agregação

$$branch\_name\ \mathcal{G}\ \textbf{sum}(balance)\ \textbf{as}\ sum\_balance\ (account)$$

Junção exterior (Outer Join)

- Uma extensão da operação junção para evitar perder informação.
- Calcula a junção e adiciona tuplos de uma relação que não encaixam na outra relação
- Usa valores nulos (*null* values):
  - *Nulo (null) significa que o valor é desconhecido ou não existe*
  - Todas as comparações que envolvem valores nulos retornam Falso por definição

Junção exterior – Exemplo

- Relação *loan (empréstimo)*

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relação *borrower (crédito)*

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

Junção exterior (Outer Join) – Exemplo

- Junção (Join)

  loan $\bowtie$ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- Left Outer Join

  loan $\mathbin{⟕}$ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Junção exterior (Outer Join) – Exemplo

- Right Outer Join

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

- Full Outer Join

  loan ⟗ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

Valores Nulos (Null Values)

- É possivel os tuplos terem valores nulos (null) em alguns dos seus atributos
- *null* significa que o valor é desconhecido ou que não existe
- O resultado de qualquer operação aritmética que envolva valores null é null.
- As funções de agregação ignoram os valores null (no SQL também)
- Para eliminar duplicados e agrupar o null é tratado como outro valor qualquer (no SQL também)

Valores Null

- As comparações com valores nulos retornam o valor: *unknown*
  - Se *false* fosse usado em vez de *unknown*, então *not (A < 5)* não seria equivalente a *A >= 5*
- Lógica a três valores o valor *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: (*true* **and** *unknown*) = *unknown,*
    (*false* **and** *unknown*) = *false,*
    (*unknown* **and** *unknown*) = *unknown*
  - NOT*:* (**not** *unknown*) = *unknown*
  - Em SQL "*P* **is unknown**" **é verdade** (true) se o predicado *P* é *unknown*
- Na selecção (σ) quando o resultado do predicado é *unknown o efeito é igual ao caso em que é false.*

Modificação da base de dados

- O conteúdo da base de dados pode ser alterado usando as seguintes operações:
  - Deletion (remover ou apagar)
  - Insertion (inserir)
  - Updating (actualizar)
- Estas operações são representadas usando o operador afectação.

Remover (Deletion)

- Um pedido de remoção representa-se como uma pergunta, mas o seu resultado é a remoção de um conjunto de tuplos de uma tabela da base de dados.
- Só se podem remover tuplos inteiros, não se podem remover valores de alguns atributos
- A remoção representa-se com a seguinte expressão da álgebra relacional:

  $$r \leftarrow r - E$$

  onde r é uma relação e E é uma expressão da álgebra relacional.

Exemplos de remoções

- Remover todas as contas da agência Perryridge.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Remover todos empréstimos com o valor entre 0 e 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

- Remover todas as contas da agência de Needham.

$$r_1 \leftarrow \sigma_{branch\_city = \text{“Needham”}}(account \bowtie branch)$$

$$r_2 \leftarrow \pi_{account\_number,\ branch\_name,\ balance}(r_1 \bowtie)$$

$$r_3 \leftarrow \pi_{customer\_name,\ account\_number}(r_2 \text{ depositor})$$

$$account \leftarrow account - r_2$$

$$depositor \leftarrow depositor - r_3$$

Inserção

- Para inserir dados numa relação podemos:
  - Especificar o tuplo
  - Ou escrever uma pergunta cujo resultado é o conjunto de tuplos a ser inserido
- Na álgebra relacional, um inserção exprime-se com a expressão:

  $$r \leftarrow r \cup E$$

  Onde r é uma relação e E é uma expressão da álgebra relacional
- A inserção de um único tuplo obtém-se quando E é uma relação constante com um só tuplo.

Exemplos

- Inserir na base de dados a informação de que o Sr Smith tem 1200 dólares na conta A-973 na agência Perryridge .

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$\text{depositor} \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

- Ofereça a todos os clientes com empréstimos na agência Perryridge, 200 dólares numa nova conta poupança. Considere que o número do émpréstimo será o número da nova conta.

$$r_1 \leftarrow (\sigma_{branch\_name = \text{“Perryridge”}} (borrower \bowtie loan))$$

$$account \leftarrow account \cup \pi_{loan\_number,\ branch\_name,\ 200} (r_1)$$

$$\text{depositor} \leftarrow depositor \cup \pi_{customer\_name,\ loan\_number} (r_1)$$

Actualização

- Uma mecanismo para alterar um valor num tuplo sem alterar todos os valores do tuplo.
- Usa-se o operador projecção generalizada para esta tarefa

  $r \leftarrow \pi_{F1,F2,..,Fm}(r)$

- Cada F_i é um de:
  - o i-ésimo atributo de r, se o i-ésimo atributo não é actualizado ou,
  - Se o atributo é actualizado F_i é uma expressão com constantes e atributos de r

Exemplos de actualização

- Faça o pagamento dos Juros aumentando 5% em todos os saldos.

$$account \leftarrow \pi_{account\_number,\ branch\_name,\ balance * 1.05}(account)$$

- Pague 6% de Juro às contas com um saldo superior a 10.000 e 5% de Juro às outras contas.

$$account \leftarrow \pi_{account\_number,\ branch\_name,\ balance * 1.06}(\sigma_{BAL > 10000}(account)) \cup \pi_{account\_number,\ branch\_name,\ balance * 1.05}(\sigma_{BAL \leq 10000}(account))$$

Fim do Capítulo 2

Database System Concepts, 5th Ed.

Figure 2.3. The *branch* relation

branch_name	*branch_city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

Figure 2.6: The *loan* relation

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Figure 2.7: The *borrower* relation

customer_name	*loan_number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Figure 2.9

Result of $\sigma_{branch\_name = \text{“Perryridge”}}$ (*loan*)

loan_number	*branch_name*	*amount*
L-15	Perryridge	1500
L-16	Perryridge	1300

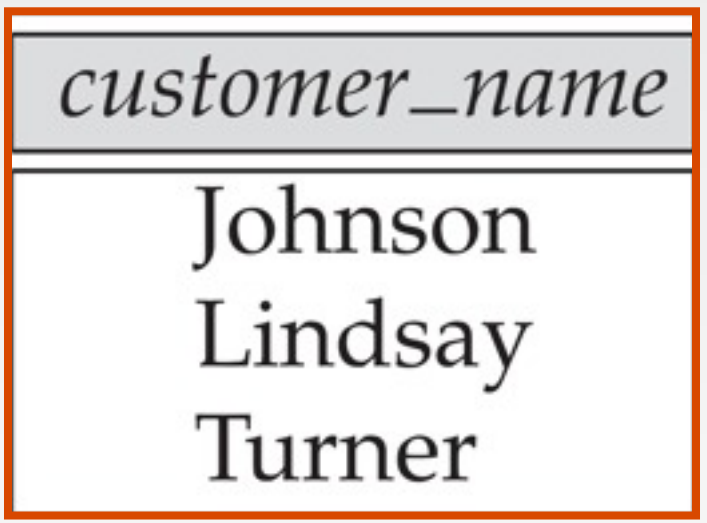
Figure 2.10: Loan number and the amount of the loan

loan_number	*amount*
L-11	900
L-14	1500
L-15	1500
L-16	1300
L-17	1000
L-23	2000
L-93	500

Figure 2.11: Names of all customers who have either an account or an loan

customer_name
Adams
Curry
Hayes
Jackson
Jones
Smith
Williams
Lindsay
Johnson
Turner

Figure 2.12: Customers with an account but no loan



customer_name
Johnson
Lindsay
Turner

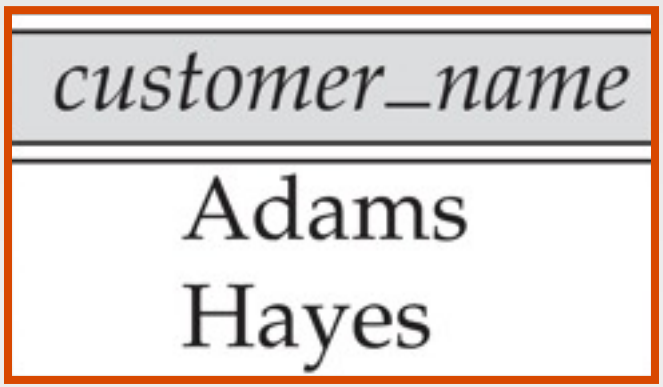
Figure 2.13: Result of *borrower* |X| *loan*

customer_name	borrower. loan_number	loan. loan_number	branch_name	amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

Figure 2.14

customer_name	*borrower. loan_number*	*loan. loan_number*	*branch_name*	*amount*
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500
Hayes	L-15	L-16	Perryridge	1300
Jackson	L-14	L-15	Perryridge	1500
Jackson	L-14	L-16	Perryridge	1300
Jones	L-17	L-15	Perryridge	1500
Jones	L-17	L-16	Perryridge	1300
Smith	L-11	L-15	Perryridge	1500
Smith	L-11	L-16	Perryridge	1300
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300

Figure 2.15



customer_name
Adams
Hayes

Figure 2.16

balance
500
400
700
750
350

Figure 2.17
Largest account balance in the bank

Figure 2.18: Customers who live on the same street and in the same city as Smith

customer_name
Curry
Smith

Figure 2.19: Customers with both an account and a loan at the bank

customer_name
Hayes
Jones
Smith

Figure 2.20

customer_name	*loan_number*	*amount*
Adams	L-16	1300
Curry	L-93	500
Hayes	L-15	1500
Jackson	L-14	1500
Jones	L-17	1000
Smith	L-23	2000
Smith	L-11	900
Williams	L-17	1000

Figure 2.21

branch_name
Brighton
Perryridge

Figure 2.22

branch_name
Brighton
Downtown

Figure 2.23

customer_name	*branch_name*
Hayes	Perryridge
Johnson	Downtown
Johnson	Brighton
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

Figure 2.24: The *credit_info* relation

customer_name	*limit*	*credit_balance*
Curry	2000	1750
Hayes	1500	1500
Jones	6000	700
Smith	2000	400

Figure 2.25

customer_name	*credit_available*
Curry	250
Jones	5300
Smith	1600
Hayes	0

Figure 2.26: The *pt_works* relation

employee_name	*branch_name*	*salary*
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

Figure 2.27
The *pt_works* relation after regrouping

employee_name	*branch_name*	*salary*
Rao	Austin	1500
Sato	Austin	1600
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300

Figure 2.28

branch_name	*sum of salary*
Austin	3100
Downtown	5300
Perryridge	8100

Figure 2.29

branch_name	*sum_salary*	*max_salary*
Austin	3100	1600
Downtown	5300	2500
Perryridge	8100	5300

Figure 2.30
The *employee* and *ft_works relations*

employee_name	*street*	*city*
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

employee_name	*branch_name*	*salary*
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

Figure 2.31

employee_name	*street*	*city*	*branch_name*	*salary*
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

Figure 2.32

employee_name	*street*	*city*	*branch_name*	*salary*
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Revolver	Death Valley	*null*	*null*

Figure 2.33

employee_name	*street*	*city*	*branch_name*	*salary*
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Gates	*null*	*null*	Redmond	5300

Figure 2.34

employee_name	*street*	*city*	*branch_name*	*salary*
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Revolver	Death Valley	*null*	*null*
Gates	*null*	*null*	Redmond	5300