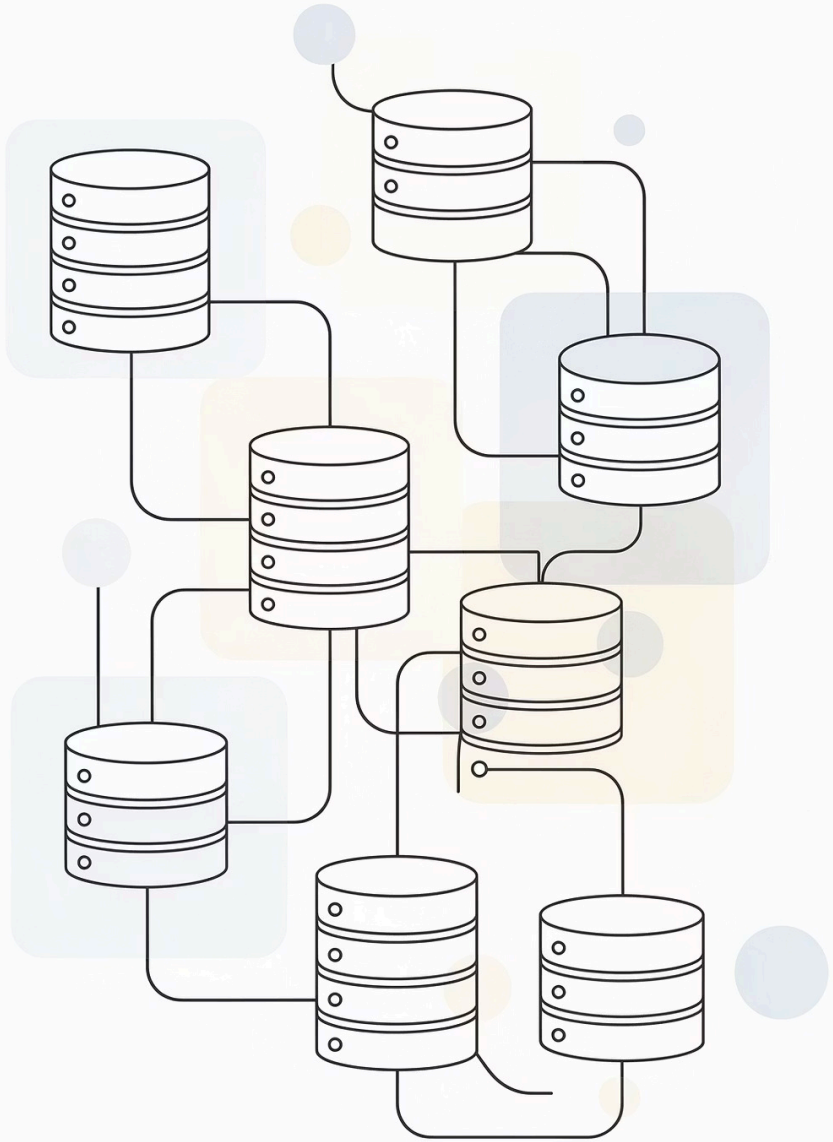


CEFET/RJ

Álgebra Relacional

Fundamentos e Operações no Modelo Relacional de Banco de Dados

Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

FUNDAMENTOS

O que é Álgebra Relacional?

Definição e Propósito

A **álgebra relacional** é um conjunto básico e fundamental de operações matemáticas projetadas especificamente para o modelo relacional de banco de dados. Essas operações formam a base teórica que sustenta todas as consultas realizadas em bancos de dados relacionais modernos.

Uma **expressão da álgebra relacional** consiste em uma sequência organizada dessas operações, permitindo a manipulação e recuperação de dados de forma estruturada e eficiente.

Objetivo Principal

O principal objetivo da álgebra relacional é fornecer uma base matemática sólida que permite o cálculo e a execução de consultas declaradas em linguagens de nível mais alto, como SQL.

Ela serve como a **ponte conceitual** entre como os usuários expressam suas necessidades de dados e como o sistema de gerenciamento de banco de dados (SGBD) efetivamente recupera essas informações.

Álgebra Relacional como Linguagem Procedimental

Natureza Procedimental

A álgebra relacional é uma **linguagem procedimental**, o que significa que ela especifica *como* obter os dados desejados através de uma sequência explícita de passos.

Sequência Ordenada

Uma consulta é definida por uma **sequência ordenada de operações**. A ordem importa, pois cada passo depende do resultado do passo anterior.

Operações e Resultados

Cada operação recebe **uma ou duas relações** como entrada e sempre produz **uma nova relação** como saída, garantindo o fechamento algébrico.

Composição

O resultado final de uma consulta complexa depende da **composição cuidadosa das operações**, onde a saída de uma operação alimenta a entrada da próxima.

Importante: Diferentemente de linguagens declarativas como SQL, onde você especifica *o que* deseja, a álgebra relacional exige que você especifique *como* obter o resultado.

ESQUEMA DE REFERÊNCIA

Esquema de Exemplo Usado

Para ilustrar as operações da álgebra relacional, utilizaremos um esquema de banco de dados bancário simplificado. Este esquema representa as principais entidades e relacionamentos típicos de um sistema bancário.

agencia

Atributos: nomeAgencia, cidadeAgencia, ativo

Armazena informações sobre as agências bancárias.

cliente

Atributos: nomeCliente, ruaCliente, cidadeCliente

Contém os dados cadastrais dos clientes do banco.

conta

Atributos: numeroConta, nomeAgencia*, saldo

Registra as contas bancárias e seus saldos.

emprestimo

Atributos: numeroEmprestimo, nomeAgencia*, quantia

Documenta os empréstimos concedidos pelo banco.

depositante

Atributos: nomeCliente*, numeroConta*

Relaciona clientes às suas contas bancárias.

tomador

Atributos: nomeCliente*, numeroEmprestimo*

Relaciona clientes aos seus empréstimos.

Nota: Os atributos marcados com asterisco (*) são chaves estrangeiras, estabelecendo relacionamentos entre as tabelas do esquema.

OPERAÇÃO UNÁRIA

Operação Seleção (σ)

A operação de **seleção**, denotada pelo símbolo grego σ **(sigma)**, é uma operação unária fundamental que permite filtrar tuplas (linhas) de uma relação com base em um predicado (condição) específico.

A seleção examina cada tupla da relação de entrada e inclui no resultado apenas aquelas tuplas que **satisfazem a condição especificada**. É análoga à cláusula WHERE em SQL.

Notação Formal

$$\sigma_{\text{predicado}}(R)$$

Onde *predicado* é a condição de filtragem e *R* é a relação de entrada.

Características

- Mantém todos os atributos (colunas) da relação original
- Reduz o número de tuplas no resultado
- O predicado pode usar operadores de comparação (=, $\neq$, <, >, $\leq$, $\geq$)
- Pode combinar múltiplas condições com operadores lógicos ($\wedge$, $\vee$, $\neg$)

Operação Seleção – Exemplo Prático

Demonstraremos como a operação de seleção funciona utilizando exemplos concretos do nosso esquema bancário. A seleção é uma operação unária que permite filtrar tuplas (linhas) de uma relação com base em uma condição específica, mantendo todas as colunas.

Relação R (conta) - Entrada:

numeroConta	nomeAgencia	saldo
A-101	Centro	5000
A-102	Paulista	12000
A-103	Centro	8500
A-104	Jardins	3200

Exemplo de Seleção

$\sigma_{\text{saldo}>5000}(\text{conta})$

Resultado: Retorna apenas as tuplas (linhas) onde o saldo da conta é maior que 5000. Todas as colunas são mantidas.

numeroConta	nomeAgencia	saldo
A-102	Paulista	12000
A-103	Centro	8500

A operação de seleção filtra linhas (subconjunto horizontal), mantendo todas as colunas.

OPERAÇÃO UNÁRIA

Operação Projeção (π)

A operação de **projeção**, representada pela letra grega π, é outra operação unária essencial que permite selecionar atributos (colunas) específicos de uma relação, descartando os demais.

Diferentemente da seleção, que filtra linhas, a projeção **filtra colunas**, retornando apenas os atributos de interesse. É equivalente à cláusula SELECT em SQL (quando usada para especificar colunas).

Notação Formal

$$\pi_{\text{atributo1, atributo2, ...}}(R)$$

Onde a lista de atributos especifica quais colunas manter.

Características Importantes

- Reduz o número de atributos (colunas) no resultado
- Mantém todas as tuplas, mas **elimina duplicatas automaticamente**
- A ordem dos atributos no resultado corresponde à ordem especificada na operação
- Pode ser combinada com outras operações para consultas mais complexas

Operação Projeção – Exemplo Prático

Demonstraremos como a projeção funciona utilizando exemplos concretos do nosso esquema bancário.

Relação R (conta) - Entrada:

numeroConta	nomeAgencia	saldo
A-101	Centro	5000
A-102	Paulista	12000
A-103	Centro	8500
A-104	Jardins	3200

Exemplo 1

$\pi_{\text{nomeAgencia, saldo}}(\text{conta})$

Resultado: Retorna apenas as colunas nomeAgencia e saldo, eliminando numeroConta.

nomeAgencia	saldo
Centro	5000
Paulista	12000
Centro	8500
Jardins	3200

Exemplo 2

$\pi_{\text{nomeAgencia}}(\text{conta})$

Resultado: Retorna apenas agências únicas (duplicatas removidas).

nomeAgencia
Centro
Paulista
Jardins

Note que "Centro" aparece apenas uma vez, apesar de existir em duas tuplas originais.

OPERAÇÃO BINÁRIA

Operação União (∪)

A operação de **união**, denotada pelo símbolo ∪, é uma operação binária que combina duas relações compatíveis, produzindo uma nova relação contendo todas as tuplas que aparecem em **pelo menos uma** das relações originais.

01

Compatibilidade de União

Para que duas relações R e S possam ser unidas, elas devem ser **compatíveis em união**, ou seja: possuir o mesmo número de atributos, e os atributos correspondentes devem ter domínios compatíveis.

02

Eliminação de Duplicatas

A união automaticamente **elimina tuplas duplicadas** do resultado. Se uma tupla aparece em ambas as relações R e S, ela aparecerá apenas uma vez no resultado $R \cup S$.

03

Propriedades Matemáticas

A união é **comutativa** ($R \cup S = S \cup R$) e **associativa** ($(R \cup S) \cup T = R \cup (S \cup T)$), permitindo flexibilidade na ordem de execução.

> **Notação Formal:** $R \cup S$ resulta em uma relação contendo todas as tuplas de R, mais todas as tuplas de S, sem duplicatas.

Operação União – Exemplo Prático

Vamos ilustrar a operação de união com um exemplo concreto usando duas relações compatíveis.

Relação R

Nome	Cidade
João	São Paulo
Maria	Rio de Janeiro
Pedro	Belo Horizonte

Clientes da região Sudeste

Relação S

Nome	Cidade
Maria	Rio de Janeiro
Ana	Curitiba
Carlos	Porto Alegre

Clientes da região Sul

$R \cup S$

Nome	Cidade
João	São Paulo
Maria	Rio de Janeiro
Pedro	Belo Horizonte
Ana	Curitiba
Carlos	Porto Alegre

Todos os clientes únicos

Observação: A tupla (Maria, Rio de Janeiro) aparecia em ambas as relações, mas aparece apenas **uma vez** no resultado da união. O resultado contém 5 tuplas, não 6.

OPERAÇÃO BINÁRIA

Operação de Interseção (∩)

A operação de **interseção**, representada pelo símbolo ∩, é uma operação binária que produz uma nova relação contendo apenas as tuplas que aparecem simultaneamente em ambas as relações de entrada.

Assim como a união, a interseção requer que as relações sejam **compatíveis em união**, com o mesmo número de atributos e domínios correspondentes compatíveis.

Notação Formal

$$R \cap S$$

Retorna todas as tuplas que existem tanto em R quanto em S.

Propriedades Essenciais

- **Comutatividade:** $R \cap S = S \cap R$ (a ordem não afeta o resultado)
- **Associatividade:** $(R \cap S) \cap T = R \cap (S \cap T)$
- **Idempotência:** $R \cap R = R$
- O resultado é sempre um subconjunto de ambas as relações originais

Equivalência com Outras Operações

A interseção pode ser expressa em termos de diferença:
$R \cap S = R - (R - S)$

Essa equivalência mostra que a interseção não é uma operação primitiva, mas é incluída por conveniência.

Operação Interseção – Exemplo Prático

Vejamos como a interseção funciona identificando elementos comuns entre duas relações.

Relação R

Produto	Categoria
Notebook	Eletrônicos
Mouse	Periféricos
Teclado	Periféricos
Monitor	Eletrônicos

Produtos da Loja A

Relação S

Produto	Categoria
Mouse	Periféricos
Webcam	Periféricos
Notebook	Eletrônicos
Impressora	Periféricos

Produtos da Loja B

$R \cap S$

Produto	Categoria
Notebook	Eletrônicos
Mouse	Periféricos

Produtos em ambas as lojas

> **Análise do Resultado:** Apenas Notebook e Mouse aparecem em ambas as relações R e S, portanto são os únicos produtos incluídos no resultado da interseção. Note que tanto "Teclado" e "Monitor" (exclusivos de R) quanto "Webcam" e "Impressora" (exclusivos de S) foram excluídos do resultado.

OPERAÇÃO BINÁRIA

Operação de Diferença (-)

A operação de **diferença**, denotada pelo símbolo **-** ou ****, produz uma relação contendo todas as tuplas que aparecem na primeira relação mas não aparecem na segunda relação.

1

Definição Formal

$R - S$ resulta em uma relação contendo todas as tuplas que estão em R mas não estão em S. As relações devem ser compatíveis em união.

2

Não-Comutatividade

Diferentemente da união e interseção, a diferença **não é comutativa**: $R - S \neq S - R$. A ordem das operações é crucial e produz resultados diferentes.

3

Aplicações Práticas

Útil para encontrar registros exclusivos, identificar clientes inativos, produtos descontinuados, ou qualquer cenário onde precisamos **excluir um conjunto de outro**.

Importante: $R - S$ e $S - R$ são operações distintas e geralmente produzem resultados completamente diferentes. Sempre verifique a ordem correta para sua consulta.

Operação de Diferença – Exemplo Prático

Demonstraremos a diferença entre conjuntos e a importância da ordem na operação.

Relação R

ID	Nome
1	Alice
2	Bruno
3	Carlos
4	Diana

Clientes cadastrados

Relação S

ID	Nome
2	Bruno
4	Diana
5	Eduardo

Clientes com compras

$R - S$

ID	Nome
1	Alice
3	Carlos

Clientes sem compras

$S - R$

ID	Nome
5	Eduardo

Compras sem cadastro

> **Interpretação dos Resultados:** R - S identifica Alice e Carlos, que estão cadastrados mas nunca fizeram compras. Por outro lado, S - R encontra Eduardo, que realizou uma compra mas não está no cadastro de clientes. Note como a mudança na ordem produz resultados completamente diferentes e ambos têm significados práticos importantes.

OPERAÇÃO BINÁRIA

Operação de Produto Cartesiano ($\times$)

Conceito e Funcionamento

A operação de **produto cartesiano**, denotada por $\times$, combina duas relações criando uma nova relação que contém todas as combinações possíveis de tuplas das duas relações originais.

Se R tem *n* tuplas e *m* atributos, e S tem *p* tuplas e *q* atributos, então $R \times S$ terá $n \times p$ tuplas e $m + q$ atributos.

Notação e Características

$R \times S$ ou $R \times S$

- Não requer compatibilidade de união
- Combina cada tupla de R com cada tupla de S
- Pode gerar relações muito grandes
- Base para a operação de junção (join)

n x p

Número de Tuplas

O resultado contém o produto do número de tuplas das duas relações

m + q

Número de Atributos

O resultado possui a soma dos atributos de ambas as relações

Operação de Produto Cartesiano – Exemplo Prático ($R \times S$)

Vejamos como o produto cartesiano combina todas as possíveis tuplas de duas relações.

Relação R

A	B
a1	b1
a2	b2

2 tuplas, 2 atributos

Relação S

C	D
c1	d1
c2	d2
c3	d3

3 tuplas, 2 atributos

R × S

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a1	b1	c3	d3
a2	b2	c1	d1
a2	b2	c2	d2
a2	b2	c3	d3

6 tuplas (2×3), 4 atributos ($2 + 2$)

Cuidado com o Tamanho: O produto cartesiano pode crescer rapidamente. Se R tem 1.000 tuplas e S tem 1.000 tuplas, R × S terá 1.000.000 de tuplas! Por isso, é geralmente seguido por uma seleção para filtrar apenas as combinações relevantes, formando a base da operação de junção.

CONCEITOS AVANÇADOS

Expressão Algébrica: Definição Formal

Uma **expressão algébrica relacional** é construída pela composição de operações da álgebra relacional, permitindo expressar consultas complexas através da combinação estruturada de operações primitivas.

Operações Atômicas

Uma relação nomeada do banco de dados é uma expressão algébrica válida (caso base).

Composição Unária

Se E é uma expressão algébrica, então $\sigma_P(E)$, $\pi_L(E)$ e $\rho_x(E)$ também são expressões válidas, onde P é um predicado, L é uma lista de atributos, e x é um novo nome.

Composição Binária

Se E1 e E2 são expressões algébricas, então $E_1 \cup E_2$, $E_1 - E_2$, $E_1 \times E_2$ também são expressões válidas (assumindo compatibilidade onde necessário).

Precedência e Agrupamento

Operações podem ser agrupadas usando parênteses para controlar a ordem de avaliação, similar a expressões matemáticas.

Exemplo de Expressão Complexa

$$\pi_{\text{nomeCliente}}(\sigma_{\text{saldo}>5000}(\text{depositante} \bowtie \text{conta}))$$

Esta expressão encontra os nomes de todos os clientes que possuem contas com saldo superior a R$ 5.000, através da composição de junção natural ($\bowtie$), seleção (σ) e projeção (π).

Propriedades Importantes

- **Fechamento:** O resultado de qualquer expressão algébrica é sempre uma relação
- **Equivalência:** Múltiplas expressões podem produzir o mesmo resultado
- **Otimização:** Expressões equivalentes podem ter custos de execução diferentes

Expressão Algébrica como Programa Relacional

Na álgebra relacional, uma expressão algébrica funciona como um programa completo de processamento de dados. Cada operador representa uma etapa específica de transformação, e o fluxo de execução é determinado pela estrutura hierárquica da expressão. Os resultados intermediários são relações temporárias que servem como entrada para as próximas operações, criando um pipeline de processamento de dados elegante e eficiente.

Exemplo de Expressão:

$\pi_{\text{nome, salario}}(\sigma_{\text{departamento}=\text{'TI'} \wedge \text{salario} > 5000}(\text{funcionario}))$

Esta expressão seleciona funcionários do departamento de TI com salário > 5000 e projeta apenas os atributos nome e salário.

FUNDAMENTOS

Exemplo de Expressão Algébrica

Uma expressão algébrica em bancos de dados relacionais combina operadores como seleção (σ), projeção (π) e junção ($\bowtie$) para manipular relações. Cada operação transforma uma ou mais relações de entrada em uma nova relação de saída.

A beleza da álgebra relacional está na sua capacidade de expressar consultas complexas através da composição de operações simples e bem definidas.

Exemplo Concreto:

$$\pi_{\text{nomeCliente, cidade}}(\sigma_{\text{saldo}>1000}(\text{cliente} \bowtie_{\text{cliente.id}=\text{conta.clienteId}} \text{conta}))$$

Esta expressão une as tabelas 'cliente' e 'conta', filtra as contas com saldo maior que 1000, e projeta apenas o nome do cliente e a cidade.

As expressões algébricas formam a base teórica para linguagens de consulta como SQL, permitindo otimizações automáticas pelo sistema gerenciador de banco de dados.

EQUIVALÊNCIA

Equivalência de Expressões Algébricas

Mesmo Resultado

Expressões diferentes podem produzir resultados idênticos, mesmo quando estruturadas de formas distintas

Independência da Forma

A equivalência depende do resultado final, não da sintaxe ou ordem das operações

Otimização de Consultas

Permite ao SGBD escolher a execução mais eficiente entre alternativas equivalentes

Um exemplo clássico de equivalência ocorre com seleções e junções: aplicar uma seleção **antes** de uma junção pode ser equivalente a aplicá-la **depois**, mas a primeira abordagem geralmente é muito mais eficiente, pois reduz o número de tuplas processadas na junção.

Exemplos Concretos:

$$\sigma_{A=5}(\sigma_{B='X'}(R)) \equiv \sigma_{B='X'}(\sigma_{A=5}(R)) \equiv \sigma_{A=5 \wedge B='X'}(R)$$

Todas as três expressões filtram a relação R para tuplas onde A=5 e B='X', demonstrando a comutatividade das operações de seleção.

Operação de Atribuição

A operação de atribuição permite armazenar o resultado de uma expressão algébrica em uma relação temporária, facilitando a construção de consultas complexas em etapas.

Sintaxe básica: $\text{nome\_relacao} \leftarrow \text{expressão\_algébrica}$

Esta operação é fundamental para:

- Dividir consultas complexas em partes menores e mais compreensíveis
- Reutilizar resultados intermediários em múltiplas operações subsequentes
- Melhorar a legibilidade e manutenibilidade das expressões algébricas
- Facilitar o processo de otimização de consultas

Exemplos Práticos:

$$\begin{gathered}\text{temp1} \leftarrow \sigma_{\text{salario}>3000}(\text{funcionario}) \\ \text{temp2} \leftarrow \pi_{\text{nome, departamento}}(\text{temp1}) \\ \text{resultado} \leftarrow \text{temp2}\end{gathered}$$

Este exemplo demonstra como uma consulta complexa é dividida em etapas menores e gerenciáveis usando relações temporárias, tornando-a mais clara e fácil de seguir.

Estrutura da Operação de Atribuição

Relação Original

Dados de entrada para processamento

Expressão Algébrica

Operações aplicadas aos dados

Símbolo ←

Operador de atribuição

Relação Temporária

Resultado armazenado

A atribuição permite construir consultas modulares, onde cada etapa representa uma transformação lógica dos dados. Por exemplo, podemos primeiro filtrar clientes ativos, depois calcular suas compras totais, e finalmente ordenar por valor.

Exemplo de Pipeline:

$$\text{clientes\_vip} \leftarrow \sigma_{\text{total\_compras} > 10000}(\text{cliente}) \quad \text{pedidos\_vip} \leftarrow \text{clientes\_vip} \bowtie_{\text{cliente.id=pedido.clienteId}} \text{pedido} \quad \text{resultado} \leftarrow \pi_{\text{nome, data\_pedido, valor}}(\text{pedidos\_vip})$$

Este exemplo demonstra como a atribuição cria um pipeline de processamento com resultados intermediários.

Junção θ como Operação Derivada

A junção θ é uma operação derivada na álgebra relacional, o que significa que ela pode ser expressa e construída a partir de operações básicas como o produto cartesiano e a seleção.

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

Nesta definição formal, a condição θ (theta) é aplicada **após** o produto cartesiano das duas relações R e S, filtrando as tuplas que satisfazem a condição especificada. A junção natural, por exemplo, é um caso particular da junção θ, onde a condição θ é baseada na igualdade de atributos com o mesmo nome entre as relações envolvidas.

Exemplo de Junção θ:

$funcionario \bowtie_{funcionario.salario > gerente.salario} gerente$

Equivalente a:

$\sigma_{funcionario.salario > gerente.salario}(funcionario \times gerente)$

Este exemplo demonstra como a junção theta pode ser expressa usando o produto cartesiano seguido de uma seleção.

JUNÇÃO

Operação de Junção Natural

A junção natural (⋈) é uma das operações mais importantes da álgebra relacional. Ela combina tuplas de duas relações que possuem valores iguais em atributos com o mesmo nome, criando uma nova relação que integra informações de ambas as fontes.

01

Identificação de Atributos Comuns

O sistema identifica automaticamente todos os atributos que aparecem em ambas as relações

02

Comparação de Valores

Para cada par de tuplas, compara os valores dos atributos comuns

03

Combinação de Tuplas

Quando os valores coincidem, as tuplas são combinadas em uma única tupla resultado

04

Eliminação de Duplicatas

Os atributos comuns aparecem apenas uma vez na relação resultante

Exemplos de Junção

- **Equijoin:** $\text{empregado} \bowtie_{\text{empregado.deptId}=\text{departamento.id}} \text{departamento}$
- **Theta join:** $\text{produto} \bowtie_{\text{produto.preco}<\text{venda.valor}} \text{venda}$
- **Natural join:** $\text{cliente} \bowtie \text{pedido}$ (une-se automaticamente em atributos comuns)

Junção Natural – Exemplo Prático

Considere duas relações **Cliente** e **Pedido**. A junção natural combina as tuplas onde os valores dos atributos comuns (neste caso, `ID_Cliente`) coincidem.

Expressão Algébrica:

$$\text{resultado} \leftarrow \text{Cliente} \bowtie \text{Pedido}$$

Ou com seleção explícita de atributos:

$$\pi_{\text{Cliente.Nome\_Cliente, Pedido.Data\_Pedido, Pedido.Valor}}(\text{Cliente} \bowtie \text{Pedido})$$

Tabela Cliente

1	Ana	Lisboa
2	Bruno	Porto
3	Carla	Lisboa

Tabela Pedido

101	1	2023-01-15	150
102	2	2023-01-16	200
103	1	2023-01-17	100
104	4	2023-01-18	300

Resultado da Junção (Cliente □ Pedido)

1	Ana	Lisboa	101	2023-01-15	150
1	Ana	Lisboa	103	2023-01-17	100
2	Bruno	Porto	102	2023-01-16	200

O resultado da junção natural $\text{Cliente} \bowtie \text{Pedido}$ contém todas as tuplas onde os valores do atributo comum `ID_Cliente` são idênticos. Tuplas sem correspondência (como o Pedido 104) são automaticamente excluídas do resultado, caracterizando esta como uma junção interna.

A junção natural é comutativa ($R \bowtie S = S \bowtie R$) e associativa ($(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$), propriedades fundamentais para otimização de consultas.

EXTENSÃO

Projeção Generalizada

A projeção generalizada estende a operação de projeção tradicional, permitindo não apenas selecionar atributos existentes, mas também criar novos atributos através de expressões aritméticas e funções aplicadas aos atributos originais.

Seleção de Atributos

Escolha dos atributos existentes que farão parte do resultado, similar à projeção tradicional

Cálculo de Expressões

Criação de novos atributos através de operações matemáticas: adição, subtração, multiplicação, divisão

Aplicação de Funções

Uso de funções como CONCAT, SUBSTRING, UPPER, LOWER para manipular strings e outros tipos

Renomeação

Atribuição de novos nomes aos atributos resultantes para maior clareza e usabilidade

A seguir, alguns exemplos práticos que ilustram a criação de novos atributos através de expressões aritméticas:

$$\pi_{\text{nome, salario, salario} \times 1.10 \text{ as novo\_salario}}(\text{funcionario})$$

$$\pi_{\text{produto, preco, quantidade, preco} \times \text{quantidade as total}}(\text{item\_pedido})$$

Estes exemplos demonstram o poder da projeção generalizada para calcular e adicionar novos campos a uma relação com base em atributos existentes, como um novo salário com aumento ou o valor total de um item de pedido.

AGREGAÇÃO

Funções e Operações Agregadas

As operações agregadas permitem realizar cálculos sobre conjuntos de tuplas, produzindo valores resumidos que sintetizam características importantes dos dados. São essenciais para análises estatísticas e relatórios gerenciais.

SUM

Calcula a soma de valores numéricos em um atributo

AVG

Determina a média aritmética dos valores

COUNT

Conta o número de tuplas ou valores não nulos

MAX

Identifica o valor máximo em um conjunto

MIN

Identifica o valor mínimo em um conjunto

Estas funções podem ser aplicadas sobre toda a relação ou sobre grupos de tuplas que compartilham valores em atributos específicos, utilizando a operação de agrupamento (GROUP BY).

Exemplos de Agregação

Estes exemplos mostram diferentes funções agregadas com cenários do mundo real:

- $\mathrm{SUM}(\mathrm{salario})(\mathrm{funcionario})$ - soma total dos salários
- $\mathrm{AVG}(\mathrm{nota})(\mathrm{aluno})$ - média das notas
- $\mathrm{COUNT}(\mathrm{id})(\mathrm{cliente})$ - contagem de clientes
- $\mathrm{MAX}(\mathrm{preco})(\mathrm{produto})$ - preço máximo

Operação Agregada – Exemplo com SUM

Relação 'vendas' Original

Considere uma relação 'vendas' com um atributo numérico 'valor' contendo os seguintes dados:

```
vendas = {(100), (250), (175), (300)}
```

A operação agregada $\mathrm{SUM}(\mathrm{valor})$ percorre todas as tuplas da relação 'vendas', acumula os valores do atributo 'valor' e retorna um único valor resultante.

Expressão Algébrica e Resultado da Agregação

A notação algébrica para a soma dos valores na relação 'vendas' é:

$$\Gamma_{\mathrm{SUM(valor)}}(\mathrm{vendas})$$

Realizando o cálculo:

```
100 + 250 + 175 + 300 = 825
```

$\mathrm{SUM}(\mathrm{valor})$
825

O valor **825** representa a soma de todos os valores presentes no atributo 'valor' da relação 'vendas'.

Este exemplo demonstra como uma operação agregada transforma múltiplas tuplas em um único valor resumido, fundamental para análises quantitativas e geração de relatórios consolidados.

Operação Agregada com Agrupamento

Quando combinamos agregação com agrupamento, podemos calcular estatísticas separadamente para cada grupo de tuplas que compartilham valores em atributos específicos. Este é um dos recursos mais poderosos da álgebra relacional.

Dados Originais

nomeAgencia	numeroConta	saldo
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

Resultado Agregado por Agência

nomeAgencia	sum(saldo)
Perryridge	1300
Brighton	1500
Redwood	700

A soma dos saldos é calculada separadamente para cada agência bancária.

A seguir, outros exemplos de expressões algébricas para operações de agrupamento e agregação:

Exemplo com Agrupamento:

A expressão abaixo agrupa os funcionários por departamento e calcula a contagem e a média salarial para cada departamento.

$$\Gamma_{\text{departamento}}^{\text{COUNT(id)},\text{AVG(salario)}}(\text{funcionario})$$

Outro exemplo:

Esta expressão agrupa produtos por categoria, calculando a soma das quantidades e o preço máximo para cada categoria.

$$\Gamma_{\text{categoria}}^{\text{SUM(quantidade)},\text{MAX(preco)}}(\text{produto})$$

JUNÇÕES ESTENDIDAS

Junção Externa (Outer Join)

A junção externa é uma extensão poderosa da operação de junção tradicional que resolve um problema importante: evitar a perda de informações quando algumas tuplas não encontram correspondência na outra relação.

Junção Tradicional

Mantém apenas tuplas com correspondência em ambas as relações

1

Junção Externa

Preserva todas as tuplas, preenchendo com NULL onde não há correspondência

3

2

Identificação de Perdas

Tuplas sem correspondência são descartadas, resultando em perda de informação

Os valores NULL representam informações desconhecidas ou inexistentes. É importante entender que todas as comparações envolvendo NULL são tratadas de forma especial pelo sistema: elas não retornam verdadeiro nem falso, mas um terceiro valor lógico que representa "desconhecido".

Existem três tipos de junção externa: esquerda (LEFT), direita (RIGHT) e completa (FULL), cada uma preservando tuplas de diferentes lados da junção.

Exemplos de Junções Estendidas:

- **Left outer join:** $\text{cliente} ⟕ \text{pedido}$ (mantém todos os clientes)
- **Right outer join:** $\text{pedido} ⟖ \text{produto}$ (mantém todos os produtos)
- **Full outer join:** $\text{funcionario} ⟗ \text{departamento}$ (mantém ambos)

Junção Interna (Inner Join)

A junção interna é a operação de junção tradicional, que mantém apenas as tuplas que possuem correspondência em ambas as relações. Tuplas sem correspondência são completamente eliminadas do resultado.

Em notação algébrica, uma junção interna entre duas relações `cliente` e `pedido` pode ser expressa como:

Expressão Algébrica:

$$\text{cliente} \bowtie_{\text{cliente.id}=\text{pedido.clienteId}} \text{pedido}$$

Ou, usando a notação de theta join, se as colunas forem distintas:

$$\text{cliente} \bowtie_{\text{id}=\text{clienteId}} \text{pedido}$$

Esta operação garante que apenas as tuplas que têm valores correspondentes na condição de junção de ambas as relações são incluídas no resultado.

Relação 1

a	b
L-170	Red
L-230	Green
L-160	Blue

Relação 2

b	c	d
Red	3000	Jones
Green	4000	Smith
Yellow	1000	Suzan

Resultado

b	c	d
Red	3000	Jones
Green	4000	Smith

Observe que a tupla com **b = Blue** da primeira relação e a tupla com **b = Yellow** da segunda relação não aparecem no resultado, pois não encontraram correspondência. A junção interna só inclui as tuplas que possuem correspondência em ambas as relações, como demonstrado no exemplo.

Junção Externa Esquerda (Left Outer Join)

A junção externa esquerda preserva **todas as tuplas da relação à esquerda**, mesmo aquelas que não encontram correspondência na relação à direita. Para tuplas sem correspondência, os atributos vindos da relação direita são preenchidos com NULL.

Expressão Algébrica:

$$\text{cliente} \, ⟕_{\text{cliente.id}=\text{pedido.clienteId}} \text{pedido}$$

Resultado: Todos os clientes aparecem. Clientes sem pedidos terão NULL nos atributos de pedido (data, valor, etc.).

Relação Esquerda

a	b
L-170	Red
L-230	Green
L-160	Blue

Relação Direita

b	c	d
Red	3000	Jones
Green	4000	Smith
Yellow	1000	Suzan

Resultado

a	b	c	d
L-170	Red	3000	Jones
L-230	Green	4000	Smith
L-160	Blue	null	null

A tupla com **b = Blue** é preservada, com NULL nos atributos c e d. Já a tupla com **b = Yellow** da relação direita é descartada, pois esta é uma junção externa *esquerda*.

Junção Externa Direita (Right Outer Join)

A junção externa direita funciona de forma simétrica à esquerda: preserva **todas as tuplas da relação à direita**, mesmo aquelas sem correspondência na relação à esquerda. Atributos sem correspondência são preenchidos com NULL.

Expressão Algébrica:

$$\text{pedido} \, ⟖_{\text{pedido.produtoId}=\text{produto.id}} \text{produto}$$

Resultado: Todos os produtos aparecem. Produtos sem pedidos terão NULL nos atributos de pedido (quantidade, data, etc.).

Relação Esquerda

a	b
L-170	Red
L-230	Green
L-160	Blue

Relação Direita

b	c	d
Red	3000	Jones
Green	4000	Smith
Yellow	1000	Suzan

Resultado

a	b	c	d
L-170	Red	3000	Jones
L-230	Green	4000	Smith
null	Yellow	1000	Suzan

Agora a tupla com **b = Yellow** é preservada com NULL no atributo a, enquanto a tupla com **b = Blue** da relação esquerda é descartada.

Junção Externa Integral (Full Outer Join)

A junção externa integral ou completa é a mais abrangente: preserva **todas as tuplas de ambas as relações**, independentemente de encontrarem correspondência ou não. É a união dos comportamentos das junções externas esquerda e direita.

Expressão Algébrica:

$$\text{funcionario} \, ⟗_{\text{funcionario.deptId}=\text{departamento.id}} \, \text{departamento}$$

Resultado:

- Funcionários sem departamento: NULL nos atributos de departamento
- Departamentos sem funcionários: NULL nos atributos de funcionario
- Funcionários com departamento: todos os atributos preenchidos

Esta operação demonstra a natureza abrangente da Junção Externa Integral, garantindo que nenhuma informação de ambas as relações seja perdida.

Relação Esquerda

a	b
L-170	Red
L-230	Green
L-160	Blue

Relação Direita

b	c	d
Red	3000	Jones
Green	4000	Smith
Yellow	1000	Suzan

Resultado Completo

a	b	c	d
L-170	Red	3000	Jones
L-230	Green	4000	Smith
L-160	Blue	null	null
null	Yellow	1000	Suzan

Tanto a tupla com **b = Blue** quanto a com **b = Yellow** são preservadas, garantindo que nenhuma informação seja perdida durante a junção. Esta é a forma mais completa de preservar dados em operações de junção.

Operação Divisão em Álgebra Relacional

A operação de divisão é uma das operações fundamentais da álgebra relacional, permitindo expressar consultas complexas que envolvem quantificação universal. Esta apresentação explora a operação divisão e suas aplicações práticas em bancos de dados relacionais, além de exercícios para consolidar o aprendizado.

Expressão Algébrica:

$$R \div S$$

Exemplo Prático:

$$\text{estudante\_curso}(\text{estudante}, \text{curso}) \div \text{cursos\_obrigatorios}(\text{curso})$$

Esta operação retorna os estudantes que cursaram TODOS os cursos obrigatórios, funcionando como a operação "para todos" na álgebra relacional.

Operação Divisão – Exemplo

Relações R e S

Expressão: $R \div S$

A operação divisão $R \div S$ retorna as tuplas de R que estão associadas a **todos** os valores em S. É como encontrar elementos que satisfazem uma condição "para todo".

Considere as relações ao lado: R contém pares (a, b) e S contém valores de b. O resultado $R \div S$ mostrará quais valores de 'a' aparecem com **todos** os valores de 'b' presentes em S, ou seja, para cada 'a' no resultado, deve haver uma tupla (a, b) em R para cada 'b' em S.

Tabelas de Exemplo

Relação R:

a	b
α	1
α	2
β	1

Relação S:

b
1
2

No exemplo acima, a Relação S contém os valores {1, 2}. O único valor de 'a' na Relação R que está associado a **ambos** 1 e 2 é 'α'. Portanto, o resultado da divisão $R \div S$ é:

Resultado $R \div S$:

a
α

Outro Exemplo de Divisão

Relação R

c	d	e
w	x	1
w	y	1
z	x	1

Relação S

d	e
x	1
y	1

Resultado $R \div S$

c
w

Apenas 'w' aparece com todas as combinações (x,1) e (y,1) de S.

Expressão Algébrica: $R \div S$, onde R é R(c, d, e) e S é S(d, e).

O resultado da divisão $R \div S$ contém a coluna 'c' porque 'd' e 'e' são os atributos comuns a ambas as relações. A operação retorna o valor 'w' da relação R porque ele é o único valor de 'c' que está associado a **todas** as tuplas presentes em S, que são (x, 1) e (y, 1). Ou seja, existem as tuplas (w, x, 1) e (w, y, 1) em R.

Neste exemplo, a divisão identifica quais valores da coluna 'c' estão associados a todas as combinações de valores (d, e) presentes em S. A operação é útil para expressar consultas do tipo "encontre X que tem relacionamento com todos os Y".

Modificação do Banco de Dados

Além das operações de consulta, a álgebra relacional também permite modificar o conteúdo armazenado no banco de dados. Essas operações são essenciais para manter os dados atualizados e refletir as mudanças no mundo real.

1

Exclusão

Remove tuplas existentes de uma relação com base em uma condição específica. Utiliza o operador de atribuição para criar uma nova relação sem as tuplas removidas.

$$\text{conta} \leftarrow \text{conta} - \sigma_{\text{saldo}<100}(\text{conta})$$

2

Inserção

Adiciona novas tuplas a uma relação existente. Pode inserir tuplas individuais ou o resultado de uma expressão de consulta em uma relação.

$$\text{cliente} \leftarrow \text{cliente} \cup \{(\text{'João'}, \text{'SP'}, 1000)\}$$

3

Atualização

Modifica valores de atributos em tuplas existentes sem remover as tuplas. Permite alterar dados específicos mantendo a estrutura da relação intacta.

$$\text{conta} \leftarrow \pi_{\text{numeroConta, nomeAgencia, saldo}\times 1.05}(\text{conta})$$

Todas essas operações são expressas usando o **operador de atribuição (**$\leftarrow$**)**, que cria uma nova relação ou modifica uma existente com base no resultado de uma expressão de álgebra relacional.

Exclusão

Sintaxe Geral

$r \leftarrow r - E$

Onde:

- **r** é a relação
- **E** é uma expressão de álgebra relacional

Como Funciona

A operação de exclusão remove tuplas de uma relação. O resultado é uma nova relação que contém todas as tuplas da relação original, **exceto** aquelas que satisfazem a condição especificada na expressão E.

A exclusão é implementada através da operação de diferença de conjuntos, onde subtraímos o conjunto de tuplas a serem removidas da relação original.

Exemplos de Exclusão:

1. Para excluir todas as contas da agência 'Perryridge':

$\text{conta} \leftarrow \text{conta} - \sigma_{\text{nomeAgencia='Perryridge'}}(\text{conta})$

2. Para excluir todos os empréstimos com quantia inferior a 1000:

$\text{emprestimo} \leftarrow \text{emprestimo} - \sigma_{\text{quantia}<1000}(\text{emprestimo})$

Inserção

Inserção de Tupla Única

$r \leftarrow r \cup \{t\}$

Adiciona uma tupla individual 't' à relação 'r' usando a operação de união. A tupla deve ter valores para todos os atributos da relação.

Inserção de Múltiplas Tuplas

$r \leftarrow r \cup E$

Insere o resultado de uma expressão de consulta 'E' na relação 'r'. Útil para copiar dados de outras relações ou inserir resultados de consultas complexas.

A operação de inserção utiliza a **união de conjuntos** para adicionar novas tuplas. É importante garantir que as tuplas inseridas tenham valores compatíveis com o esquema da relação, respeitando tipos de dados e restrições de integridade.

Expressões de Inserção:

$$\text{conta} \leftarrow \text{conta} \cup \{(\text{'A-9732'}, \text{'Perryridge'}, 1200)\}$$

$$\text{conta} \leftarrow \text{conta} \cup \sigma_{\text{saldo}>2000}(\text{conta\_temporaria})$$

EXERCÍCIO #1

Encontre Todos os Empréstimos de Mais de US$ 1200

Esquema da Relação

emprestimo (numeroEmprestimo, nomeAgencia, quantia)

Objetivo

Recuperar todos os registros completos de empréstimos cuja quantia seja superior a US$ 1200.

Solução em Álgebra Relacional

Para recuperar **todos os atributos** dos empréstimos:

$$\sigma_{\text{quantia}>1200}(\text{emprestimo})$$

Ou, para recuperar **apenas atributos específicos** (número e agência) dos empréstimos que satisfazem a condição:

$$\pi_{\text{numeroEmprestimo, nomeAgencia}}(\sigma_{\text{quantia}>1200}(\text{emprestimo}))$$

As expressões acima utilizam o operador de **seleção (σ)** para filtrar as tuplas da relação 'emprestimo', retornando apenas aquelas onde o atributo 'quantia' é maior que 1200. A segunda opção adiciona o operador de **projeção (π)** para selecionar as colunas desejadas.

Resultado Esperado

O resultado incluirá todas as colunas (numeroEmprestimo, nomeAgencia, quantia) para cada empréstimo que satisfaça a condição, permitindo uma visão completa dos empréstimos de maior valor.

EXERCÍCIO #2

Encontre o Número de Empréstimo para Quantias Maiores que US$ 1200

01

Selecionar Empréstimos

Primeiro, aplicamos a seleção para filtrar empréstimos com quantia > 1200

02

Projetar Atributo

Em seguida, projetamos apenas a coluna numeroEmprestimo do resultado

Solução Completa

$$\pi_{\text{numeroEmprestimo}}(\sigma_{\text{quantia}>1200}(\text{emprestimo}))$$

Esta expressão combina dois operadores: **seleção (σ)** seguida de **projeção (π)**.

Diferença do Exercício #1

Ao contrário do exercício anterior, aqui retornamos **apenas** os números dos empréstimos, não todas as informações. A projeção elimina colunas desnecessárias do resultado.

EXERCÍCIO #3

Clientes com Empréstimo, Conta, ou Ambos

Este exercício demonstra o uso da operação de **união** para combinar resultados de duas relações distintas.

Esquemas das Relações

depositante (nomeCliente, numeroConta)

tomador (nomeCliente, numeroEmprestimo)

Solução

$$\pi_{\text{nomeCliente}}(\text{depositante}) \cup \pi_{\text{nomeCliente}}(\text{tomador})$$

Projetamos os nomes de clientes de cada relação separadamente e então aplicamos a **união (∪)** para combinar os conjuntos.

Clientes Depositantes

Extraímos nomes de clientes que têm contas através da relação depositante

Clientes Tomadores

Extraímos nomes de clientes que têm empréstimos através da relação tomador

União dos Conjuntos

A união garante que cada cliente apareça apenas uma vez, mesmo que tenha tanto conta quanto empréstimo

EXERCÍCIO #4

Clientes com Empréstimo E Conta no Banco

Diferente do exercício anterior que usava união, aqui precisamos da **interseção** para encontrar clientes que satisfazem ambas as condições simultaneamente.

Esquemas das Relações

depositante (nomeCliente, numeroConta)

tomador (nomeCliente, numeroEmprestimo)

Abordagem

Precisamos identificar nomes que aparecem em **ambas** as relações, ou seja, clientes que são simultaneamente depositantes e tomadores de empréstimo.

Solução

$$\pi_{\text{nomeCliente}}(\text{depositante}) \cap \pi_{\text{nomeCliente}}(\text{tomador})$$

A operação de **interseção (∩)** retorna apenas os nomes que aparecem em ambos os conjuntos, garantindo que o cliente tenha tanto conta quanto empréstimo.

Conceito-chave: União (∪) = "ou", Interseção (∩) = "e"

EXERCÍCIO #5

Clientes com Empréstimo e Suas Quantias

Este exercício demonstra o uso da operação de **junção natural** para combinar informações de múltiplas relações relacionadas.

1

Relação Emprestimo

Contém numeroEmprestimo e quantia

2

Junção Natural

Une as relações pelo atributo comum numeroEmprestimo

3

Relação Tomador

Contém nomeCliente e numeroEmprestimo

4

Projeção

Seleciona apenas nomeCliente e quantia

Esquemas

emprestimo (numeroEmprestimo, nomeAgencia, quantia)

tomador (nomeCliente, numeroEmprestimo)

Solução Completa

$$\pi_{\text{nomeCliente, quantia}}(\text{tomador} \bowtie \text{emprestimo})$$

A junção natural ($\bowtie$) combina automaticamente as tuplas onde numeroEmprestimo é igual em ambas as relações, e a projeção seleciona apenas os atributos desejados.

EXERCÍCIO #6

Calcular Crédito Disponível para Clientes

Esquema da Relação

infoCredito (nomeCliente, limite, saldoCredito)

Objetivo

Descobrir quanto mais cada cliente pode gastar, ou seja, calcular a diferença entre o limite de crédito e o saldo atual utilizado.

Solução com Operador de Atribuição

$$\pi_{\text{nomeCliente, limite - saldoCredito}}(\text{infoCredito})$$

Esta expressão usa **projeção** com uma **expressão aritmética** para calcular o crédito disponível (limite - saldoCredito) para cada cliente.

R$ 5.000

Limite Exemplo

Limite de crédito total do cliente

R$ 2.300

Saldo Usado

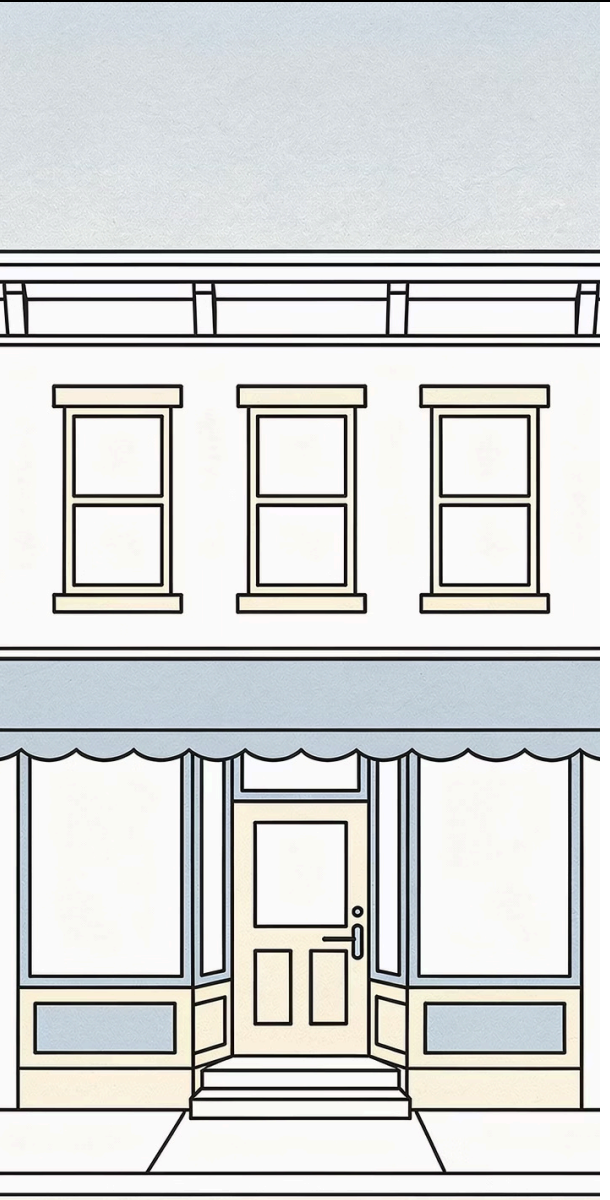
Valor já utilizado do crédito

R$ 2.700

Disponível

Valor que ainda pode ser gasto

A álgebra relacional permite realizar operações aritméticas diretamente nos atributos durante a projeção, facilitando cálculos como este sem necessidade de criar atributos intermediários.



EXERCÍCIO #7

Clientes com Contas em Todas as Agências de Brooklyn

Este é um exemplo clássico de aplicação da **operação divisão**, expressando uma consulta com quantificação universal ("para todo").

Agências em Brooklyn

$\pi_{\text{nomeAgencia}}(\sigma_{\text{cidadeAgencia} = \text{"Brooklyn"}}(\text{agencia}))$

Primeiro identificamos todas as agências localizadas em Brooklyn

Contas por Cliente

$\pi_{\text{nomeCliente, nomeAgencia}}(\text{depositante} \bowtie \text{conta})$

Depois obtemos pares (cliente, agência) através da junção

Operação Divisão

Aplicamos a divisão para encontrar clientes associados a todas as agências de Brooklyn

Esquemas

agencia (nomeAgencia, cidadeAgencia, ativo)

conta (numeroConta, nomeAgencia, saldo)

depositante (nomeCliente, numeroConta)

Interpretação

A divisão garante que apenas clientes com contas em **todas** as agências de Brooklyn sejam retornados, não apenas em algumas delas.

EXERCÍCIO #8

Excluir Todas as Contas da Agência Perryridge

Esquema da Relação

conta (numeroConta, nomeAgencia, saldo)

Objetivo

Remover todos os registros de contas que pertencem à agência Perryridge do banco de dados.

Solução usando Exclusão

$$\text{conta} \leftarrow \text{conta} - \sigma_{\text{nomeAgencia} = \text{"Perryridge"}}(\text{conta})$$

Esta operação utiliza o **operador de atribuição (←)** combinado com a **diferença de conjuntos (-)** para remover as tuplas filtradas pela seleção.

Importante: A operação remove permanentemente os registros. O resultado é uma nova versão da relação conta sem as contas da agência Perryridge.

01

Identificar Contas

Seleção identifica todas as tuplas onde nomeAgencia = "Perryridge"

02

Aplicar Diferença

Subtrai o conjunto de contas identificadas da relação original

03

Atribuir Resultado

A relação conta recebe o resultado sem as tuplas removidas

EXERCÍCIO #9

Inserir Nova Conta no Banco de Dados

Este exercício demonstra como adicionar um novo registro ao banco de dados usando a operação de **inserção**.

Esquema da Relação

conta (numeroConta, nomeAgencia, saldo)

Informações a Inserir

- **Cliente:** Smith
- **Número da Conta:** A-973
- **Agência:** Perryridge
- **Saldo:** US$ 1200

Solução

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

A operação usa a **união (∪)** para adicionar a nova tupla à relação existente. A tupla contém os valores na ordem dos atributos do esquema.

Observação: Note que o nome do cliente (Smith) não aparece na tupla porque não faz parte do esquema da relação *conta*. Essa informação seria armazenada na relação *depositante*.

Preparar Tupla

Organizar os valores de acordo com o esquema: (numeroConta, nomeAgencia, saldo)

Aplicar União

Adicionar a nova tupla à relação usando o operador de união

Atualizar Relação

A relação conta agora inclui o novo registro com a conta A-973

Referências

Elmasri & Navathe

Fundamentals of Database Systems

Pearson, 2016

Referência abrangente sobre fundamentos de sistemas de bancos de dados, cobrindo aspectos teóricos e práticos.

Korth, Sudarshan & Silberschatz

Database System Concepts

McGraw-Hill, 2019

Texto fundamental que serviu como base para a maioria dos exemplos apresentados nesta apresentação.

Özsu & Valduriez

Principles of Distributed Database Systems

Springer Nature, 2019

Obra especializada em sistemas de bancos de dados distribuídos, essencial para compreensão avançada.

Nota: Os conceitos e exemplos apresentados baseiam-se principalmente na literatura clássica de sistemas de bancos de dados, em especial *Database System Concepts* e *Fundamentals of Database Systems*.