Álgebra Relacional e SQL

Banco de Dados

Profa. Dra. Cristina Dutra de Aguiar Ciferri

Álgebra Relacional

- Maneira teórica de se manipular o banco de dados relacional
- Linguagem de consulta procedural
  - usuários especificam os dados necessários e como obtê-los
- Consiste de um conjunto de operações
  - entrada: uma ou duas relações
  - saída: uma nova relação resultado

Operações

- Fundamentais
  - seleção
  - projeção
  - produto cartesiano
  - renomear
  - união
  - diferença de conjuntos
- Adicionais
  - intersecção de conjuntos
  - junção natural
  - divisão
  - agregação

SQL DML

```
SELECT <lista de atributos e funções>
FROM <lista de tabelas>
[ WHERE predicado ]
[ GROUP BY <atributos de agrupamento> ]
[ HAVING <condição para agrupamento> ]
[ ORDER BY <lista de atributos> ] ;
```

Álgebra Relacional: Seleção

- Seleciona tuplas que satisfaçam à condição de seleção

$$\sigma_{\text{condição\_seleção}} (\text{relação argumento})$$

- pode envolver operadores de comparação (=, >, ≥, <, ≤, ≠)
- pode combinar condições usando-se $\wedge$, $\vee$, $\neg$

- relação
- resultado de alguma operação da álgebra relacional

Álgebra Relacional: Seleção

cliente (nro_cli, nome_cli, end_cli, saldo, cod_vend)

nro_cli	nome_cli	end_cli	saldo	cod_vend
1	Márcia	Rua X	100,00	1
2	Cristina	Avenida 1	10,00	1
3	Manoel	Avenida 3	234,00	1
4	Rodrigo	Rua X	137,00	2

Álgebra Relacional: Projeção

- Projeta as colunas solicitadas (i.e. produz um subconjunto vertical)

$$\pi_{lista\_atributos} (\text{relação argumento})$$

- lista de atributos
- os atributos são separados por vírgula

- relação
- resultado de alguma operação da álgebra relacional

Álgebra Relacional: Projeção

cliente (nro_cli, nome_cli, end_cli, saldo, cod_vend)

nro_cli	nome_cli	end_cli	saldo	cod_vend
1	Márcia	Rua X	100,00	1
2	Cristina	Avenida 1	10,00	1
3	Manoel	Avenida 3	234,00	1
4	Rodrigo	Rua X	137,00	2

Álgebra Relacional: Produto Cartesiano

- Combina tuplas de duas relações
  - relações não precisam ter atributos comum
- Tuplas da relação resultante
  - todas as combinações de tuplas possíveis entre as relações participantes

relação argumento 1 × relação argumento 2

- relação
- resultado de alguma operação da álgebra relacional

Relações Cliente e Vendedor

cliente (nro_cli, nome_cli, end_cli, saldo, cod_vend)

nro_cli	nome_cli	end_cli	saldo	cod_vend
1	Márcia	Rua X	100,00	1
2	Cristina	Avenida 1	10,00	1
3	Manoel	Avenida 3	234,00	1
4	Rodrigo	Rua X	137,00	2

vendedor (cod_vend, nome_vend)

cod_vend	nome_vend
1	Adriana
2	Roberto

Cliente × Vendedor

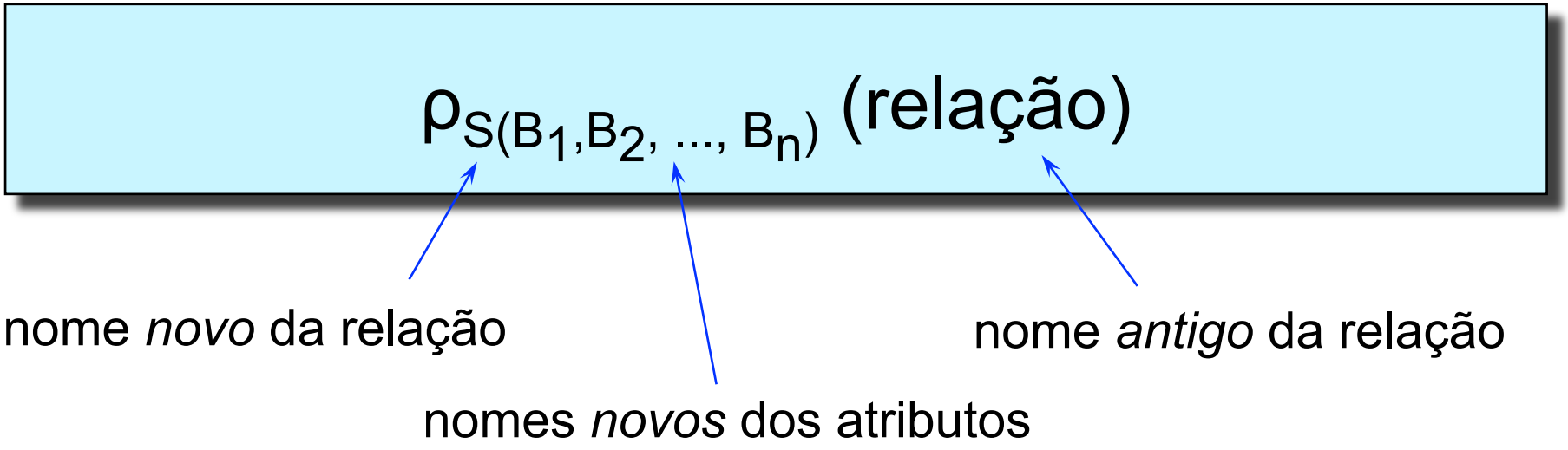
nro_cli	nome_cli	end_cli	saldo	cliente. cod_vend	vendedor. cod_vend	nome_vend
1	Márcia	Rua X	100,00	1	1	Adriana
1	Márcia	Rua X	100,00	1	2	Roberto
2	Cristina	Avenida 1	10,00	1	1	Adriana
2	Cristina	Avenida 1	10,00	1	2	Roberto
3	Manoel	Avenida 3	234,00	1	1	Adriana
3	Manoel	Avenida 3	234,00	1	2	Roberto
4	Rodrigo	Rua X	137,00	2	1	Adriana
4	Rodrigo	Rua X	137,00	2	2	Roberto

grau: número de atributos de cliente + número de atributos de vendedor

número de tuplas: número de tuplas de cliente * número de tuplas de vendedor

Álgebra Relacional: Renomear

- Renomeia
  - nome da relação
  - nomes dos atributos da relação
  - nome da relação e nomes dos atributos



Álgebra Relacional: Renomear

- Exemplos
  - $\rho_{comprador}$ (cliente)
  - $\rho_{(código, nome, rua, saldo, vendedor)}$ (cliente)
  - $\rho_{comprador\ (código, nome, rua, saldo, vendedor)}$ (cliente)
- Observação
  - indicada para ser utilizada quando uma relação é usada mais do que uma vez para responder à consulta

SQL: SELECT-FROM-WHERE

```
SELECT <lista de atributos>
FROM <lista de tabelas>
[WHERE condições de seleção]
```

SQL	Álgebra Relacional
SELECT	projeção
FROM	produto cartesiano
WHERE	seleção

Cláusula ORDER BY

- Ordena as tuplas que aparecem no resultado de uma consulta
  - asc (padrão): ordem ascendente
  - desc: ordem descendente
- Ordenação pode ser especificada em vários atributos
  - a ordenação referente ao primeiro atributo é prioritária. Se houver valores repetidos, então é utilizada a ordenação referente ao segundo atributo, e assim por diante

Cláusula AS

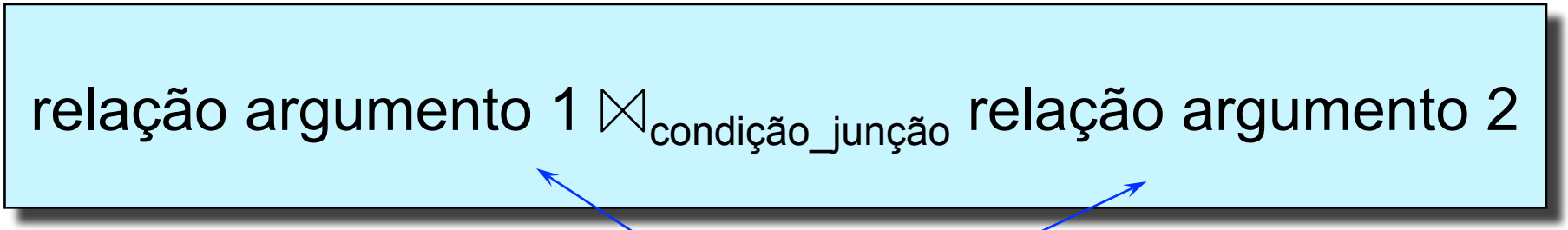
- Renomeia
  - atributos
    - deve aparecer na cláusula SELECT
    - útil para a visualização das respostas na tela
  - relações
    - deve aparecer na cláusula FROM
    - útil quando a mesma relação é utilizada mais do que uma vez na mesma consulta
- Sintaxe
  - nome_antigo AS nome_novo

Álgebra Relacional: Junção

- Concatena tuplas relacionadas de duas relações
- Passos:
  - forma um produto cartesiano das relações
  - faz uma seleção forçando igualdade sobre os atributos que aparecem nas relações

Álgebra Relacional: Junção

- Sintaxe



- relação
- resultado de alguma operação da álgebra relacional

Junção (Exemplo)

nro_cli	nome_cli	end_cli	saldo	cliente. cod_vend	vendedor. cod_vend	nome_vend
1	Márcia	Rua X	100,00	1	1	Adriana
1	Márcia	Rua X	100,00	1	2	Roberto
2	Cristina	Avenida 1	10,00	1	1	Adriana
2	Cristina	Avenida 1	10,00	1	2	Roberto
3	Manoel	Avenida 3	234,00	1	1	Adriana
3	Manoel	Avenida 3	234,00	1	2	Roberto
4	Rodrigo	Rua X	137,00	2	1	Adriana
4	Rodrigo	Rua X	137,00	2	2	Roberto

- Passo 1:
  - formar um produto cartesiano das relações

Junção (Exemplo)

nro_cli	nome_cli	end_cli	saldo	cliente. cod_vend	vendedor. cod_vend	nome_vend
1	Márcia	Rua X	100,00	1	1	Adriana
1	Márcia	Rua X	100,00	1	2	Roberto
2	Cristina	Avenida 1	10,00	1	1	Adriana
2	Cristina	Avenida 1	10,00	1	2	Roberto
3	Manoel	Avenida 3	234,00	1	1	Adriana
3	Manoel	Avenida 3	234,00	1	2	Roberto
4	Rodrigo	Rua X	137,00	2	1	Adriana
4	Rodrigo	Rua X	137,00	2	2	Roberto

- Passo 2:
  - fazer uma seleção forçando igualdade sobre os atributos que aparecem nas relações

Junção (Exemplo)

nro_cli	nome_cli	end_cli	saldo	cliente. cod_vend	vendedor. cod_vend	nome_vend
1	Márcia	Rua X	100,00	1	1	Adriana
2	Cristina	Avenida 1	10,00	1	1	Adriana
3	Manoel	Avenida 3	234,00	1	1	Adriana
4	Rodrigo	Rua X	137,00	2	2	Roberto

SQL: Junção (Primeiras Versões)

- Cláusulas SELECT e WHERE
  - especificam atributos com mesmo nome usando o nome da relação e o nome do atributo (nome_relação.nome_atributo)
- Cláusula FROM
  - possui mais do que uma relação
- Cláusula WHERE
  - inclui as condições de junção (igualdade sobre os atributos que aparecem nas relações)

Junção (Exemplo)

```
SELECT nro_cli, nome_cli, end_cli,
       saldo, vendedor.cod_vend,
       nome_vend
FROM cliente, vendedor
WHERE cliente.cod_vend =
       vendedor.cod_vend
```

SQL-92: Junção

```
SELECT nro_cli, nome_cli, end_cli,
       saldo, vendedor.cod_vend,
       nome_vend
FROM cliente JOIN vendedor ON
     cliente.cod_vend =
     vendedor.cod_vend
```

Álgebra e SQL: Junção

- [INNER] JOIN
  - R ⋈ S
    - somente as tuplas de R que têm tuplas correspondentes em S – e vice-versa – aparecem no resultado
- LEFT [OUTER] JOIN
  - R ⟕ S
    - mantém cada tupla de R na tabela de junção
    - preenche com valores nulos as tuplas de S que não correspondem à coluna de junção em R

Álgebra e SQL: Junção

- RIGHT [OUTER] JOIN
  - R ⟖ S
    - mantém cada tupla de S na tabela de junção
    - preenche com valores nulos as tuplas de R que não correspondem à coluna de junção em S
- FULL [OUTER]JOIN
  - R ⟗ S
    - mantém cada tupla de R e de S na tabela de junção
    - preenche com valores nulos as tuplas que não correspondem à coluna de junção

[INNER] JOIN

R

A	B	C
1	a	x
2	b	y
3	a	y
4	c	y

S

A	D
1	d
2	d
5	e

R ⋈S

R.A	S.A	B	C	D
1	1	a	x	d
2	2	b	y	d

LEFT [OUTER] JOIN

R

A	B	C
1	a	x
2	b	y
3	a	y
4	c	y

S

A	D
1	d
2	d
5	e

R ⟕ S

R.A	S.A	B	C	D
1	1	a	x	d
2	2	b	y	d
3	Null	a	y	Null
4	Null	c	y	Null

RIGHT [OUTER] JOIN

R

A	B	C
1	a	x
2	b	y
3	a	y
4	c	y

S

A	D
1	d
2	d
5	e

R ⟖ S

R.A	S.A	B	C	D
1	1	a	x	d
2	2	b	y	d
Null	5	Null	Null	e

FULL [OUTER]JOIN

R

A	B	C
1	a	x
2	b	y
3	a	y
4	c	y

S

A	D
1	d
2	d
5	e

R ⟗ S

R.A	S.A	B	C	D
1	1	a	x	d
2	2	b	y	d
3	Null	a	y	Null
4	Null	c	y	Null
Null	5	Null	Null	e

Operações sobre Conjuntos

- Unem duas relações
- Operações
  - união
  - intersecção
  - diferença
- Características
  - atuam sobre relações compatíveis
  - eliminam tuplas duplicadas da relação resultado

Duas relações são compatíveis se:
- possuem o mesmo grau
- seus atributos possuem os mesmos domínios (os domínios dos i-ésimos atributos de cada relação são os mesmos)

Álgebra Relacional: Operações sobre Conjuntos

- União ($R \cup S$)
  - gera uma relação que contém todas as tuplas pertencentes a R, a S, ou a ambas R e S
- Intersecção ($R \cap S$)
  - gera uma relação que contém todas as tuplas pertencentes tanto a R quanto a S
- Diferença ($R - S$)
  - gera uma relação que contém todas as tuplas pertencentes a R que não pertencem a S

SQL: Operações sobre Conjuntos

SQL	Álgebra Relacional
UNION	∪
INTERSECT	∩
MINUS	–

- Observação
  - operações oferecidas dependem do SGBD

Exemplo

- Liste os nomes dos clientes que possuem nomes iguais aos nomes de vendedores.

```
SELECT nome_cli
FROM cliente
INTERSECT
SELECT nome_vend
FROM vendedor
```

SQL: Subconsultas Aninhadas

- Subconsulta
  - expressão SELECT ... FROM ... WHERE ... aninhada dentro de outra consulta
- Aplicações mais comuns
  - testes para membros de conjuntos
  - comparações de conjuntos
  - cardinalidade de conjuntos
- Observação
  - a mesma consulta SQL pode ser escrita de diversas maneiras

Membros de um Conjunto

- IN
  - testa se um atributo ou uma lista de atributos é membro do conjunto
- NOT IN
  - verifica a ausência de um membro em um conjunto
- Conjunto:
  - coleção de valores produzidos por uma cláusula SELECT ... FROM ... WHERE ...

Exemplo

- Liste os números dos clientes que têm nome igual ao nome de um vendedor.

```
SELECT nro_cli
FROM cliente
WHERE nome_cli IN
    (SELECT nome_vend
     FROM vendedor)
```

Cardinalidade de Conjuntos

- EXISTS
  - ... WHERE EXISTS (lista)
  - a condição é verdadeira quando a lista (resultado de uma consulta) não for vazia
- NOT EXISTS
  - ... WHERE NOT EXISTS (lista)
  - a condição é verdadeira quando a lista for vazia

Exemplo

- Liste os números dos clientes que têm nome igual ao nome de um vendedor.

```
SELECT nro_cli
FROM cliente
WHERE EXISTS
    (SELECT *
     FROM vendedor
     WHERE cliente.nome_cli =
                vendedor.nome_vend)
```

Álgebra Relacional: Divisão

- Divisão de duas relações R e S
  - todos os valores de um atributo de R que fazem referência a todos os valores de um atributo de S
- Utilizada para consultas que incluam o termo para todos ou em todos

Álgebra Relacional: Exemplo

- Liste os números dos clientes que já foram atendidos por todos os vendedores.

R: atende

nro_cli	cod_vend
9	12
1	04
1	66
4	03
5	11
8	04
8	74

S: π_{cod_vend}(vendedor)

cod_vend
66
04

R ÷ S

nro_cli
1

cliente (<u>nro_cli</u>, nome_cli, end_cli, saldo)

atende (<u>nro_cli</u>, <u>cod_vend</u>)

vendedor (<u>cod_vend</u>, nome_vend)

SQL: Exemplo

- Liste os números dos clientes que já foram atendidos por todos os vendedores.

```
SELECT nro_cli
FROM cliente
WHERE NOT EXISTS
(   (SELECT cod_vend
      FROM vendedor)
      MINUS
    (SELECT cod_vend
      FROM atende
      WHERE cliente.nro_cli = atende.nro_cli)
)
```

Álgebra Relacional: Agregação

- Permite a utilização de funções de agregação

- lista de atributos de agrupamento
- os atributos são separados por vírgula

- relação
- resultado de alguma operação da álgebra relacional

SQL: GROUP BY-HAVING

SELECT <lista de atributos e **funções**>
FROM <lista de tabelas>
[WHERE predicado]
[GROUP BY <atributos de agrupamento>]
[HAVING <condição para agrupamento>]
[ORDER BY <lista de atributos>] ;

Funções de Agregação

- Funções
  - Média: AVG( )
  - Mínimo: MIN( )
  - Máximo: MAX( )
  - Total: SUM( )
  - Contagem: COUNT( )
- Observação
  - DISTINCT: não considera valores duplicados
  - ALL: inclui valores duplicados

Funções de Agregação

- Características
  - recebem uma coleção de valores como entrada
  - retornam um único valor
- SQL: Entrada
  - sum( ) e avg( ): conjunto de números
  - demais funções: tipos de dados numéricos e não-numéricos

Funções de Agregação

vinho (vinho_id, nome_vinho, tipo_vinho, preço, vinícola_id)

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1
15	Daniela	branco	250,00	2
27	Eduarda	branco	150,00	2
48	Fernanda	tinto	7,00	2
13	Gabriela	tinto	397,00	3
12	Helena	branco	333,00	3

Exemplos

- Qual a *média* dos preços?
  - $\xi_{AVG(preço)}$ (vinho)
  - SELECT AVG (preço)
    FROM vinho

217,125

- Qual o vinho mais barato e qual o vinho mais caro?
  - $\xi_{MIN(preço), MAX(preço)}$ (vinho)
  - SELECT MIN (preço), MAX (preço)
    FROM vinho

7,00, 397,00

Exemplos

- *Quantos* vinhos existem na relação vinho?
  - $\xi_{\text{COUNT (vinho\_id)}}$ (vinho)
  - SELECT COUNT (vinho_id)
    FROM vinho

8

- Quantos tipos de vinho *diferentes* existem na relação vinho?
  - $\xi_{\text{COUNT-DISTINCT (tipo\_vinho)}}$ (vinho)
  - SELECT COUNT (DISTINCT tipo_vinho
    FROM vinho

3

SQL: Cláusula GROUP BY

- Funcionalidade
  - permite aplicar uma função de agregação não somente a um conjunto de tuplas, mas também a um grupo de conjunto de tuplas
- Grupo de conjunto de tuplas
  - conjunto de tuplas que possuem o mesmo valor para os atributos de agrupamento
- Semântica da respostas
  - atributos de agrupamento no GROUP BY também devem aparecer no SELECT

Exemplo

- Qual o preço mais alto e a *média* dos preços *por tipo de vinho*?

ρ (tipo_vinho, “maior preço”, “preço médio”)

($_{\text{tipo\_vinho}}$ ξ $_{\text{MAX (preço), AVG (preço)}}$ (vinho))

```
SELECT tipo_vinho,
        MAX (preço) AS “maior preço”,
        AVG (preço) AS “preço médio”
FROM vinho
GROUP BY tipo_vinho
```

SQL: Solução do Exemplo

– As tuplas da tabela vinho são divididas em grupo, cada grupo contendo o mesmo tipo de valor para o atributo de agrupamento tipo_vinho

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1
15	Daniela	branco	250,00	2
27	Eduarda	branco	150,00	2
48	Fernanda	tinto	7,00	2
13	Gabriela	tinto	397,00	3
12	Helena	branco	333,00	3

SQL: Solução do Exemplo

- Considerações adicionais
  - a função MAX e a função AVG são aplicadas a cada grupo de tuplas separadamente
  - a cláusula SELECT inclui somente os atributos de agrupamento e as funções a serem aplicadas a cada grupo de tuplas
  - o comando SELECT pode possuir cláusula WHERE de qualquer complexidade

SQL: Solução do Exemplo

tipo_vinho	maior preço	preço médio
branco	333	233,25
rosê	300	300
tinto	397	168

SQL: Cláusula HAVING

- Funcionalidade
  - permite especificar uma condição de seleção para grupos, melhor do que para tuplas individuais
- Resposta
  - recupera os valores para as funções somente para aqueles grupos que satisfazem à condição imposta na cláusula HAVING

SQL: Exemplo

- Qual o preço mais alto e a *média* dos preços *por tipo de vinho*, para médias de preços superiores a R$200,00

```
SELECT tipo_vinho, MAX (preço), AVG (preço)
FROM vinho
GROUP BY tipo_vinho
HAVING AVG (preço) > 200
```

tipo_vinho	max(preço)	avg(preço)
branco	333	233,25
rosê	300	300

SQL: Processamento da Consulta

- Passos
  - aplica-se o predicado que aparece na cláusula WHERE
  - coloca-se as tuplas que satisfazem a cláusula WHERE em grupos por meio da cláusula GROUP BY
  - aplica-se a cláusula HAVING a cada grupo
  - remove-se os grupos que não satisfazem o predicado da cláusula HAVING
  - exibe-se as colunas listadas na cláusula SELECT