- SQL –
Linguagem de Manipulação de Dados

Laboratório de Bases de Dados

Profa. Dra. Cristina Dutra de Aguiar Ciferri

SELECT

```
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

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

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
  - as relações não precisam possuir atributos em comum
- Tuplas da relação resultante
  - todas as combinações de tuplas possíveis entre as relações participantes

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

SELECT-FROM-WHERE (SQL)

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

- Idéia:
  - concatenar tuplas relacionadas de duas relações em tuplas únicas
- Passos:
  - formar um produto cartesiano das relações
  - 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
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 (Primeiras Versões)

- Não representa junção explicitamente
- 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

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
```

JOIN: SQL-92

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

JOIN (Álgebra e SQL)

- [INNER] JOIN
  - $R \bowtie S$
    - somente as tuplas de R que têm tuplas correspondentes em S – e vice-versa – aparecem no resultado
- LEFT [OUTER] JOIN
  - $R \mathbin{⟕} S$
    - 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

JOIN (Álgebra e SQL)

- 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

Operações sobre Conjuntos (SQL)

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
```

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)
```

Comparação de Conjuntos

- SOME
  - ... WHERE ano_vinho > SOME (lista)
  - a condição é verdadeira quando ano_vinho for maior que algum dos resultados presentes na lista (resultado de uma consulta)
  - outros operadores
    - [< | <= | >= | > | = | <>]
- ANY
  - palavra-chave sinônimo

Exemplo

- Liste os nomes dos clientes que têm saldo superior ao saldo de algum cliente que mora na ‘Rua X’

```
SELECT nome_cli
FROM cliente
WHERE saldo > SOME
      ( SELECT saldo
        FROM cliente
        WHERE end_cli = ‘Rua X’ );
```

Comparação de Conjuntos

- ALL
  - ... WHERE ano_vinho > ALL (lista)
  - a condição é verdadeira quando ano_vinho for maior que todos os resultados presentes na lista (resultado de uma consulta)
  - outros operadores
    - [< | <= | >= | > | = | <>]

Exemplo

- Liste os nomes dos clientes que têm saldo superior ao saldo de todos os clientes que moram na ‘Rua X’

```
SELECT nome_cli
FROM cliente
WHERE saldo > ALL
      ( SELECT saldo
        FROM cliente
        WHERE end_cli = ‘Rua X’ );
```

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

Exemplo (Álgebra Relacional)

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

R: $\pi_{nro\_cli,\ cod\_vend}(atende)$

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

S: $\pi_{cod\_vend}(vendedor)$

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)

Exemplo (SQL)

- 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)
)
```