Bases de Dados

Módulo 09b: Consultas básicas em SQL

Prof. André Bruno de Oliveira

16/04/24 09:58

Ordenação de resultados

Consultas básicas em SQL – Ordenação de resultados

- A linguagem SQL oferece solução para retornar as linhas de uma consulta numa ordem desejada. Há muitas situações que uma lista precisa estar ordenada atender uma necessidade real. Por exemplo, a lista de aprovados no concurso usa o critério de pontuação para exibir os candidatos numa ordem de classificação.
- A cláusula ORDER BY é de uso opcional com o comando SELECT e permite especificar uma lista de atributos separados por vírgula para indicar o critério de importância na ordenação. A sintaxe SQL define que seu uso deve ser no final do comando SELECT.
- Veja este exemplo, as linhas estão ordenadas de modo que os preços estão na ordem crescente.

```
SELECT * FROM Laptop
  WHERE screen > 15
ORDER BY price;
```

	model	speed	ram	hd	screen	price
1	2003	1.8	512	60	15.4	549
2	2008	1.6	1024	100	15.4	900
3	2002	1.73	1024	80	17	949
4	2006	2	2048	80	15.4	1700
5	2010	2	2048	160	15.4	2300
6	2005	2.16	1024	120	17	2500
7	2001	2	2048	240	20.1	3673

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - O exemplo seleciona as telas com tamanho maior do que 15 polegadas ordenadas pela coluna price (preço). Os laptops com preços menores aparecem primeiro. É possível visualmente fazer uma avaliação inicial dos dados e sugerir que as melhores configurações estejam no final da lista com maior preço.
  - Sem o uso do ORDER BY os resultados retornados após a execução de um comando SELECT não seguem um critério explícito de ordenação. Sem o uso da ordenação SGBD retorna as linhas numa ordem que seja o mais rápido para realizar a execução da consulta.

```
SELECT * FROM Laptop
WHERE screen > 15
ORDER BY price;
```

;

	model	speed	ram	hd	screen	price
1	2003	1.8	512	60	15.4	549
2	2008	1.6	1024	100	15.4	900
3	2002	1.73	1024	80	17	949
4	2006	2	2048	80	15.4	1700
5	2010	2	2048	160	15.4	2300
6	2005	2.16	1024	120	17	2500
7	2001	2	2048	240	20.1	3673

Consultas básicas em SQL – Ordenação de resultados

- **ORDER BY e Álgebra Relacional Básica**
- Na álgebra relacional “básica”, apresentada nas últimas aulas, não existe o operador equivalente ao ORDER BY. Ele existe apenas na álgebra relacional “estendida”, que não é coberta em nosso curso.

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - O exemplo utiliza a cláusula ORDER BY com mais de uma coluna.
  - A consulta retorna todos os registros de Laptop ordenados de maneira ascendente pela quantidade de memória (primeiro critério) e de maneira ascendente pelo tamanho do hard disk (como segundo critério). Não há limite para o número de colunas a serem especificadas na cláusula ORDER BY.

```
SELECT * FROM Laptop

ORDER BY ram, hd;
```

	model	speed	ram	hd	screen	price
1	2003	1.8	512	60	15.4	549
2	2004	2	512	60	13.3	1150
3	2009	1.6	512	80	14.1	680
4	2002	1.73	1024	80	17	949
5	2008	1.6	1024	100	15.4	900
6	2005	2.16	1024	120	17	2500
7	2007	1.83	1024	120	13.3	1429
8	2006	2	2048	80	15.4	1700
9	2010	2	2048	160	15.4	2300
10	2001	2	2048	240	20.1	3673

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - O exemplo utiliza a cláusula ORDER BY com uma ordenação **descendente**.
  - A consulta retorna todos os registros da tabela Laptop por ordem descendente de preço. A palavra-chave DESC é utilizada na cláusula ORDER BY para esta finalidade.

```
SELECT * FROM Laptop
    ORDER BY price DESC;
```

	model	speed	ram	hd	screen	price
1	2001	2	2048	240	20.1	3673
2	2005	2.16	1024	120	17	2500
3	2010	2	2048	160	15.4	2300
4	2006	2	2048	80	15.4	1700
5	2007	1.83	1024	120	13.3	1429
6	2004	2	512	60	13.3	1150
7	2002	1.73	1024	80	17	949
8	2008	1.6	1024	100	15.4	900
9	2009	1.6	512	80	14.1	680
10	2003	1.8	512	60	15.4	549

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - A consulta com ORDER BY combina as ordenações ascendente e descendente.
  - A consulta retorna todos os registros de Laptop ordenados de maneira descendente pela quantidade de memória como primeiro critério e de maneira ascendente pelo tamanho do hard disk como segundo critério. A palavra-chave ASC ao lado de "hd" poderia ser omitida, já que a cláusula ORDER BY classifica os resultados de maneira ascendente por padrão.

```
SELECT * FROM Laptop
ORDER BY ram DESC, hd ASC;
```

	model	speed	ram	hd	screen	price
1	2006	2	2048	80	15.4	1700
2	2010	2	2048	160	15.4	2300
3	2001	2	2048	240	20.1	3673
4	2002	1.73	1024	80	17	949
5	2008	1.6	1024	100	15.4	900
6	2005	2.16	1024	120	17	2500
7	2007	1.83	1024	120	13.3	1429
8	2003	1.8	512	60	15.4	549
9	2004	2	512	60	13.3	1150
10	2009	1.6	512	80	14.1	680

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - A consulta com ORDER BY usa a ordenação por posição relativa de um campo.
  - Os SGBD também permitem utilizar a cláusula ORDER BY para ordernar resultados pela posição relativa em um resultado, onde o primeiro campo do resultado tem a posição relativa 1, o campo seguinte 2, e assim por diante.
  - Veja que o resultado da consulta está ordenado por "hd", pois esse é o segundo campo especificado no SELECT (de maneira análoga ORDER BY 3 ordenaria os resultados pelo campo "price" e ORDER BY 1 por "model").
  - É preciso ter muito cuidado com este tipo de ordenação, pois uma mudança que altere a disposição da lista de colunas implica que a ordenação continue sendo feita pela segunda coluna.

```
SELECT model, hd, price FROM Laptop ORDER BY 2;
```

	model	hd	price
1	2003	60	549
2	2004	60	1150
3	2002	80	949
4	2006	80	1700
5	2009	80	680
6	2008	100	900
7	2005	120	2500
8	2007	120	1429
9	2010	160	2300
10	2001	240	3673

Consultas básicas em SQL – Ordenação de resultados

- Considere o seguinte exemplo:
  - Exemplo de ordenação por um campo não selecionado pelo SELECT.
  - Esta consulta retorna o modelo, velocidade e memória de todos os laptops classificados de maneira ascendente pelo preço. Observe, entretanto, que o campo "price" (critério de ordenação) não é uma coluna retornada pela instrução SQL. (Dica: execute novamente o SQL selecionando também o campo "price", para confirmar que a ordenação foi realizada de forma correta).

```
SELECT model, speed, ram
FROM Laptop ORDER BY price;
```

	model	speed	ram
1	2003	1.8	512
2	2009	1.6	512
3	2008	1.6	1024
4	2002	1.73	1024
5	2004	2	512
6	2007	1.83	1024
7	2006	2	2048
8	2010	2	2048
9	2005	2.16	1024
10	2001	2	2048

Consultas básicas em SQL – Ordenação de resultados

- Definição do SELECT com o uso do ORDER BY.
  - De acordo com os exemplos vistos, podemos incluir o ORDER BY na definição de sintaxe do SELECT.
  - As cláusulas opcionais estão entre colchetes.

SELECT [DISTINCT] * | col_1 [AS $apelido_1$],..., col_n [AS $apelido_n$]

FROM tabela

[WHERE $condição_1$,..., $condição_n$]

[ORDER BY col_1,..., col_n]

- **Atenção: Referências a Apelidos de Colunas**
  - Na maioria dos SGBDs, os apelidos de colunas podem ser referenciados em uma cláusula ORDER BY, mas não podem ser referenciados em uma cláusula WHERE.

OPERAÇÕES COM NULL

Consultas básicas em SQL – Operações com NULL

- Esta seção apresentará os operadores IS NULL e IS NOT NULL, utilizados em condições associadas em cláusulas WHERE para tratar valores nulos.
- Para poder executar os exemplos, inseriremos um produto do tipo “printer” (impressora) na base de dados com o preço nulo. Para tal, execute os comandos INSERT abaixo. Primeiro é preciso inserir na tabela *Product* e depois em *Printer*, para que não ocorra violação de chave estrangeira.

```
INSERT INTO Product VALUES ('H',3010,'printer');
INSERT INTO Printer VALUES (3010,0,'laser',NULL);
```

Consultas básicas em SQL – Operações com NULL

- O conceito de valor Nulo (ou NULL) é utilizado em banco de dados relacionais para representar a ausência de informação sobre um determinado campo. Detalhando melhor, se um campo de uma linha contém valor nulo, isto indica que seu conteúdo é desconhecido, inexistente, ou não-aplicável.
- É importante entender que nulo não é a mesma coisa que zero. Zero é um número! Considere, por exemplo, uma tabela que armazene os dados de funcionários de uma empresa. Se a coluna "número de filhos" de um funcionário armazena o valor 0, isso indica que este não possui filhos (possui 0 filhos).
- Se a coluna armazena o valor NULL, isto significa que não é sabido se o funcionário possui filhos ou não. De maneira análoga, também é importante ressaltar que o valor nulo não é a mesma coisa que espaço em branco, já que um espaço em branco representa um caractere.
- Esta seção apresentará os operadores IS NULL e IS NOT NULL, utilizados em condições associadas em cláusulas WHERE para tratar valores nulos. Para poder executar os exemplos, inseriremos um produto do tipo "printer" (impressora) na base de dados com o preço nulo. Para tal, execute os comandos INSERT abaixo. Primeiro é preciso inserir na tabela Product e depois em Printer, para que não ocorra violação de chave estrangeira.

Consultas básicas em SQL – Operações com NULL

- Considere o seguinte exemplo:
  - Exemplo de SELECT WHERE com operador IS NULL.
  - O operador IS NULL deve ser utilizado sempre que se desejar avaliar se um campo é nulo. Não funciona fazer o teste utilizando o operador "=" você precisa usar IS NULL. No exemplo acima, estão sendo selecionados apenas os dados das impressoras cujo valor do campo "price" seja nulo.
  - A consulta retorna apenas um registro, exatamente correspondente ao modelo 3010, recém-inserido.

```
SELECT * FROM Printer
WHERE price IS NULL;
```

	model	color	type	price
1	3010	0	laser	*NULL*

Consultas básicas em SQL – Operações com NULL

- Considere o seguinte exemplo:
  - Exemplo de SELECT WHERE com operador IS NOT NULL.
  - O operador IS NOT NULL é utilizado quando é preciso avaliar se um campo não é nulo. Esta consulta seleciona as impressoras cuja coluna "price" não tem valor nulo. Não funciona fazer o teste utilizando o operador **"<>"** você precisa usar **IS NOT NULL**.

```
SELECT * FROM Printer
WHERE price IS NOT NULL;
```

	model	color	type	price
1	3001	1	ink-jet	99
2	3002	0	laser	239
3	3003	1	laser	899
4	3004	1	ink-jet	120
5	3005	0	laser	120
6	3006	1	ink-jet	100
7	3007	1	laser	200

Consultas básicas em SQL – Operações com NULL

- Considere o seguinte exemplo:
  - Exemplo de SELECT WHERE onde a linha com preço nula não faz parte do resultado.
  - A consulta 1 retorna os valores de model e price quando o preço é superior a 200. A consulta 2 retorna estes valores quando o preço é menor ou igual a 200.
  - Observe que nenhuma das duas consultas retornou a impressora com o número de modelo 3010. Por que? A teoria dos bancos relacionais interpreta NULL como valor "desconhecido" (unknown), "inexistente" ou "não aplicável". Espaço em branco e zero são valores conhecidos. Como o valor é desconhecido, o SGBD não consegue responder se price (preço) é igual, menor ou maior do que o valor comparado.

1. SELECT model, price FROM Printer WHERE price > 200

	model	price
1	3002	239
2	3003	899

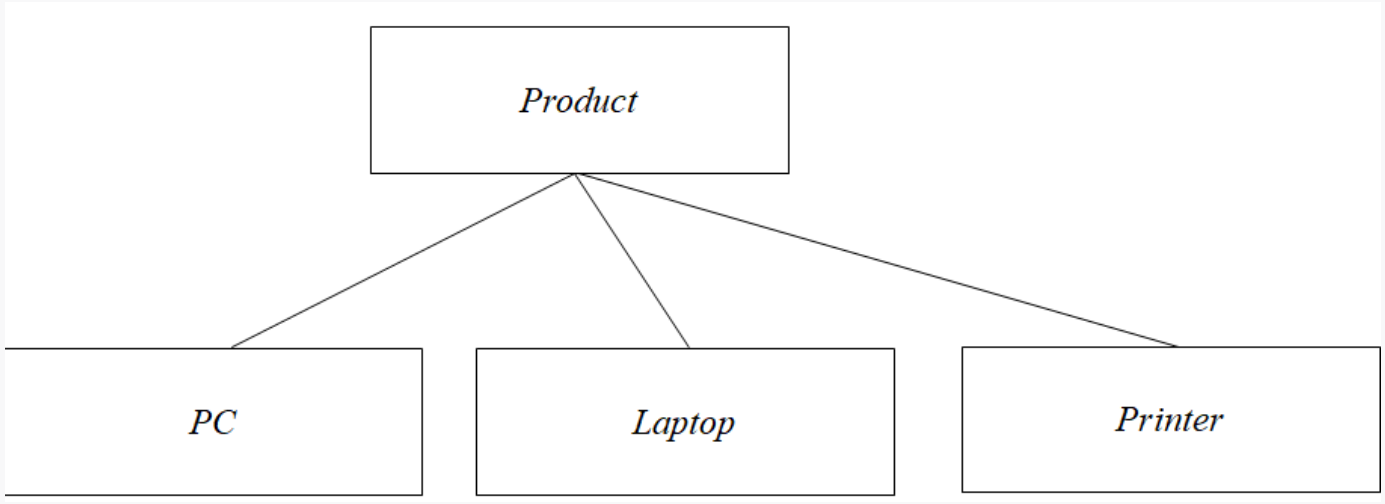
2. ELECT model, price FROM Printer WHERE price <= 200

	model	price
1	3001	99
2	3004	120
3	3005	120
4	3006	100
5	3007	200

Consultas básicas em SQL – Exercícios Propostos 3

1. Crie uma consulta para recuperar a chave primária, tipo e preço de todas as impressoras com preço nulo ou preço abaixo de US$ 200,00.

2. Modifique a consulta acima para retornar os resultados ordenados por preço (descendente) e tipo (ascendente), excluindo as impressoras com valor nulo no preço.



(1)

```
SELECT model, type, price
FROM Printer
WHERE price IS NULL OR price < 200
```

(2)

```
SELECT model, type, price
FROM Printer
WHERE price < 200 / *(mesmo que price is not null and price < 200)*/
ORDER BY price DESC, type
```

Operações de Conjunto

UNIÃO DE CONSULTAS

Consultas básicas em SQL – Operações $R \cup S$

- Considere o seguinte exemplo:
  - Exemplo de **UNION** de consultas.
  - O operador UNION foi utilizado para retornar os modelos e preços dos computadores que estão na tabela PC adicionados aos modelos e preços dos computadores da tabela Laptop.
  - De modo análogo a Álgebra Relacional, a utilização do operador UNION requer que os resultados dos dois (ou mais) SELECT envolvidos tenham os campos do mesmo tipo (neste caso "model" é do tipo INT e "price" do tipo NUM em ambas as tabelas PC e Laptop) e na mesma ordem.

```
SELECT model, price FROM PC
UNION
SELECT model, price FROM Laptop;
```

Consultas básicas em SQL – Operações *R* U *S*

- Considere o seguinte exemplo:

```
SELECT model, price FROM PC
UNION
SELECT model, price FROM Laptop
```

	model	price
1	1001	2114
2	1002	995
3	1003	478
4	1004	649
5	1005	630
6	1006	1049
7	1007	510
8	1008	770
9	1009	650
10	1010	770
11	1011	959
12	1012	649
13	1013	529
14	2001	3673
15	2002	949
16	2003	549
17	2004	1150
18	2005	2500
19	2006	1700
20	2007	1429
21	2008	900
22	2009	680
23	2010	2300

Consultas básicas em SQL – Operações R U S

- Considere o seguinte exemplo:
  - Exemplo de **UNION** com mais de duas consultas.

```
SELECT model, price FROM PC
UNION
SELECT model, price FROM Laptop
UNION
SELECT model, price FROM Printer
```

	model	price
1	1001	2114
2	1002	995
3	1003	478
4	1004	649
5	1005	630
6	1006	1049
7	1007	510
8	1008	770
9	1009	650
10	1010	770
11	1011	959
12	1012	649
13	1013	529
14	2001	3673
15	2002	949
16	2003	549
17	2004	1150
18	2005	2500
19	2006	1700
20	2007	1429
21	2008	900
22	2009	680
23	2010	2300
24	3001	99
25	3002	239
26	3003	899
27	3004	120
28	3005	120
29	3006	100
30	3007	200
31	3010	*NULL*

Expressão equivalente em Álgebra Relacional

$\pi_{model,price}(PC) \cup \pi_{model,price}(Laptop) \cup \pi_{model,price}(Printer)$

Consultas básicas em SQL – Operações *R* U *S*

- Considere o seguinte exemplo:
  - Exemplo de **união** com uso de **WHERE** e **ORDER BY .**
  - A cláusula ORDER BY pode ser usada uma única vez após o último SELECT. Já a cláusula WHERE é preciso usar para cada SELECT.
  - Vejam que todos os PC com preço inferior a US$ 800,00, todos os laptops com preço inferior a US$ 1000 e todas as impressoras com preço inferior a US$ 200 são recuperados e o resultado final está ordenado por preço.

```
SELECT model, price FROM PC WHERE price < 800
UNION
SELECT model, price FROM Laptop WHERE price < 1000
UNION
SELECT model, price FROM Printer WHERE price < 200
ORDER BY price
```

	model	price
1	3001	99
2	3006	100
3	3004	120
4	3005	120
5	1003	478
6	1007	510
7	1013	529
8	2003	549
9	1005	630
10	1004	649
11	1012	649
12	1009	650
13	2009	680
14	1008	770
15	1010	770
16	2008	900
17	2002	949

Consultas básicas em SQL – Operações *R* U *S*

- Considere os seguintes exemplos:
  - Exemplos de **UNION** e **UNION ALL**.
  - Veja que no resultado da consulta de esqueda com **UNION** não há linhas repetidas e na diereita, com o uso do **UNIAL ALL** há linhas repetidas.
  - Há situações que as linhas repetidas são necessárias, assim a SQL permite que o usuário tome esta decisão.

```
SELECT ram, hd FROM PC
UNION
SELECT ram, hd FROM Laptop
ORDER BY ram, hd
```

	ram	hd
1	512	60
2	512	80
3	512	250
4	1024	80
5	1024	100
6	1024	120
7	1024	160
8	1024	200
9	1024	250
10	1024	320
11	2048	80
12	2048	160
13	2048	240
14	2048	250
15	2048	300

```
SELECT ram, hd FROM PC
UNION ALL
SELECT ram, hd FROM Laptop
ORDER BY ram, hd
```

	ram	hd
1	512	60
2	512	60
3	512	80
4	512	80
5	512	80
6	512	250
7	512	250
8	1024	80
9	1024	100
10	1024	120
11	1024	120
12	1024	160
13	1024	200
14	1024	250
15	1024	250
16	1024	250
17	1024	320
18	2048	80
19	2048	160
20	2048	160
21	2048	240
22	2048	250
23	2048	300

Consultas básicas em SQL – Operações $R \cap S$

- Considere o seguinte exemplo:
  - Exemplo faz uso do operador INTERSECT.
  - O operador INTERSECT retorna a **interseção** de dois ou mais conjuntos de resultados. Assim como na união, cada conjunto de resultados é definido por um comando SELECT. Se um registro existe em ambos os conjuntos ele será incluído no resultado final. Neste exemplo, apenas as combinações de valores de "ram" e "hd" comuns às tabelas PC e Laptop são levadas para o resultado final.

```
SELECT ram, hd FROM PC
INTERSECT
SELECT ram, hd FROM Laptop
```

	ram	hd
1	512	80
2	2048	160

Expressão equivalente em Álgebra Relacional

$\pi_{ram,hd}(PC) \cap \pi_{ram,hd}(Laptop)$

Consultas básicas em SQL – Operações *R* - *S*

- Considere o seguinte exemplo:
  - Exemplo faz uso do operador **EXCEPT**.
  - O operador EXCEPT é utilizado para retornar todas as linhas do primeiro conjunto de resultados (definido pelo primeiro SELECT) e então remover deste resultado todas as linhas do segundo conjunto de resultados (definido pelo segundo SELECT). Equivale ao operador de diferença da Álgebra Relacional. IMPORTANTE: em alguns SGBDs, o operador EXCEPT possui outro nome, como por exemplo MINUS.

```
SELECT ram, hd FROM PC
EXCEPT
SELECT ram, hd FROM Laptop
```

	ram	hd
1	512	250
2	1024	160
3	1024	200
4	1024	250
5	1024	320
6	2048	250
7	2048	300

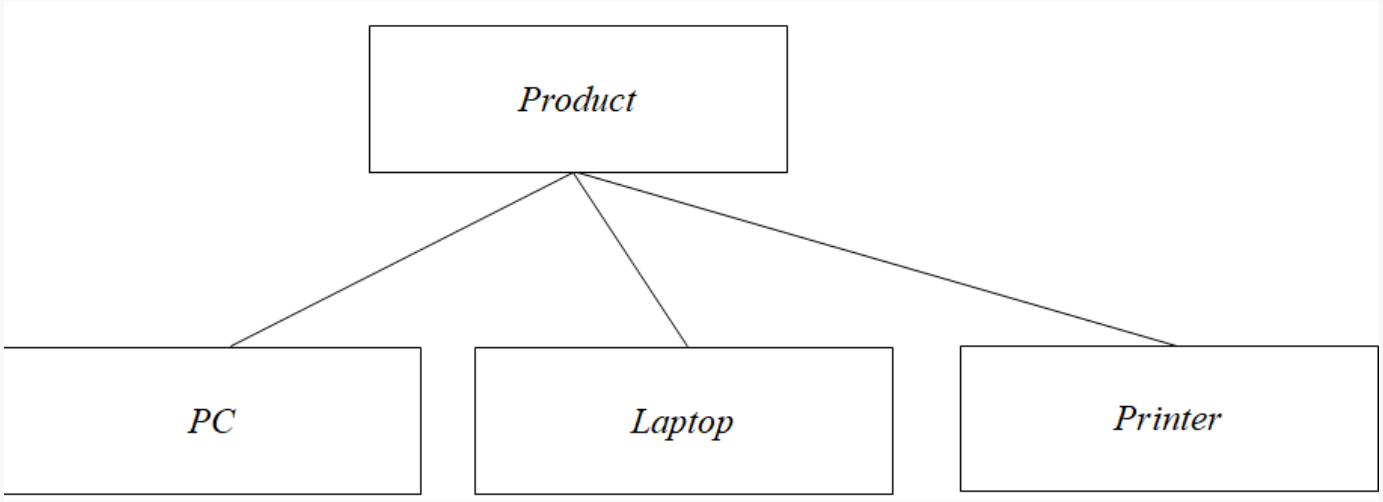
Expressão equivalente em Álgebra Relacional

$\pi_{ram,hd}(PC) - \pi_{ram,hd}(Laptop)$

Consultas básicas em SQL – Exercícios Propostos 4

- Elabore instruções SELECT com os operadores UNION, UNION ALL, INTERSECT ou EXCEPT para resolver as seguintes consultas:

1. Encontrar todos os tamanhos de hard disk distintos, considerando as tabelas Laptop e PC. Eliminar os valores repetidos do resultado final.
2. Encontrar todos os tamanhos de hard disk comuns às tabelas Laptop e PC. Ordenar os resultados de forma descendente.
3. Encontrar todos os fabricantes que produzem laptops, mas não PC.



(1)

```
SELECT hd FROM PC
UNION
SELECT hd FROM Laptop
```

(2)

```
SELECT hd FROM PC
INTERSECT
SELECT hd FROM Laptop
ORDER BY hd
```

(3)

```
SELECT maker FROM Product WHERE type='laptop'
EXCEPT
SELECT maker FROM Product WHERE type='pc'
```

Operações de Conjunto

Combinando Linhas de Duas ou Mais Tabelas

Consultas básicas em SQL – Combinação de linhas

- Na maioria das situações práticas, será necessário trabalhar com informações contidas em mais de uma tabela de um SGBD relacional.
  - Por exemplo: uma consulta onde deseja-se recuperar todos os fabricantes que vendem laptops com hard disk acima de 100GB.
    - Como a informação sobre o fabricante está na tabela *Product* e o tamanho do hard disk na tabela *Laptop*, é preciso elaborar um comando SELECT que realize o casamento de linhas de *Product* com as linhas de *Laptop*.
    - Conforme já foi visto, existem dois tipos de operações de combinação de linhas das tabelas.
      - Produto Cartesiano: Combina as linhas de duas tabelas de todas as formas possíveis.
      - Junção: Combina de forma seletiva as linhas de duas tabelas. A **junção natural** realiza a combinação baseada em atributos que são comuns às duas tabelas, enquanto a **junção theta** realiza a combinação baseado em qualquer outro tipo de critério.
- O objetivo principal desta seção é mostrar como criar consultas que realizam as operações de combinação de linhas na linguagem SQL.

Consultas básicas em SQL – $R \times S$

- Considere o seguinte exemplo:
  - Exemplo faz uso do SELECT combinando duas tabelas.
  - O produto cartesiano é a operação que combina todos os registros da tabela *Product* com todos da tabela *Pc*. Veja que na SQL o produto cartesiano é representado pela expecificação de tabelas separadas por vírgula na clásula **FROM**. As colunas do resultado são formadas pelos campos da primeira tabela (*Product*) seguido dos campos da segunda tabela (*Pc*). O número de colunas finais é igual a soma das colunas de *Product* adicionadas o número de colunas de *Pc* (3+5=8).

```
SELECT a.*, b.*
FROM Product a, PC b
```

	maker	model	type	model:1	speed	ram	hd	price
1	A	1001	pc	1001	2.66	1024	250	2114
2	A	1001	pc	1002	2.1	512	250	995
3	A	1001	pc	1003	1.42	512	80	478
4	A	1001	pc	1004	2.8	1024	250	649
...	...	...	...	...	...	...	...	...
399	H	3010	printer	1009	2	1024	250	650
400	H	3010	printer	1010	2.8	2048	300	770
401	H	3010	printer	1011	1.86	2048	160	959
402	H	3010	printer	1013	3.06	512	80	529
403	H	3010	printer	1012	2.8	1024	160	649

Consultas básicas em SQL – $R \times S$

- Considere o seguinte exemplo:
  - Exemplo faz uso do SELECT combinando duas tabelas.
  - Observe que em nosso exemplo a tabela *Product* foi apelidada como "a" e *PC* apelidada como "b". Desta forma, torna-se possível referenciar qualquer coluna de *Product* pelo prefixo "a." (ex: a.maker); analogamente, qualquer coluna de *PC* pode ser referenciada como "b." (ex: b.price). Neste exemplo, utilizou-se SELECT a.*, b.*, para recuperar todas as colunas de Product e PC. Como uma coluna de nome "model" existe nas duas tabelas (coluna comum aos dois esquemas), o SQLite exibiu o rótulo "model:1" para identificar a coluna "model" da tabela PC (na prática, cada SGBD utiliza uma forma diferente para remover a ambiguidade de nomes).

```
SELECT a.*, b.*
FROM Product a, PC b
```

	maker	model	type	model:1	speed	ram	hd	price
1	A	1001	pc	1001	2.66	1024	250	2114
2	A	1001	pc	1002	2.1	512	250	995
3	A	1001	pc	1003	1.42	512	80	478
4	A	1001	pc	1004	2.8	1024	250	649
...	...	...	...	...	...	...	...	...
399	H	3010	printer	1009	2	1024	250	650
400	H	3010	printer	1010	2.8	2048	300	770
401	H	3010	printer	1011	1.86	2048	160	959
402	H	3010	printer	1013	3.06	512	80	529
403	H	3010	printer	1012	2.8	1024	160	649

Operações de Conjunto

Junção Natural

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo faz uso do SELECT combinando duas tabelas com o uso do **NATURAL JOIN**.
  - A junção natural equivalente no SQL **JOIN NATURAL** serve para combinar linhas das duas tabelas, neste caso *Product* e *Pc*. O junção natural combina apenas as linhas das duas tabelas que coincidem em quaisquer atributos que são comuns as duas relações (tabelas no SQL), conforme visto nas aulas anteriores.

```
SELECT a.maker, a.model, a.type, b.speed, b.ram, b.hd, b.price
FROM Product a NATURAL JOIN PC b;
```

	maker	model	type	speed	ram	hd	price
1	A	1001	pc	2.66	1024	250	2114
2	A	1002	pc	2.1	512	250	995
3	A	1003	pc	1.42	512	80	478
4	B	1004	pc	2.8	1024	250	649
5	B	1005	pc	3.2	512	250	630
6	B	1006	pc	3.2	1024	320	1049
7	C	1007	pc	2.2	1024	200	510
8	D	1008	pc	2.2	2048	250	770
9	D	1009	pc	2	1024	250	650
10	D	1010	pc	2.8	2048	300	770
11	E	1011	pc	1.86	2048	160	959
12	E	1012	pc	2.8	1024	160	649
13	E	1013	pc	3.06	512	80	529

Expressão AR equivalente
Product $|\times|$ *Pc*

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - O comando SQL utiliza a cláusula NATURAL JOIN para implementar a junção natural entre as tabelas *Product* e *PC*. Ela recupera o fabricante (“maker”), modelo (“model”), tipo de produto (“type”), velocidade (“speed”), memória RAM (“ram”), hard disk (“hd”) e preço (“price”) de todos os PC's da base de dados.
    - As colunas “maker”, “model” e “type” são recuperadas da tabela Product; e as demais colunas de PC.
    - Ao contrário do que ocorre no produto cartesiano, uma linha de *Product* só será “casada” com uma linha de *PC* caso elas coincidam no valor do atributo em comum (neste caso, “model”, que é chave primária em *Product* e chave estrangeira em PC). A resposta da consulta retorna 13 linhas, menor do que se fosse um produto cartesiano (3 x 5=18 linhas).

```
SELECT a.maker, a.model, a.type, b.speed, b.ram, b.hd, b.price
FROM Product a NATURAL JOIN PC b;
```

	maker	model	type	speed	ram	hd	price
1	A	1001	pc	2.66	1024	250	2114
2	A	1002	pc	2.1	512	250	995
3	A	1003	pc	1.42	512	80	478
4	B	1004	pc	2.8	1024	250	649
5	B	1005	pc	3.2	512	250	630
6	B	1006	pc	3.2	1024	320	1049
7	C	1007	pc	2.2	1024	200	510
8	D	1008	pc	2.2	2048	250	770
9	D	1009	pc	2	1024	250	650
10	D	1010	pc	2.8	2048	300	770
11	E	1011	pc	1.86	2048	160	959
12	E	1012	pc	2.8	1024	160	649
13	E	1013	pc	3.06	512	80	529

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo não discrimina as colunas usadas pelo SELECT para fazer NATURAL JOIN.
  - Observe que o SQL retorna model duas vezes e troca o nome para model:1. Este comportamento difere da álgebra relacional, onde o resultado de *Product* |×| Pc contém apenas uma ocorrência do atributo model.

```
SELECT *

FROM Product a NATURAL JOIN PC b;
```

	maker	model	type	model:1	speed	ram	hd	price
1	A	1001	pc	1001	2.66	1024	250	2114
2	A	1002	pc	1002	2.1	512	250	995
3	A	1003	pc	1003	1.42	512	80	478
4	B	1004	pc	1004	2.8	1024	250	649
5	B	1005	pc	1005	3.2	512	250	630
6	B	1006	pc	1006	3.2	1024	320	1049
7	C	1007	pc	1007	2.2	1024	200	510
8	D	1008	pc	1008	2.2	2048	250	770
9	D	1009	pc	1009	2	1024	250	650
10	D	1010	pc	1010	2.8	2048	300	770
11	E	1011	pc	1011	1.86	2048	160	959
12	E	1012	pc	1012	2.8	1024	160	649
13	E	1013	pc	1013	3.06	512	80	529

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **WHERE** e **ORDER BY**.
  - As cláusulas WHERE e ORDER BY podem ser utilizadas normalmente em instruções SQL que possuem produto cartesiano ou junção de tabelas (seja junção natural ou qualquer outro tipo). Neste exemplo, a consulta usa uma restrição para listar apenas o modelo e preço dos PC produzidos pelo fabricante 'E' (cláusula WHERE) e ordenar os resultados de forma ascendente pelo preço do produto (cláusula ORDER BY). As cláusula WHERE e ORDER BY podem ser utilizadas com colunas de ambas as tabelas, sem qualquer tipo de limitação.
  - Tanto para a cláusula WHERE quanto para o ORDER BY é fundamental especificar a origem do atributo (nome ou apelido da tabela seguido de ponto e nome da coluna) caso seja uma variável comum entre as tabelas usadas no FROM.

```
SELECT a.model, b.price
FROM Product a NATURAL JOIN PC b
WHERE a.maker IN ('E')
ORDER BY b.price;
```

	model	price
1	1013	529
2	1012	649
3	1011	959

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **DISTINCT**. **WHERE** e **ORDER BY** sem incluir a coluna de ligação na lista de variáveis do SELECT.
  - Este exemplo envolve uma operação de junção entre as tabelas *Product* e *Laptop*. Para cada fabricante que produz um laptop, apresenta-se os diferentes tamanhos de tela disponíveis em seus modelos de laptop com preço inferior a US$ 2000. Os resultados são ordenados pelo tamanho da tela em ordem decrescente.
  - **IMPORTANTE** notar neste exemplo que o campo model, que conecta as tabelas, não tem a obrigatoriedade de estar na lista de campos do SELECT para ser exibido.

```
SELECT DISTINCT a.maker, b.screen
FROM Product a NATURAL JOIN Laptop b
WHERE b.price < 2000
ORDER BY b.screen DESC;
```

	maker	screen
1	E	17
2	E	15.4
3	A	15.4
4	F	15.4
5	F	14.1
6	A	13.3
7	B	13.3

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **subconsulta**.
  - A consulta retorna o número do modelo, fabricante, tipo de produto e preço de todos os produtos.
  - Aqui está um dos mais poderosos recursos da SQL (também presente na Álgebra Relacional). A instruções SELECT tem complexidade arbitrária através da aplicação de consulta sobre resultados obtidos por outras consultas.

```
SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN (
    SELECT model, price FROM PC
    UNION
    SELECT model, price FROM Laptop
    UNION
    SELECT model, price FROM Printer
) b
```

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **subconsulta**.
  - Neste caso, os parênteses são utilizados para indicar as subconsultas (consultas internas) que serão executadas inicialmente e depois terão seus resultados "consumidos" pela(s) consulta(s) externa(s) (responsáveis por prover o conjunto de resultados final).

```
SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN (
    SELECT model, price FROM PC
    UNION
    SELECT model, price FROM Laptop
    UNION
    SELECT model, price FROM Printer
) b
```

	model	maker	type	price
1	1001	A	pc	2114
2	1002	A	pc	995
3	1003	A	pc	478
4	1004	B	pc	649
5	1005	B	pc	630
6	1006	B	pc	1049
7	1007	C	pc	510
8	1008	D	pc	770
9	1009	D	pc	650
10	1010	D	pc	770
11	1011	E	pc	959
12	1012	E	pc	649
13	1013	E	pc	529
14	2001	E	laptop	3673
15	2002	E	laptop	949
16	2003	E	laptop	549
17	2004	A	laptop	1150
18	2005	A	laptop	2500
19	2006	A	laptop	1700
20	2007	B	laptop	1429
21	2008	F	laptop	900
22	2009	F	laptop	680
23	2010	G	laptop	2300
24	3001	E	printer	99
25	3002	E	printer	239
26	3003	E	printer	899
27	3004	D	printer	120
28	3005	D	printer	120
29	3006	H	printer	100
30	3007	H	printer	200
31	3010	H	printer	NULL

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **subconsulta**.
  - A consulta gera como resultado uma relação (resultado intermediário) contendo os valores das colunas "model" e "price" das tabelas PC, Laptop e Printer. Este resultado intermediário é apelidado de "b". A consulta externa realiza a junção natural de Product com o resultado da consulta interna, retornando uma nova relação (resultado final) que recupera modelo, fabricante, tipo de produto e preço de todos os produtos cadastrados no BD.

```
SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN (
    SELECT model, price FROM PC
    UNION
    SELECT model, price FROM Laptop
    UNION
    SELECT model, price FROM Printer
) b
```

	model	maker	type	price
1	1001	A	pc	2114
2	1002	A	pc	995
3	1003	A	pc	478
4	1004	B	pc	649
5	1005	B	pc	630
6	1006	B	pc	1049
7	1007	C	pc	510
8	1008	D	pc	770
9	1009	D	pc	650
10	1010	D	pc	770
11	1011	E	pc	959
12	1012	E	pc	649
13	1013	E	pc	529
14	2001	E	laptop	3673
15	2002	E	laptop	949
16	2003	E	laptop	549
17	2004	A	laptop	1150
18	2005	A	laptop	2500
19	2006	A	laptop	1700
20	2007	B	laptop	1429
21	2008	F	laptop	900
22	2009	F	laptop	680
23	2010	G	laptop	2300
24	3001	E	printer	99
25	3002	E	printer	239
26	3003	E	printer	899
27	3004	D	printer	120
28	3005	D	printer	120
29	3006	H	printer	100
30	3007	H	printer	200
31	3010	H	printer	NULL

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- Considere o seguinte exemplo:
  - Exemplo usa **NATURAL JOIN** com **subconsulta**.

```
SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN (
    SELECT model, price FROM PC
    UNION
    SELECT model, price FROM Laptop
    UNION
    SELECT model, price FROM Printer
) b
```

	model	maker	type	price
1	1001	A	pc	2114
2	1002	A	pc	995
3	1003	A	pc	478
4	1004	B	pc	649
5	1005	B	pc	630
6	1006	B	pc	1049
7	1007	C	pc	510
8	1008	D	pc	770
9	1009	D	pc	650
10	1010	D	pc	770
11	1011	E	pc	959
12	1012	E	pc	649
13	1013	E	pc	529
14	2001	E	laptop	3673
15	2002	E	laptop	949
16	2003	E	laptop	549
17	2004	A	laptop	1150
18	2005	A	laptop	2500
19	2006	A	laptop	1700
20	2007	B	laptop	1429
21	2008	F	laptop	900
22	2009	F	laptop	680
23	2010	G	laptop	2300
24	3001	E	printer	99
25	3002	E	printer	239
26	3003	E	printer	899
27	3004	D	printer	120
28	3005	D	printer	120
29	3006	H	printer	100
30	3007	H	printer	200
31	3010	H	printer	NULL

Expressão AR equivalente

b := $\pi_{model,price}(PC)$ U $\pi_{model,price}(Laptop)$ U $\pi_{model,price}(Printer)$

Resposta : = $\pi_{a.model,\ a.maker,\ a.type,\ b.price}(Product \,|\times|\, b)$

Consultas básicas em SQL – $R \,|\times|\, S$, $R \bowtie S$

- **IMPORTANTE**:
  - A resolução de consultas na álgebra relacional costuma usar uma sequência de operações, o que facilita a compreensão do resultado final. No caso do SQL, as consultas precisam ser elaboradas como expressão única, mesmo que sejam complexas.
  - Quando uma consulta é muito importante, pode-se utilizar o recurso de criar um visão de dados (*View*) que contenha o resultado de uma seleção para usá-la em outras consultas. Por exemplo, a subconsulta interna pode ser substituída por uma view chamada *Vmodel_preco*.

```
SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN (
        SELECT model, price FROM PC
        UNION
        SELECT model, price FROM Laptop
        UNION
        SELECT model, price FROM Printer
) b
```

```
create view Vmodel_preco as
SELECT model, price FROM PC
UNION
SELECT model, price FROM Laptop
UNION
SELECT model, price FROM Printer;

SELECT a.model, a.maker, a.type, b.price
FROM Product a NATURAL JOIN
Vmode_preco b;
```

Expressão AR equivalente

b := $\pi_{model,price}(PC)$ U $\pi_{model,price}(Laptop)$ U $\pi_{model,price}(Printer)$

Resposta : = $\pi_{model,price}(Product \,|\times|\, b)$

Consultas básicas em SQL – $R \bowtie_C S$

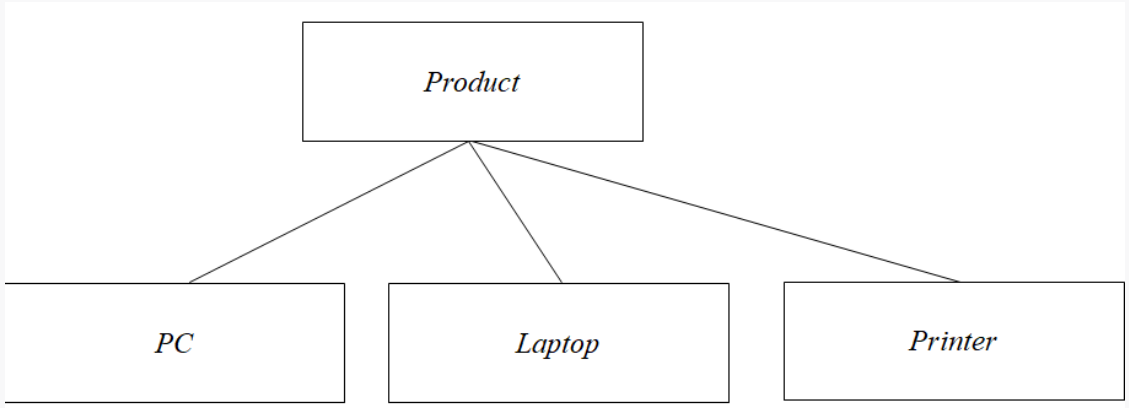
- **Considere o seguinte exemplo:**
  - Exemplo usa junção de tabelas com o uso do **WHERE**.
  - A **junção tetha** da álgebra relacional permite combinar registros de duas tabelas utilizando uma condição qualquer.
  - O resultado de uma operação de junção theta entre duas tabelas é construído em duas etapas:
    - (i) realizar o produto cartesiano entre as tabelas; e
    - (ii) selecionar apenas as tuplas que satisfaçam à condição especificada. Veja que o exemplo realiza o produto cartesiano da tabela PC com ela mesma no trecho: FROM PC a, PC b.
      - Observe que foi preciso apelidar a instância *Pc* como "a" e "b" para viabilizar o uso do resultado do produto cartesiano. A restrição aplicada nas linhas seleciona apenas os registros que os modelos de "a" e "b" sejam diferentes e que os hard disk tenham mesmo tamanho: WHERE a.model < b.model AND a.hd = b.hd.

```
SELECT DISTINCT a.hd
FROM PC a, PC b
WHERE a.model < b.model AND a.hd = b.hd;
```

	hd
1	250
2	80
3	160

Consultas básicas em SQL – Exercícios Propostos 5

1. Crie uma consulta para recuperar os fabricantes que produzem impressoras coloridas. CUIDADO: as tabelas Product e Printer possuem dois atributos com nomes iguais: “model” e “type”.
2. Modifique a consulta acima da seguinte forma: retornar o fabricante, número do modelo e preço de todas as impressoras coloridas.
3. Crie uma consulta para recuperar todos os dados (modelo, velocidade, ram, hd e preço) dos PC que não são produzidos pelos fabricantes A e B. Ordene o resultado por velocidade de modo descendente (primeiro critério) e modelo em ordem ascendente (segundo critério).
4. Encontre os fabricantes e modelos de computadores (PC ou laptops) com velocidade igual ou superior a 2.00 giga-hertz.
5. Encontre o preço do laptop mais barato.
6. Encontre todas as combinações diferentes de pares de modelos de PC. Um par deve ser listado apenas uma vez; ex: listar (i,j) mas não (j,i).
7. Modifique o exercício acima para exibir apenas os pares formados por dois modelos que custem menos de US$ 600,00.



(1)

```
SELECT DISTINCT a.maker
FROM Product a, Printer b
WHERE a.model = b.model
AND b.color = 1
```

ou

```
SELECT DISTINCT  a.maker
FROM Product a NATURAL JOIN
(select model FROM Printer WHERE color=1) b
```

(2)

```
SELECT a.maker, a.model, b.price
FROM Product a, Printer b
WHERE a.model = b.model
AND b.color = 1
```

(3)

```
SELECT a.* FROM PC a NATURAL JOIN Product b
WHERE b.maker NOT IN ('A','B')
ORDER BY speed DESC, a.model
```

(4)

```
SELECT a.maker, a.model
FROM Product a NATURAL JOIN
(
SELECT model, speed FROM PC
UNION
SELECT model, speed FROM Laptop
) b
WHERE b.speed >= 2.0
```

(5)

```
SELECT price FROM Laptop
EXCEPT
Select a.price FROM
Laptop a, Laptop b
WHERE a.price > b.price
```

(6)

```
SELECT a.model, b.model
FROM PC a, PC b
WHERE a.model < b.model
```

(7)

```
SELECT a.model, b.model

FROM PC a, PC b

WHERE a.model < b.model AND a.price < 600 AND b.price < 600
```

Obrigado