Bases de Dados

Módulo 9a: Consultas básicas em SQL e suas equivalências em AR

Prof. André Bruno de Oliveira

12/04/24 11:52

Tópicos

- Preparação dos banco de dados
- Conhecendo a Instrução SELECT
  - DISTINCT
- Restringindo e Ordenando o Conjunto de Resultados
  - WHERE
    - Operadores de comparação
    - Regras de precedência
  - ORDER BY (10b)
  - Trabalhando com o Valor NULO
- Operações de Conjunto
  - Produto Cartesiano
  - Junção Natural
  - Junção Theta

Introdução

- Os SGBD's relacionais são manipulados através de uma linguagem padrão, desenvolvida especialmente para o ambiente relacional, denominada SQL (Structured Query Language).
- A linguagem SQL é composta por um reduzido conjunto de instruções que permitem manipular um banco de dados com diferentes finalidades.
- Considera-se que a SQL seja principal responsável pela imensa popularidade conquistada pelos SGBD relacionais nos últimos 30 anos.
- A SQL é oferecida em praticamente todos os ambientes de programação (Python, Java, PHP, C#, etc.) e está disponível até mesmo em linguagens especialmente direcionadas para estatística e ciência de dados como R e SAS.

Consultas básicas em SQL

- **INTRODUÇÃO**

O objetivo central desta aula é apresentar a forma de realizar o mapeamento ou "tradução" direta das instruções da Álgebra Relacional que aprendemos nas últimas aulas para comandos da SQL.

Será dado ênfase a instrução SELECT, que representa a instrução SQL mais utilizada pelos profissionais da área de estatística ou ciência de dados.

Posteriormente (principalmente na segunda parte deste tópico), serão apresentados outros recursos e formas de utilização desta poderosa instrução do SQL, o SELECT.

Consultas básicas em SQL

INSTRUÇÃO: SELECT

FINALIDADE : Recuperar dados

DESCRIÇÃO: Recuperar registros armazenados em tabelas do banco de dados.

Exemplo: SELECT FROM Regiao

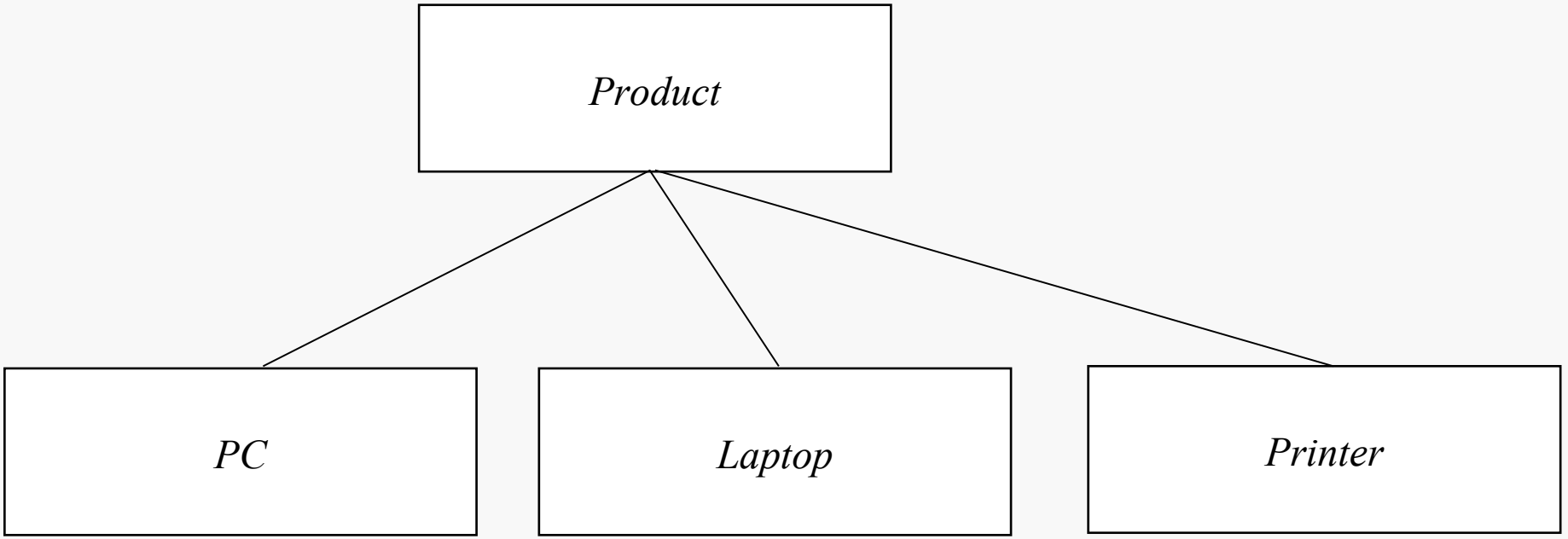
Regiao

sigla_regiao	nome_regiao
NT	NORTE
ND	NORDESTE
CO	CENTRO-OESTE
SD	SUDESTE
SL	SUL

Todas as linhas ou registros da tabela *Regiao* são exibidas numa forma de lista. Isso inclui os atributos no topo de lista organizados em colunas.

Consultas básicas em SQL

Vamos trabalhar com exemplos práticos, para isso será usado o banco de dados Products.db, que está implementado no SQLite com 4 tabelas. O banco de dados foi extraído do capítulo 2 do livro “Database Systems: The Complete Book.



Consultas básicas em SQL

Primeiramente é preciso conhecer a especificação completa de cada tabela.

As seguintes informações são apresentadas para no seguinte formato:

- Atributo (coluna) = nome do atributo;
- Tipo = tipo de dado (TEXT, INT ou NUM);
- PK = se estiver marcado com X, a coluna é chave primária
- NOT NULL = se estiver marcado com X, a coluna é do tipo NOT NULL;
- FK = se estiver marcado com X, a coluna é chave estrangeira.
- Descrição / Comentários = descrição do atributo e informações adicionais.

Consultas básicas em SQL

Product

•Tabela que contém as informações gerais sobre diversos produtos.

- Fabricante (maker),
- número do modelo (model),
- tipo (type) de aparelho ('PC', 'laptop' ou 'printer') de cada produto.

Vamos considerar, por conveniência, que os números de modelo (model) são únicos entre todos os fabricantes e tipos de produto. Todas as demais tabelas possuem chave estrangeira para esta tabela, conforme definido no esquema de relações anteriormente.

Atributo	**Tipo**	**PK**	**NOT NULL**	**FK**	**Descrição**
maker	TEXT		X		Código do fabricante
model	INT	X	X		Número do modelo
type	TEXT		X		Tipo de produto ('PC', 'laptop' ou 'printer')

Consultas básicas em SQL

PC

•Tabela que contém as características de cada modelo de um *personal computer* (PC) computador desktop (conhecido como torre).

Atributo	**Tipo**	**PK**	**NOT NULL**	**FK**	**Descrição**
model	INT	X	X	X	Número do modelo. (FK ref. "model" em Product)
speed	INT	X	X		Velocidade do processador em giga-hertz.
ram	TEXT		X		Quantidade de memória RAM em megabytes
hd	INT		X		Tamanho do hard disk em gigabytes
price	NUM				Preço do produto.

Consultas básicas em SQL

Laptop

•Tabela que contém as características de cada modelo de um computador portátil. O diferencial em relação a tabela *PC* é o campo que armazena o tamanho da tela, *screen*.

Atributo	**Tipo**	**PK**	**NOT NULL**	**FK**	**Descrição**
model	INT	X	X	X	Número do modelo. (FK ref. "model" em Product)
speed	INT	X	X		Velocidade do processador em giga-hertz.
ram	TEXT		X		Quantidade de memória RAM em megabytes
hd	INT		X		Tamanho do hard disk em gigabytes
screen	NUM		X		Tamanho da tela em polegadas.
price	NUM				Preço do produto.

Consultas básicas em SQL

Printer

•Tabela que contém as características de cada modelo de impressora. A impressora do tipo colorida tem o atributo color igual 1 e não colorida zero.

Atributo	Tipo	PK	NOT NULL	FK	Descrição
model	INT	X	X	X	Número do modelo. (FK ref. "model" em Product)
color	BOOLEAN		X		Indica se impressora é colorida (1=SIM, 0=NÃO).
type	TEXT		X		Tipo de impressora ('laser' ou 'ink-jet').
price	NUM				Preço do produto.

Consultas básicas em SQL

- A partir das informações de cada tabela há condições de montar os *scripts* SQL para criar as tabelas e incluir os dados de cada registro ou linha.
- É importante adotar uma estratégia para criar as tabelas e incluir os registros, pois há relações de integridade referencial entre as tabelas do nosso modelo.
- Uma solução comum é criar primeiro a tabela que contém a chave estrangeira da tabela que usa seus dados para criar a restrição de integridade referencial. Neste caso, a tabela *Product* é uma tabela que centraliza as validações de integridade referencial do atributo model usado nas demais tabelas.

Consultas básicas em SQL

- Adotando esta proposta de criar primeiro tabela que contém a chave pai (*parent*) e depois a tabela que é criada a chave estrangeira (tabela dependente). É possível popular as informações das respectivas tabelas sem que haja inconsistência de restrição referencial.
- Por conveniência, podemos chamar a tabela com a chave pai de TABELA PAI e a tabela que contém a FK de TABELA DEPENDENTE.

Consultas básicas em SQL

- A primeira tabela que deve ser criada e preenchida é a *Product*.

```
CREATE TABLE Product
( maker TEXT NOT NULL,
model INT NOT NULL,
type TEXT NOT NULL,
PRIMARY KEY (model)
);
```

Consultas básicas em SQL

- A tabela *PC* é criada com chave primária model referenciando a o atributo model de *Product*. Atenção, o *script* usado neste exemplo para criar as restrições de chave primária e chave estrangeira é único. É importante saber identificar estas palavras chaves de criação de restrição para os casos que você estiver lendo um script criado por outro modelo de banco de dados ou até mesmo as variações do próprio banco de dados, como é o caso no SQLite.

```
CREATE TABLE PC
(model INT NOT NULL,
speed NUM NOT NULL,
ram INT NOT NULL,
hd INT NOT NULL,
price NUM,
PRIMARY KEY (model,SPEED),
FOREIGN KEY (model) REFERENCES Product(model));
```

Script alternativo para o SQLite

```
CREATE TABLE PC (
model INT NOT NULL PRIMARY KEY,
speed NUM NOT NULL,
ram INT NOT NULL,
hd INT NOT NULL,
price NUM,
FOREIGN KEY (model) REFERENCES Product
(model) );
```

Consultas básicas em SQL

- A tabela *Laptop* é criada com chave primária model referenciando ao atributo model de *Product*.

```
CREATE TABLE Laptop
(model INT NOT NULL,
speed NUM NOT NULL,
ram INT NOT NULL,
hd INT NOT NULL,
screen NUM NOT NULL,
price NUM,
PRIMARY KEY (model, speed),
FOREIGN KEY (model) REFERENCES Product(model));
```

Consultas básicas em SQL

- A tabela *Printer* é criada com chave primária model referenciando ao atributo model de *Product*.

```
CREATE TABLE Printer
(model INT NOT NULL,
color BOOLEAN NOT NULL,
type TEXT NOT NULL,
price NUM,
PRIMARY KEY (model,color),
FOREIGN KEY (model) REFERENCES Product(model));
```

Consultas básicas em SQL

- Seguindo a estratégia proposta de preencher primeiro a tabela pai e depois a tabela dependtente, devemos fazer os comando INSERT na tabela *Product*. Veja que o *script* adotado para o INSERT utilizado o modelo tradicional sem declarar os atributos/campos da tabela, além disso é feito um INSERT para cada registro. Lembre-se que vimos o script para inserir múltiplas linhas INSERT INTO Tabela (atributo$_1$, atrubuto$_n$) VALUES (A_1,..A_N), (B_1,..B_N).

```
INSERT INTO Product VALUES('A',1001,'pc');
INSERT INTO Product VALUES('A',1002,'pc');
INSERT INTO Product VALUES('A',1003,'pc');
INSERT INTO Product VALUES('A',2004,'laptop');
INSERT INTO Product VALUES('A',2005,'laptop');
INSERT INTO Product VALUES('A',2006,'laptop');
INSERT INTO Product VALUES('B',1004,'pc');
INSERT INTO Product VALUES('B',1005,'pc');
INSERT INTO Product VALUES('B',1006,'pc');
INSERT INTO Product VALUES('B',2007,'laptop');
INSERT INTO Product VALUES('C',1007,'pc');
INSERT INTO Product VALUES('D',1008,'pc');
INSERT INTO Product VALUES('D',1009,'pc');
INSERT INTO Product VALUES('D',1010,'pc');
INSERT INTO Product VALUES('D',3004,'printer');
INSERT INTO Product VALUES('D',3005,'printer');
```

Consultas básicas em SQL

- INSERT em *Product*

```
INSERT INTO Product VALUES('E',1011,'pc');
INSERT INTO Product VALUES('E',1012,'pc');
INSERT INTO Product VALUES('E',1013,'pc');
INSERT INTO Product VALUES('E',2001,'laptop');
INSERT INTO Product VALUES('E',2002,'laptop');
INSERT INTO Product VALUES('E',2003,'laptop');
INSERT INTO Product VALUES('E',3001,'printer');
INSERT INTO Product VALUES('E',3002,'printer');
INSERT INTO Product VALUES('E',3003,'printer');
INSERT INTO Product VALUES('F',2008,'laptop');
INSERT INTO Product VALUES('F',2009,'laptop')
INSERT INTO Product VALUES('G',2010,'laptop');
INSERT INTO Product VALUES('H',3006,'printer');
INSERT INTO Product VALUES('H',3007,'printer');
```

Consultas básicas em SQL

- INSERT em *Pc*

```
INSERT INTO PC VALUES(1001,2.66,1024,250,2114);
INSERT INTO PC VALUES(1002,2.1,512,250,995);
INSERT INTO PC VALUES(1003,1.42,512,80,478);
INSERT INTO PC VALUES(1004,2.8,1024,250,649);
INSERT INTO PC VALUES(1005,3.2,512,250,630);
INSERT INTO PC VALUES(1006,3.2,1024,320,1049);
INSERT INTO PC VALUES(1007,2.2,1024,200,510);
INSERT INTO PC VALUES(1008,2.2,2048,250,770);
INSERT INTO PC VALUES(1009,2,1024,250,650);
INSERT INTO PC VALUES(1010,2.8,2048,300,770);
INSERT INTO PC VALUES(1011,1.86,2048,160,959);
INSERT INTO PC VALUES(1013,3.06,512,80,529);
INSERT INTO PC VALUES(1012,2.8,1024,160,649);
```

Consultas básicas em SQL

- INSERT EM *Laptop*

```
INSERT INTO Laptop VALUES(2001,2,2048,240,20.1,3673);
INSERT INTO Laptop VALUES(2002,1.73,1024,80,17,949);
INSERT INTO Laptop VALUES(2003,1.8,512,60,15.4,549);
INSERT INTO Laptop VALUES(2004,2,512,60,13.3,1150);
INSERT INTO Laptop VALUES(2005,2.16,1024,120,17,2500);
INSERT INTO Laptop VALUES(2006,2,2048,80,15.4,1700);
INSERT INTO Laptop VALUES(2007,1.83,1024,120,13.3,1429);
INSERT INTO Laptop VALUES(2008,1.6,1024,100,15.4,900);
INSERT INTO Laptop VALUES(2009,1.6,512,80,14.1,680);
INSERT INTO Laptop VALUES(2010,2,2048,160,15.4,2300);
```

- INSERT EM *Printer*

```
INSERT INTO Printer VALUES(3001,1,'ink-jet',99);
INSERT INTO Printer VALUES(3002,0,'laser',239);
INSERT INTO Printer VALUES(3003,1,'laser',899);
INSERT INTO Printer VALUES(3004,1,'ink-jet',120);
INSERT INTO Printer VALUES(3005,0,'laser',120);
INSERT INTO Printer VALUES(3006,1,'ink-jet',100);
INSERT INTO Printer VALUES(3007,1,'laser',200);
```

Conhecendo a Instrução SELECT

Consultas básicas em SQL – SELECT

- Selecionar todas as linhas da tabela *Product*.
  - Script padrão: SELECT * FROM Tabela
    - ***** é usado para listar todos os atributos/colunas da tabela
    - **From** informa de qual tabela as linhas serão selecionadas
- SELECT * FROM Product;
- A expressão equivalente em álgebra relacional para este script é:

  Product

Consultas básicas em SQL – SELECT

- SELECT * FROM Product;

	maker	model	type
1	A	1001	pc
2	A	1002	pc
3	A	1003	pc
4	A	2004	laptop
5	A	2005	laptop
6	A	2006	laptop
7	B	1004	pc
8	B	1005	pc
9	B	1006	pc
10	B	2007	laptop
11	C	1007	pc
12	D	1008	pc
13	D	1009	pc
14	D	1010	pc
15	D	3004	printer
16	D	3005	printer
17	E	1011	pc
18	E	1012	pc
19	E	1013	pc
20	E	2001	laptop
21	E	2002	laptop
22	E	2003	laptop
23	E	3001	printer
24	E	3002	printer
25	E	3003	printer
26	F	2008	laptop
27	F	2009	laptop
28	G	2010	laptop
29	H	3006	printer
30	H	3007	printer

Consultas básicas em SQL – SELECT

- Selecionar colunas específicas da tabela *Product*.
  - Script padrão: SELECT $a_1,...,a_n$ FROM Tabela
    - Uma lista com nomes dos após campos O SELECT deve ser especificada e separada com por vírgulas.
- SELECT model, hd

  FROM Pc;
- A expressão equivalente em álgebra relacional para este script é:

$\pi_{model,hd}(Pc)$

	model	hd
1	1001	250
2	1002	250
3	1003	80
4	1004	250
5	1005	250
6	1006	320
7	1007	200
8	1008	250
9	1009	250
10	1010	300
11	1011	160
12	1013	80
13	1012	160

Observe que as colunas são exibidas na ordem especificada: model e depois pc. As linhas são exibidas na ordem que foram incluídas. Modelo 1012 vem depois do modelo 1013.

Consultas básicas em SQL – SELECT

- Verificamos que a instrução básica do SELECT para recuperar linhas possui duas cláusulas:
  - Script padrão: SELECT * | col_1,...,col_n

    FROM Tabela

    •A primeira cláusula especifica que todas as colunas da tabela sejam exibidas. Note que não é preciso saber os nomes das colunas.

    •A segunda cláusula especifica qual conjunto de colunas se deseja exibir.

- É importante esclarecer que as instruções SQL quando usadas na maioria dos SGBD não fazem distinção entre letras maiúsculas e minúsculas.
- Sobre a organização do comando SELECT há pessoas que preferem por as cláusulas em linhas separadas (como na definição acima) com a única finalidade de melhorar a legibilidade (ou seja: não há problema em colocar a cláusula FROM na mesma linha da cláusula SELECT).

SELECT DISTINCT

Consultas básicas em SQL – SELECT DISTINCT

- **Diferentemente da álgebra relacional as instruções não tem por padrão suprimir as tuplas repetidas**.
  - Por exemplo, a instrução de projeção de um atributo retorna as tuplas distintas, sem repetição.
  - A relação Pc possui 13 tuplas de hd, só que entre elas há somente 6 tuplas com valores distintos para o tamanho do hd. Isto ocorre porque a teoria relacional define que toda relação é um conjunto e por definição um conjunto não pode ter elementos repetidos. Assim, o resultado de uma relação não pode ter tuplas repetidas.

$\pi_{hd}(Pc)$

250
80
320
200
300
160

Consultas básicas em SQL – SELECT DISTINCT

- **Diferença entre AR e SQL**
  - Por questões práticas, o comportamento da maioria das instruções SQL por padrão é de não suprimir linhas mesmo que existam linhas com valores repetidos. O comando SELECT por padrão não elimina linhas duplicadas.
  - Isso ocorre porque os fabricantes de SGBDs preferiram definir a tabela como uma estrutura do tipo *multiset* ou bag ("saco"). Uma tabela é um *multiset* de tuplas enquanto uma relação é um conjunto de tuplas. Um *multiset* é um conjunto que permite elementos repetidos.
  - De modo prático, os SGBD permitem que os usuários decidam se o resultado das consultas devem ter ou não tuplas repetidas. Vejamos o exemplo a seguir.

Consultas básicas em SQL – SELECT DISTINCT

```
SELECT hd
    FROM PC;
```

	hd
1	250
2	250
3	80
4	250
5	250
6	320
7	200
8	250
9	250
10	300
11	160
12	80
13	160

Veja que no resultado do SELECT há linhas diferentes com valores de atributos iguais. É possível identificar, por exemplo, o total de linhas da tabela *Pc*.

Consultas básicas em SQL – SELECT DISTINCT

A supressão das linhas no comando SELECT é feita com o uso da palavra-chave DISTINCT:

```
SELECT DISTINCT hd
        FROM PC;
```

	hd
1	250
2	80
3	320
4	200
5	300
6	160

O DISTINCT retira as tuplas duplicadas do resultado, fazendo com que as linhas repetidas aparecem uma única vez na resposta. O uso do DISTINCT permite realizar um estudo sobre as tuplas diferentes na tabela. Veja que após a execução do comando anterior, o número de linhas reduz e fica mais fácil identificar os hd menor, 80, e o maior, 320.

Consultas básicas em SQL – SELECT DISTINCT

Quando o comando SELECT, com o DISTINCT, é usado com um conjunto de atributos que inclua uma chave única da tabela, a resposta contém somente tuplas distintas: SELECT model, hd FROM Pc. Como model é PK de *Pc* e faz parte da tupla, todas as linhas são distintas. Neste caso, a projeção $\pi_{model,hd}(Pc)$ será igual a seleção $\sigma_{model,hd}(R)$.

	model	hd
1	1001	250
2	1002	250
3	1003	80
4	1004	250
5	1005	250
6	1006	320
7	1007	200
8	1008	250
9	1009	250
10	1010	300
11	1011	160
12	1013	80
13	1012	160

Consultas básicas em SQL – SELECT DISTINCT

DISTINCT: remoção de linhas duplicadas

É importante compreender que o principal objetivo do DISTINCT é remover linhas duplicadas, reproduzindo o mesmo comportamento da álgebra relacional.

I. SELECT type FROM Product

II. SELECT DISTINCT type Product

O comando I retorna todas as linhas contendo o valor do atributo type da tabela *Product*. O comando II retorna um número de linhas menor, pois inclui somente ocorrência de tuplas distintas.

Consultas básicas em SQL – SELECT DISTINCT

DEFINIÇÃO DO SELECT com DISTINCT

- Do que foi visto até agora a definição padrão do SELECT com ou sem DISTINCT fica assim: :

  - Script padrão: SELECT [DISTINCT] | * | a_1,...,a_n FROM Tabela.
  - O DISTINCT pode ser usado tanto para o * quando para uma lista de atributos separadas por vírgula, a_1,...,a_n .
    - SELECT DISTINCT * FROM Tabela
    - SELECT DISTINCT a_1,...,a_n FROM Tabela

Consultas básicas em SQL – SELECT DISTINCT

DISTINCT e contagem de Frequência na tabela.

O comando DISTINCT é útil em situações onde estamos "explorando" uma base de dados e desejamos identificar todos os valores diferentes de uma ou mais colunas. No entanto, ele não serve para computar tabelas de frequência. Ou seja: o comando não determina a frequência com que cada valor ou combinação de valores ocorre. Para computar tabelas de frequência, é preciso trabalhar com funções de grupo – tema muito importante, mas que será abordado em aulas futuras deste curso.

Ex.: O resultado abaixo exibe a frequência de hd distintos existentes na tabela original *Pc*.

	hd	Freq
1	80	2
2	160	2
3	200	1
4	250	6
5	300	1
6	320	1

Consultas básicas em SQL – SELECT DISTINCT

Definição de rótulos (ou "apelidos") para colunas.

- A exibição da consulta usa automaticamente o nome do atributo como rótulo do cabeçalho da tabela resultado. Conforme foi visto na álgebra relacional, há situações que se torna necessário alterar o nome da coluna através da renomeação.
- Isto pode ser feito no comando SELECT através da palavra chave **AS** seguido do nome novo para o qual se deseja mudar o rótulo.

```
SELECT model AS Modelo, maker AS Fabricante, type AS Tipo
FROM Product;
```

Consultas básicas em SQL – SELECT DISTINCT

Definição de rótulos (ou "apelidos") para colunas.

•Os SGBD de modo geral permitem que a clausula **AS** seja omitida, assim o exemplo 1 e 2 produzem o mesmo resultado.

•Alguns SGBD permitem o uso de caracteres especiais como espaço, *, # ou $ no nome do rótulo. Neste caso, é preciso definir o rótulo novo entre aspas duplas.

1. SELECT model **AS** Modelo, maker **AS** Fabricante, type AS Tipo
FROM *Product*;

2. SELECT model Modelo, maker Fabricante, type AS Tipo
FROM *Product*;

	Modelo	Fabricante	Tipo
1	1001	A	pc
2	1002	A	pc
3	1003	A	pc
4	2004	A	laptop
5	2005	A	laptop
6	2006	A	laptop
7	1004	B	pc
	...	...	...
27	2009	F	laptop
28	2010	G	laptop
29	3006	H	printer
30	3007	H	printer

Consultas básicas em SQL – SELECT DISTINCT

Modelo estendido para a sintaxe do comando SELECT básico.

Do que foi visto até agora a o comando SELECT podo ser definido como:

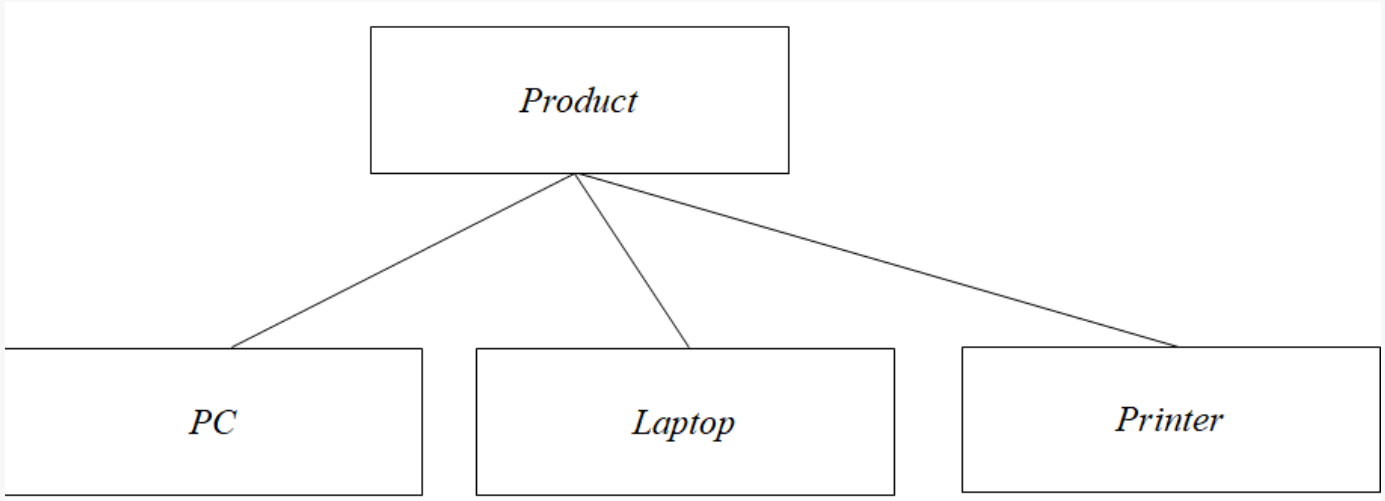
SELECT [DISTINCT] * | Col_1 AS [AS apelido$_1$], **…,**Col_2 [AS apelido$_n$]
FROM *Tabela*;

A expressão de algebra relacional equivalente a esta esta definição em SQL é o seguinte:

$$\sigma_c(R)$$

Consultas básicas em SQL – Exercícios Proposto 1

1. Crie e execute uma consulta para recuperar todas as linhas e colunas da tabela *Printer*.
2. Crie e execute uma consulta para recuperar todas as linhas e colunas da tabela *Printer* Nesta consulta, utilize o rotulo "COLORIDA" para a coluna "color".
3. Por que a coluna "model" não será incluída no resultado do comando SELECT DISTINCT speed, ram FROM PC ?
4. Crie uma consulta para recuperar as colunas "ram" e "hd" (nesta ordem) da tabela *Laptop*.
5. Crie uma consulta para recuperar todas as combinacoes distintas de valores das colunas "ram" e "hd" armazenados em *Lapto.*



(1) SELECT * FROM Printer

(2) SELECT model, color AS COLORIDA, type, price FROM Printer

(3) Porque "model" é a PK da tabela.

Se "model" fosse incluída, não seria possível retornar as combinações distintas de valores de "speed" e "ram"

(4) SELECT ram, hd FROM Laptop

(5) SELECT DISTINCT ram, hd FROM Laptop

Restringindo resultados

Consultas básicas em SQL – Restringindo resultados

- A recuperação de tuplas da tabela geralmente requer algum tipo de critério para que sejam selecionadas somente as linhas de interesse.
  - Cláusula WHERE é utilizada para este propósito. Ela atua como um operador de seleção na SQL.

```
SELECT * FROM Product
WHERE type = 'printer';
```

	maker	model	type
1	D	3004	printer
2	D	3005	printer
3	E	3001	printer
4	E	3002	printer
5	E	3003	printer
6	H	3006	printer
7	H	3007	printer

- A instrução SELECT recupera os produtos cujo valor da coluna "type" seja igual a 'printer' (ou seja: os produtos do tipo impressora).
- A instrução SQL faz uma avaliação de cada tupla e recupera somente as linhas que o conjunto de testes da condição do WHERE tenha como resultado booleano verdade.
- O exemplo usado, testa para cada linha se type = 'printer' é verdadeiro, retornando 7 linhas.

- A expressão equivalente em Álgebra Relacional é:
  $\sigma_{type= 'printer'}(Product)$

Consultas básicas em SQL – Restringindo resultados

- A instrução SQL de consulta permite usar como critério um campo não incluído na lista de campos recuperados.
  - A cláusula WHERE, neste caso, é utilizada com o atributo price e recupera os valores do campo speed .

```
SELECT DISTINCT speed FROM Pc
WHERE price < 900;
```

	speed
1	1.42
2	2.8
3	3.2
4	2.2
5	2
6	3.06

- A expressão equivalente em Álgebra Relacional é:
  $\pi_{speed}(\sigma_{price < 900}(Pc))$

Consultas básicas em SQL – Restringindo resultados

- **Colunas Exibidas *versus* Colunas Testadas**
  - Toda instrução SELECT básica começa com a palavra-chave SELECT seguida do nome de uma ou mais colunas que serão exibidas para o usuário. Depois vem a palavra FROM seguida do nome de uma tabela (as colunas exibidas devem pertencer a esta tabela).
  - Quando é preciso restringir quais linhas devem ser recuperadas, usa-se WHERE com um conjunto de testes. Os testes devem conter as variáveis (atributo ou campo) da tabela especificada no FROM.
  - É importante deixar claro que existe uma independência entre as colunas exibidas e as colunas testadas.
    - Por exemplo, se uma tabela X possui as colunas “a”, “b”, “c”, é possível montar uma instrução que recupere as colunas “a” e “b”, baseada em um teste na coluna “c”.
      - Ex.: SELECT a, b FROM X WHERE c = 1000

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Os operadores lógico do WHERE combinam os resultados de duas ou mais condições.

Operador	Significado
AND	Retorna TRUE se todas as condições avaliadas forem verdadeiras.
OR	Retorna TRUE se ao menos uma das condições avaliadas for verdadeira.
NOT	Retorna TRUE se a condição seguinte for FALSE.

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Este exemplo recupera todos os laptops que possuem velocidade de 2.0, memória de 2048 e hard disk com capacidade superior a 100. Todas as colunas são retornadas (SELECT *).

```
SELECT * FROM Laptop
    WHERE speed = 2.0 AND ram = 2048 AND hd > 100;
```

	model	speed	ram	hd	screen	price
1	2001	2	2048	240	20.1	3673
2	2010	2	2048	160	15.4	2300

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Este exemplo recupera o número do modelo de todas as impressoras laser coloridas.

```
SELECT model FROM Printer
    WHERE color = 1 AND type = 'laser';
```

	model
1	3003
2	3007

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Este exemplo recupera os PC com memória de 2048 ou hard disk de 250.

```
SELECT *
    FROM PC
    WHERE ram = 2048 OR hd = 250;
```

	model	speed	ram	hd	price
1	1001	2.66	1024	250	2114
2	1002	2.1	512	250	995
3	1004	2.8	1024	250	649
4	1005	3.2	512	250	630
5	1008	2.2	2048	250	770
6	1009	2	1024	250	650
7	1010	2.8	2048	300	770
8	1011	1.86	2048	160	959

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Este exemplo recupera todos os produtos que não foram produzidos pelo fabricante 'A'.

```
SELECT * FROM Product
    WHERE (maker <> 'A');
```

	maker	model	type
1	B	1004	pc
2	B	1005	pc
3	B	1006	pc
4	B	2007	laptop
5	C	1007	pc
6	D	1008	pc
7	D	1009	pc
8	D	1010	pc
9	D	3004	printer
10	D	3005	printer
11	E	1011	pc
12	E	1012	pc
13	E	1013	pc
14	E	2001	laptop
15	E	2002	laptop
16	E	2003	laptop
17	E	3001	printer
18	E	3002	printer
19	E	3003	printer
20	F	2008	laptop
21	F	2009	laptop
22	G	2010	laptop
23	H	3006	printer
24	H	3007	printer

Consultas básicas em SQL – Restringindo resultados

- **WHERE - Utilizando operadores lógicos**
  - Este exemplo recupera todos os produtos que não foram produzidos pelo fabricante 'A'.

```
SELECT * FROM Product
    WHERE NOT (maker = 'A');
```

	maker	model	type
1	B	1004	pc
2	B	1005	pc
3	B	1006	pc
4	B	2007	laptop
5	C	1007	pc
6	D	1008	pc
7	D	1009	pc
8	D	1010	pc
9	D	3004	printer
10	D	3005	printer
11	E	1011	pc
12	E	1012	pc
13	E	1013	pc
14	E	2001	laptop
15	E	2002	laptop
16	E	2003	laptop
17	E	3001	printer
18	E	3002	printer
19	E	3003	printer
20	F	2008	laptop
21	F	2009	laptop
22	G	2010	laptop
23	H	3006	printer
24	H	3007	printer

Consultas básicas em SQL – Restringindo resultados

- **DEFINIÇÃO FORMAL DO SELECT COM WHERE**
  - A cláusula WHERE é utilizada para restringir as linhas que serão retornadas de acordo com um ou mais critérios.
  - Trata-se de uma clausula opcional que deve ser usado logo após a lista de tabelas declarada na cláusula FROM. Ainda veremos que a clausula FROM pode conter uma ou mais tabelas sempre que necessário.
  - Quando o WHERE é usado deve haver obrigatoriamente uma condição especificada.

SELECT [DISTINCT] * | Col_1 AS [AS $apelido_1$], …,Col_2 [AS $apelido_n$]

FROM *Tabela* [WHERE condição];

A expressão de algebra relacional equivalente a esta esta definição em SQL é a seguinte:

$\rho_c(\sigma_c(R))$

Consultas básicas em SQL – Operadores de comparação

- **Um conjunto importante de operadores de comparação pode ser usado com o WHERE para formar a condição de seleção das tuplas desejadas.**
- Alguns dos operadores existentes na tabela abaixo já foram usados em exemplos anteriores.

Operador	Significado
=	Igual a
<>	Diferente
>	Maior que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
BETWEEN	Verifica se valor está compreendido entre uma faixa de valores
IN	Verifica se valor pertence a um conjunto de valores
LIKE	Verifica se parte do texto (valor alfanumérico) se encontra numa cadeia de caracteres.

Consultas básicas em SQL – Operadores de comparação

- Este exemplo recupera os produtos que não foram produzidos pelo fabricante A.

```
SELECT *
    FROM Product
    Where maker <> 'A';
```

A expressão de algebra relacional equivalente a esta esta definição em SQL é a seguinte:

$(\sigma_{maker \neq 'A'}(Product))$

maker	model	type
B	1004	pc
B	1005	pc
B	1006	pc
B	2007	laptop
C	1007	pc
D	1008	pc
D	1009	pc
...	...	...
F	2009	laptop
G	2010	laptop
H	3006	printer
H	3007	printer

Consultas básicas em SQL – Operadores de comparação

- Este exemplo recupera os os números de modelo dos PC com preço igual ou inferior a 900 e que possuem memória de pelo menos 1024.

```
SELECT model
    FROM Pc
    Where price <=900 AND ram >=1024;
```

A expressão de algebra relacional equivalente a esta esta definição em SQL é a seguinte:

$(\sigma_{\text{price <=900 AND ram >=1024}}(Pc))$

	model
1	1004
2	1007
3	1008
4	1009
5	1010
6	1012

Consultas básicas em SQL – Regras de precedência

- O uso dos operadores lógico AND, NOT E OR possui regras de precedência quando 2 ou mais operadores estão presentes na mesma instrução SQL. O operador NOT tem precedência maior, seguido do operador AND e do operador OR, respectivamente. Estas regras de precedência podem ser sobrepostas através do uso de parênteses (isto também é válido para a Álgebra Relacional).

Operador	Ordem de avaliação
NOT	1
AND	2
OR	3

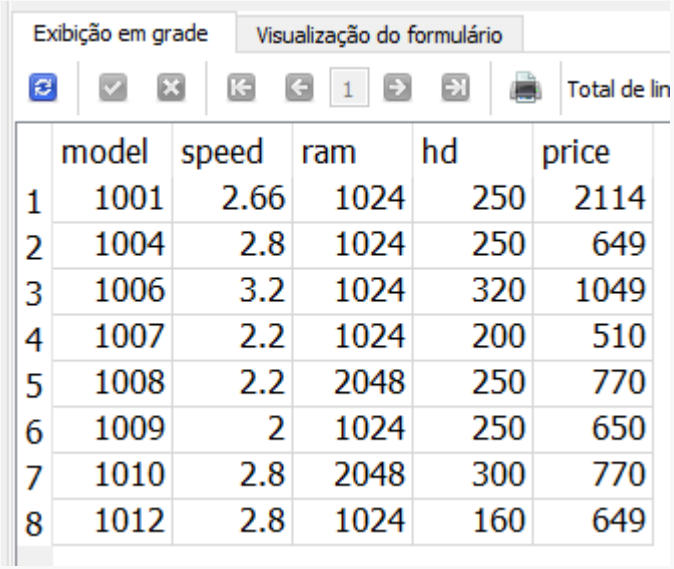
Consultas básicas em SQL – Regras de precedência

- Considere o seguinte exemplo:
  - Suponha que um usuário deseja obter uma lista de PCs que possuam memória de 1024 ou 2048 e que tenham preço inferior a 900. Neste caso, o SELECT abaixo não atenderia ao desejo do usuário, pois retornaria o conjunto de modelos de PC errado.

```
SELECT * FROM PC

    WHERE ram = 1024 OR ram = 2048 AND price < 900;
```

  - Como o operador AND tem precedência sobre o operador OR, o SGBD resolve está condição deste modo: resolver a primeira parte, seleciona os PC com memória de 2014, resolve a segunda parte, seleciona os PC com memória de 2024 e que tenha preço menor que 900. Depois encontra todas as linhas com ram de 2014 ou que tenha ao mesmo tempo ram de 2048 e preço menor do que 900. **Veja que há preços maiores do que 900.**


Exibição em grade | Visualização do formulário

Total de lin

	model	speed	ram	hd	price
1	1001	2.66	1024	250	2114
2	1004	2.8	1024	250	649
3	1006	3.2	1024	320	1049
4	1007	2.2	1024	200	510
5	1008	2.2	2048	250	770
6	1009	2	1024	250	650
7	1010	2.8	2048	300	770
8	1012	2.8	1024	160	649

Consultas básicas em SQL – Regras de precedência

- Considere o seguinte exemplo:
  - Veja a consulta a seguir, onde os parênteses são empregados para alterar a precedência e, desta forma, recuperar o conjunto de PC corretamente:

```
SELECT * FROM PC
    WHERE (ram = 1024 OR ram = 2048) AND price < 900;
```

  - Deste vez o SGBD selecionar as linhas com memória ram de 1024 ou de 2048 e que na mesma linha tenha preço menor do que 900. Veja que só há preços menores do que 900.

	model	speed	ram	hd	price
1	1004	2.8	1024	250	649
2	1007	2.2	1024	200	510
3	1008	2.2	2048	250	770
4	1009	2	1024	250	650
5	1010	2.8	2048	300	770
6	1012	2.8	1024	160	649

Consultas básicas em SQL – Operador BETWEEN

- Considere o seguinte exemplo:
  - O operador BETWEEN seleciona as linhas que o conteúdo do campo informado esteja entre uma determinada faixa de valores inclusive:

```
SELECT * FROM PC

    WHERE price BETWEEN  700 AND 1000;
```

  - O comando SELECT retorna as tuplas com preço entre 700 e 1000 inclusive.
  - A declaração da faixa de valores deve começar com o limite inferior primeiro, conforme o padrão usado neste exemplo.

	model	speed	ram	hd	price
1	1002	2.1	512	250	995
2	1008	2.2	2048	250	770
3	1010	2.8	2048	300	770
4	1011	1.86	2048	160	959

Consultas básicas em SQL – Operador BETWEEN

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas onde maker está entre 'E' e 'G' inclusive.
  - O operador BETWEEN é muito utilizado para campos do tipo numérico (inteiro e real), mas o valor inicial e valor final podem ser alfanumérico. Quando se usa valor alfanumérico é usada uma comparação lexicográfica (usa a ordem alfabética ou numérica).
  - Veja que são selecionados os fabricantes E, F e G.

```
SELECT * FROM Product
    WHERE maker BETWEEN  'E' AND 'G' and type='laptop'
```

	maker	model	type
1	E	2001	laptop
2	E	2002	laptop
3	E	2003	laptop
4	F	2008	laptop
5	F	2009	laptop
6	G	2010	laptop

Consultas básicas em SQL – Operador BETWEEN

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas onde maker está entre 'E' e 'G' inclusive.
  - O operador BETWEEN é muito utilizado para campos do tipo numérico (inteiro e real), mas o valor inicial e valor final podem ser alfanumérico. Quando se usa valor alfanumérico é usada uma comparação lexicográfica (usa a ordem alfabética ou numérica como se fosse livros de uma biblioteca organizados numa prateleira).
  - Veja que são selecionados os fabricantes E, F e G.

```
SELECT * FROM Product
    WHERE maker BETWEEN  'E' AND 'G' and type='laptop‘;
```

	maker	model	type
1	E	2001	laptop
2	E	2002	laptop
3	E	2003	laptop
4	F	2008	laptop
5	F	2009	laptop
6	G	2010	laptop

Consultas básicas em SQL – Operador IN

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas quando maker está contido na lista de elementos formada por 'E' , ' F ' e 'G' **com o uso do operador IN**.
  - O operador IN permite identificar se o valor do campo avaliado pertence a um determinado conjunto, que deve ser especificado entre parênteses. Este operador pode ser utilizado tanto para avaliar campos alfanuméricos, como para a avaliação de campos numéricos do tipo inteiro. A ordem dos elementos dentro do conjunto não muda o resultado da cláusula IN.
  - Veja que são selecionados os fabricantes E, F e G.

```
SELECT * FROM Product
WHERE maker IN ('E', 'F', 'G') AND type='laptop'
```

	maker	model	type
1	E	2001	laptop
2	E	2002	laptop
3	E	2003	laptop
4	F	2008	laptop
5	F	2009	laptop
6	G	2010	laptop

Consultas básicas em SQL – Operador IN

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas quando hd pertence ao conjunto de elementos formado por 80, 200 e 300 **com o uso do operador IN**.
  - Veja que são selecionados os hd que o tamanho pertença ao conjunto formado pelos elementos numéricos 80, 200 e 300.

```
SELECT * FROM Pc

    WHERE hd IN (80, 200, 300);
```

	model	speed	ram	hd	price
1	1003	1.42	512	80	478
2	1007	2.2	1024	200	510
3	1010	2.8	2048	300	770
4	1013	3.06	512	80	529

Consultas básicas em SQL – Operador NOT IN

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas quando maker não pertence ao conjunto de elementos **com o uso do operador NOT IN**.
  - Veja que são selecionados os produtos que não foram feitos pelos fabricantes A, E e F. O operador **NOT IN** permite identificar se o valor do campo avaliado não pertence a um determinado conjunto.

```
SELECT * FROM Product

WHERE maker NOT IN ('A', 'E', 'F');
```

	maker	model	type
1	B	1004	pc
2	B	1005	pc
3	B	1006	pc
4	B	2007	laptop
5	C	1007	pc
6	D	1008	pc
7	D	1009	pc
8	D	1010	pc
9	D	3004	printer
10	D	3005	printer
11	G	2010	laptop
12	H	3006	printer
13	H	3007	printer

Expressão AR correspondente

$$Product - \sigma_{(Maker = 'A' \text{ OR } maker = 'E' \text{ OR } maker = 'E')}(Product)$$

Consultas básicas em SQL – Comparação Lexicográfica

- Considere o seguinte exemplo:
  - O exemplo seleciona as linhas quando maker tem valor alfanumérico acima de F.
  - Quando um campo deste tipo é avaliado pelos operadores “<”, “<=”, “>=” ou “>” e também pelo operador BETWEEN, o SGBD executará uma comparação lexicográfica, que é baseada em ordenação alfabética.
  - A situações que a comparação lexicográfica permite, por exemplo, encontrar 100 primeiros livros da biblioteca que o sobrenome do autor comece com "G" a primeira letra do título com " m". Exemplo, Código do autor da obra, formado pela primeira letra do sobrenome do autor + número + primeira letra do título. Ex.: G562m.

```
SELECT * FROM Biblioteca

WHERE codigo BETWEEN 'G1m' AND 'G100m';
```

Consultas básicas em SQL – Operações aritméticas

- Considere o seguinte exemplo:
  - O exemplo inclui a execução de operações aritméticas para processar a condição de seleção das linhas.
  - É possível utilizar operações aritméticas sobre colunas na cláusula WHERE. O exemplo retorna todos os modelos e preços dos PC cujo preço seja inferior a R$ 2500.
  - Este exemplo considere que "price" armazena os preços em dólar e que o valor de 1 dólar no dia em que a consulta foi executada equivalia a R$ 3,30. Em SQL, o símbolo "*" é utilizado como operador de multiplicação. Já os símbolos "+", "-" e "/" são utilizados respectivamente como operadores de adição, subtração e divisão.

```
SELECT model, price FROM PC

WHERE (price * 3.30) <= 2500;
```

	model	price
1	1003	478
2	1004	649
3	1005	630
4	1007	510
5	1009	650
6	1013	529
7	1012	649

Consultas básicas em SQL – Operações aritméticas

- Considere o seguinte exemplo:
  - O exemplo inclui a execução de operações aritméticas com resultado do campo do SELECT.
  - É possível utilizar operações aritméticas sobre colunas na cláusula WHERE. O exemplo retorna todos os modelos e preços dos PC cujo preço seja inferior a R$ 2500.
  - Veja que o resultado da expressão aritmética de preço recebe um rótulo novo e o valor é corrido para o suposto valor real.

```
SELECT model, price AS preco_dolar, (price * 3.30) as preco_real FROM PC

WHERE (price * 3.30) <= 2500;
```

	model	preco_dolar	preco_real
1	1003	478	1577.3999999999999
2	1004	649	2141.7
3	1005	630	2079
4	1007	510	1683
5	1009	650	2145
6	1013	529	1745.6999999999998
7	1012	649	2141.7

Consultas básicas em SQL – WHERE na SQL

- A Cláusula WHERE nas Instruções DELETE e UPDATE
- O SQL é uma linguagem de fácil assimilação e de fundamental importância nas operações de banco de dados Relacional. As cláusulas ou palavras chaves seguem uma sintaxe comum nos campos da linguagem.
- O WHERE, por exemplo, é utilizada nas instruções DELETE e UPDATE. Quando utilizada junto à instrução DELETE, a condição especificada na WHERE determinará as linhas que serão removidas. Já na instrução UPDATE, a condição determina as linhas que serão atualizadas. A sintaxe de uso do WHERE com DELETE e UPDATE segue o mesmo padrão do SELECT.

DELETE FROM *Tabela*

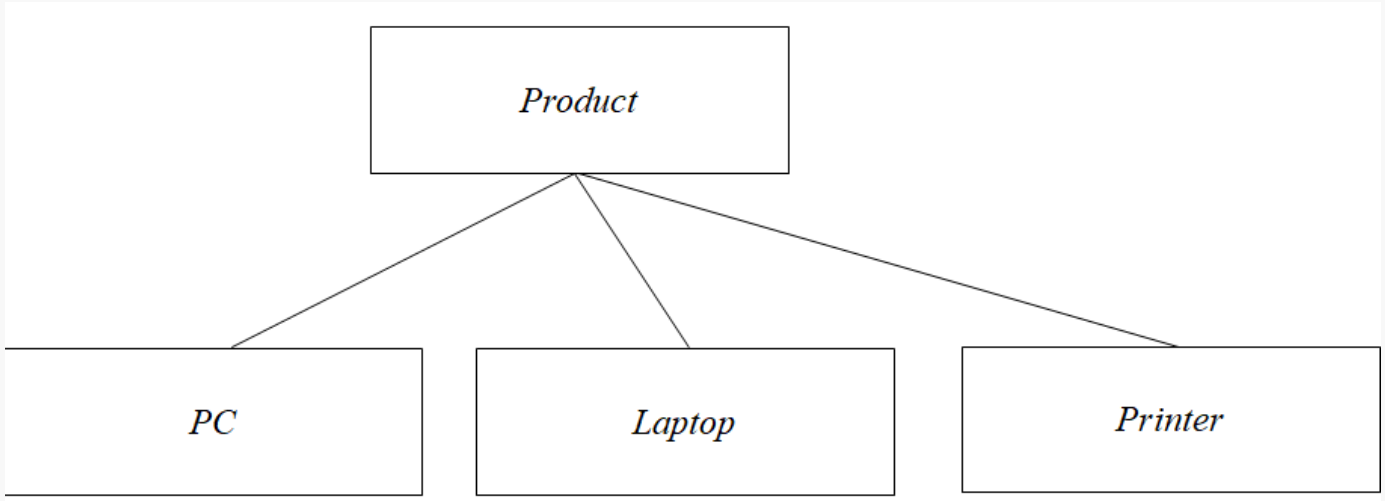
[WHERE condição];

UPDATE *Tabela SET* $atributo_1 = valor_1, \ldots, atributo_1 = valor_1$

[WHERE condição];

Consultas básicas em SQL – Exercícios Propostos 2

1. Crie uma consulta para recuperar todas as impressoras laser coloridas.
2. Crie uma consulta para recuperar todas as impressoras laser coloridas com preço igual ou inferior US$ 200.
3. Cria uma consulta para recuperar todos os laptops com as seguintes características: - Com preço igual ou inferior a US$ 1000,00 (independente das demais características). - Com tamanho de tela igual ou superior a 15 polegadas e preço igual ou inferior a US$ 2000,00 (independente das demais características).
4. Crie duas consultas distintas, uma utilizando o operador IN e outra sem utilizar este operador, para recuperar todos os modelos produzidos pelos fabricantes 'B', 'E' e 'H'.



(1) SELECT * FROM Printer WHERE color=1

(2) SELECT * FROM Printer WHERE color=1 AND price <= 200

(3) SELECT *

FROM Laptop

WHERE price <= 1000 OR (price <= 2000 AND screen >=15)

(4) SELECT model FROM Product WHERE maker IN ('B','E','H')

ou

SELECT model FROM Product

WHERE maker = 'B' OR maker='E' OR maker = 'H'

Consultas básicas em SQL – Exercício de fixação 1

Construa um de BD com o nome de Aula10. Use o script SQL a seguir para criar a tabela Nota_Fiscal e execute os comandos de inserção para popular a tabela com informações.

```
CREATE TABLE NOTA_FISCAL (
ID_ITEM INTEGER,
NFISCAL INTEGER,
COD_PRODUTO INTEGER,
VALOR_UNITARIO REAL,
QUANTIDADE INTEGER,
DESCONTO REAL,
PRIMARY KEY (ID_ITEM,NFISCAL)
                        ) ;
```

Consultas básicas em SQL – Exercícios fixação 1

Script de INSERT

```
INSERT INTO NOTA_FISCAL VALUES(1,1,11,50,10,5);
INSERT INTO NOTA_FISCAL VALUES(1,2,12,55,1,0);
INSERT INTO NOTA_FISCAL VALUES(1,3,13,60,2,0);
INSERT INTO NOTA_FISCAL VALUES(1,4,14,65,5,0);
INSERT INTO NOTA_FISCAL VALUES(2,1,15,70,6,0);
INSERT INTO NOTA_FISCAL VALUES(2,2,16,80,7,0);
INSERT INTO NOTA_FISCAL VALUES(2,3,11,50,9,10);
INSERT INTO NOTA_FISCAL VALUES(2,4,12,55,1,4);
INSERT INTO NOTA_FISCAL VALUES(4,1,13,60,3,4);
INSERT INTO NOTA_FISCAL VALUES(4,2,14,65,6,3);
INSERT INTO NOTA_FISCAL VALUES(4,3,15,70,7,1);
```

Consultas básicas em SQL – Exercícios fixação 1

Script de INSERT

```
INSERT INTO NOTA_FISCAL VALUES(4,4,16,80,2,2);
INSERT INTO NOTA_FISCAL VALUES(5,1,11,50,5,0);
INSERT INTO NOTA_FISCAL VALUES(5,2,12,55,4,0);
INSERT INTO NOTA_FISCAL VALUES(5,3,13,60,3,0);
INSERT INTO NOTA_FISCAL VALUES(5,4,16,80,2,0);
INSERT INTO NOTA_FISCAL VALUES(6,1,15,70,1,0);
INSERT INTO NOTA_FISCAL VALUES(6,2,16,80,1,0);
INSERT INTO NOTA_FISCAL VALUES(7,3,11,50,2,1);
INSERT INTO NOTA_FISCAL VALUES(7,4,12,55,3,5);
INSERT INTO NOTA_FISCAL VALUES(7,5,13,60,4,5);
```

Consultas básicas em SQL – Exercícios fixação 2

(a) Liste a identificação de todas as notas fiscais (NFISCAL), sem repetição, que tiveram, desconto em seus itens.

(b) Elabore uma query que liste a identificação do item, código do produto, valor unitário e preço total de cada item das notas ficais. Atenção o desconto de cada item está espresso em percentual (5 equivale a 5%) (use o rótulo preco_total).

(c) Retorne os códigos dos produtos que seus preços totais em cada nota fiscal estão entre 100 e 200 reais.

(d) Retorne cada produto que não teve desconto e seu preço total de venda na nota ultrapassa 400 reais.

(e) Atualize os descontos das notas fiscais para que seus valores sejam espressos em números decimais inferioriores a 1 (5%=0,05). Os descontos nulos não devem ser alterados.

(f) Liste sem repetição todas a identificação das notas ficais que incluam os produtos 11 ou 12.

Consultas básicas em SQL – Exercícios fixação 2

(g) Liste sem repetição todos os itens de cada nota fiscal com quantidade vendida maior do que 3 e menor do que 7.

(h) Quais notais fiscais o valor total do desconto de um desterminado item está entre 25 e 45 reais inclusive.

(i) Liste somente os valores em reais de todos os descontos fornecidos (use o rótulo desconto_total).

(J) Liste a(s) linhas que contenha(m) quantidade(s) vendida(s) 1ou 2 ou 3 ou 4 e o código do produto seja 13 e o valor do desconto em reais seja menor do que 10 reais.

Consultas básicas em SQL – Exercícios fixação 2

(a) Liste a identificação de todas as notas fiscais (NFISCAL), sem repetição, que tiveram, desconto em seus itens.

```
SELECT DISTINCT NFISCAL
    FROM NOTA_FISCAL WHERE DESCONTO > 0;
```

	NFISCAL
1	1
2	3
3	4
4	2

Consultas básicas em SQL – Exercícios fixação 2

(b) Elabore uma query que liste a identificação do item, código do produto, valor unitário e preço total de cada item das notas ficais. Atenção o desconto de cada item está espresso em percentual (5 equivale a 5%) (use o rótulo preco_total).

```
SELECT ID_ITEM, COD_PRODUTO,
VALOR_UNITARIO,
VALOR_UNITARIO* QUANTIDADE*(1- desconto)/100 as Preco_total
        FROM NOTA_FISCAL;
```

	ID_ITEM	COD_PRODUTO	VALOR_UNITARIO	Preco_total
1	1	11	50	475
2	1	12	55	55
3	1	13	60	120
4	1	14	65	325
5	2	15	70	420
6	2	16	80	560
7	2	11	50	405
8	2	12	55	52.8
9	4	13	60	172.79999999999998
10	4	14	65	378.3
11	4	15	70	485.1
12	4	16	80	156.8
13	5	11	50	250

Consultas básicas em SQL – Exercícios fixação 2

(c) Retorne os códigos dos produtos que seus preços totais em cada nota fiscal estão entre 100 e 200 reais.

```
SELECT COD_PRODUTO

FROM NOTA_FISCAL

WHERE (VALOR_UNITARIO* QUANTIDADE*(1-desconto)/100) BETWEEN 100 AND 200;
```

	COD_PRODUTO
1	13
2	13
3	16

Consultas básicas em SQL – Exercícios fixação 2

(d) Retorne cada produto que não teve desconto e seu preço total de venda na nota ultrapassa 400 reais.

```
SELECT COD_PRODUTO
FROM NOTA_FISCAL
WHERE (VALOR_UNITARIO*QUANTIDADE) <=400
and (DESCONTO=0)
```

	COD_PRODUTO
1	12
2	13
3	14
4	11

Consultas básicas em SQL – Exercícios fixação 2

(e) Atualize os descontos das notas fiscais para que seus valores sejam espressos em números decimais inferioriores a 1 (5%=0,05). Os descontos nulos não devem ser alterados.

```
UPDATE NOTA_FISCAL SET desconto=1,0*desconto/100;
```

Consultas básicas em SQL – Exercícios fixação 2

(f) Liste sem repetição todas a identificação das notas ficais que incluam os produtos 11 ou 12.

```
SELECT  distinct NFISCAL

FROM NOTA_FISCAL

WHERE ID_ITEM=11 OR ID_ITEM =12;
```

Consultas básicas em SQL – Exercícios fixação 2

(g) Liste sem repetição todos os itens de cada nota fiscal com quantidade vendida maior do que 3 e menor do que 7.

```
SELECT  distinct NFISCAL
FROM NOTA_FISCAL
WHERE quantidade>=3 and quantidade<7;
```

Consultas básicas em SQL – Exercícios fixação 2

(h) Quais notais fiscais o valor total do desconto de um desterminado item está entre 25 e 45 reais inclusive.

```
SELECT  NFISCAL
FROM NOTA_FISCAL
WHERE (VALOR_UNITARIO*QUANTIDADE*desconto)
between 25 and 45;
```

	NFISCAL
1	1
2	3

Consultas básicas em SQL – Exercícios fixação 2

(i) Liste somente os valores em reais de todos os descontos fornecidos (use o rótulo desconto_total).

```
SELECT VALOR_UNITARIO*QUANTIDADE*(IFNULL(desconto,0))  as desconto_total
FROM NOTA_FISCAL;
```

	desconto_total
1	25
2	0
3	0
4	0
5	0
6	0
7	45
8	2.2
9	7.2
10	11.7
11	4.9
12	3.2
13	0

Consultas básicas em SQL – Exercícios fixação 2

(J) Liste a(s) linhas que contenha(m) quantidade(s) vendida(s) 1ou 2 ou 3 ou 4 e o código do produto seja 13 e o valor do desconto em reais seja menor do que 10 reais.

```
SELECT *

FROM NOTA_FISCAL

Where quantidade in (1,2,3,4) and cod_produto=13

and (VALOR_UNITARIO*QUANTIDADE*desconto) <10
```

	ID_ITEI	NFISCA	COD_PI	VALOR_	QUANTIDADE	DESCONTO
1	1	3	13	60	2	*NULL*
2	4	1	13	60	3	0.04

Obrigado