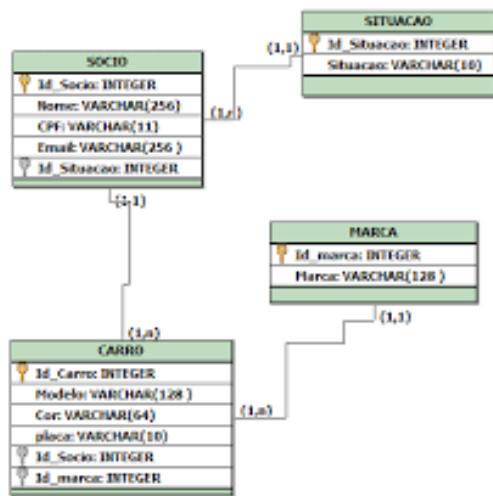


Bancos de Dados

Linguagem SQL – DML – Data Manipulation Language

Comandos INSERT E SELECT



Clóvis José Ramos Ferraro
cferraro@unicid.edu.br

INSERT

DML – INSERÇÃO

- Métodos para inserção em tabelas :
 - INSERT VALUES.
 - INSERT SELECT.
 - INSERT EXEC.
 - SELECT INTO.

(W3SCHOOLS, 2019)

DML – INSERT

- Insere uma linha em uma tabela.
- Não é necessário inserir todas as colunas da tabela de uma vez, desde que as colunas omitidas possam receber NULL ou tenham valor DEFAULT definido.
- Deve ser emitido um comando INSERT para cada linha que se queira inserir na tabela, entretanto, alguns SGBDs permitem que se emitam vários VALUES para um único INSERT, como, por exemplo, o MySQL.
- Colunas com numeração automática (IDENTITY, AUTO_INCREMENT, etc) não precisam ter seus valores inseridos.

(W3SCHOOLS, 2019)

DML – INSERT

- Sintaxe Padrão (**Single Line** – funciona em todos os SGBDs):

```
INSERT INTO table_name (column1) VALUES (value1);
```

```
INSERT INTO table_name (column2) VALUES (value2);
```

TIPO_VEICULO
* <u>COD_TIPO_VEICULO</u> INT
*DSC TIPO VEICULO VARCHAR(45)

- Exemplo:

```
INSERT INTO TIPO_VEICULO (DSC_TIPO_VEICULO) VALUES ('CAMINHÃO');
```

```
INSERT INTO TIPO_VEICULO (DSC_TIPO_VEICULO) VALUES ('SEDAN');
```

```
INSERT INTO TIPO_VEICULO (DSC_TIPO_VEICULO) VALUES ('PICKUP');
```

- Campos com valores automáticos (IDENTITY, AUTOINCREMENT, etc) não precisam ser inseridos.
- É opcional a inserção de campos com valor padrão (DEFAULT).
- É obrigatório a inserção de campos NOT NULL.

(W3SCHOOLS, 2019)

DML – INSERT

- Insert into **Multiple Line** Sintaxe

```
INSERT INTO table_name (column1,column2,column3,...) VALUES (value1,  
value2, value3, ...);
```

TIPO_VEICULO
* <u>COD_TIPO_VEICULO INT</u>
*DSC TIPO VEICULO VARCHAR(45)

- Exemplo:

```
INSERT INTO TIPO_VEICULO (DSC_TIPO_VEICULO) VALUES ('HATCHBACK'), ('STATION  
WAGON'), ('SUV'), ('MOTOCICLETA'), ('COUPE'), ('FASTBACK');
```

- Você pode emitir vários VALUES para apenas um INSERT, separados por vírgulas.

(W3SCHOOLS, 2019)

DML – INSERT

```
Select * from TIPO_VEICULO;  
DESC TIPO_VEICULO;  
Truncate TIPO_VEICULO;  
COMMIT;
```

(W3SCHOOLS, 2019)

DML – INSERT

Drop - apaga a tabela e a estrutura desta;

Truncate - apaga todos os dados e mantém a estrutura desta, retorna ao tamanho inicial;

Delete - apaga todos os dados(deleção lógica);

(W3SCHOOLS, 2019)

DML – INSERT

- Insert into **Multiple Line** Sintaxe

✓ O problema é que não funciona em todos os SGBDs.

The image displays two side-by-side screenshots of a 'Query Builder' interface, illustrating the difference in INSERT syntax between Oracle and PostgreSQL.

Left Screenshot (Oracle): The SQL editor shows three separate INSERT statements:
`INSERT INTO PESSOA (COD_PESSOA ,NOM_PESSOA) VALUES (1, 'EMERSON FITIPALDI');
INSERT INTO PESSOA (COD_PESSOA ,NOM_PESSOA) VALUES (2, 'NELSON PIQUET');
INSERT INTO PESSOA (COD_PESSOA ,NOM_PESSOA) VALUES (3, 'AYRTON SENNA');`
The 'Saída do Script' (Script Output) window at the bottom shows the successful execution of these three statements, each inserting one row.

Right Screenshot (PostgreSQL): The SQL editor shows a single INSERT statement with multiple values:
`INSERT INTO PESSOA (COD_PESSOA ,NOM_PESSOA) VALUES (1, 'EMERSON FITIPALDI'),
(2, 'NELSON PIQUET'),(3, 'AYRTON SENNA');`
The 'Saída do Script' window shows an error: 'Erro a partir da linha : 1 no comando - INSERT INTO PESSOA (COD_PESSOA ,NOM_PESSOA) VALUES (1, 'EMERSON FITIPALDI'), (2, 'NELSON PIQUET'),(3, 'AYRTON SENNA')', followed by 'Erro na Linha de Comandos : 1 Coluna : 76' and 'Relatório de erros - Erro de SQL: ORA-00933: comando SQL não encerrado adequadamente 00933. 00000 - "SQL command not properly ended"'. This demonstrates that the multiple-line syntax is not supported in all databases.

DML – INSERT

- Insert Select

- ✓ Insere o conjunto de resultado devolvido por uma consulta em uma tabela especificada.
- ✓ A especificação dos nomes das colunas também é opcional.
- ✓ Também pode-se omitir colunas com propriedade IDENTITY (SQL Server) ou AUTO_INCREMENT (MySQL), valores DEFAULT ou com permissão de NULLs.
- ✓ **Consultas serão nosso próximo assunto!**

- Sintaxe

```
INSERT INTO table2 (column1, column2, column3, ...)
    SELECT column1, column2, column3, ...
    FROM table1
    WHERE condition;
```

DML – INSERT

- Insert Select Exemplo:

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)

      SELECT      SupplierName,    ContactName,    Address,    City,
PostalCode, Country      FROM Suppliers;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Customers

Suppliers

SupplierID	SupplierName	ContactName	Address	City	Postal Code	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

DML – INSERT

- O **MySQL** utiliza aspas simples (apóstrofos) para textos.
- Para inserir um campo do tipo data, você deve passar uma string com o formato dependendo do tipo do dado:

DATE esta no formato AAAA-MM-DD 2020-09-04

DATETIME esta no formato AAAA-MM-DD HH:MM:SS 2020-09-04 20:59:45

TIMESTAMP esta no formato AAAAMMDDHHMMSS 20200904205945

- Nos valores decimais, deve-se utilizar ponto.

(W3SCHOOLS, 2019)

DML – INSERT

- O **Oracle** utiliza aspas simples (apóstrofes) para textos.
- Para inserir um campo do tipo DATE, você deve utilizar a função TO_DATE() passando a string de data :

```
1  INSERT INTO PRODUTO (  
2      → COD_PROD,  
3      → NOM_PROD,  
4      → PCO_PROD,  
5      → VEM_PROD  
6  ) VALUES (  
7      → 1,  
8      → 'EXTRATO TOMATE MAMUTE',  
9      → 1.35,  
10     → TO_DATE('31/03/2009', 'DD/MM/YYYY')  
11 );
```

- Nos valores decimais, deve-se utilizar ponto.

(W3SCHOOLS, 2019)

DML – INSERT

- O **PostgreSQL** e o **SQL Server** são muito similares ao MySQL, sendo necessário utilizar um dos dois formatos abaixo na inserção de datas:

# DATE	esta no formato AAAA-MM-DD
# DATETIME	esta no formato AAAA-MM-DD HH:MM:SS

(W3SCHOOLS, 2019)

DML – INSERT

- Exemplo no **MySQL**, **PostgreSQL** e **SQL Server**:

```
1  INSERT INTO PRODUTO (  
2      → COD_PROD,  
3      → NOM_PROD,  
4      → PCO_PROD,  
5      → VEM_PROD  
6  ) VALUES (  
7      → 1,  
8      → 'EXTRATO TOMATE MAMUTE',  
9      → 1.35,  
10     → '2008-03-31')  
11 );
```

(W3SCHOOLS, 2019)

DML – INSERT

- O **FireBird** utiliza aspas simples (apóstrofos) para textos.
- Para inserir um campo do tipo DATE, você deve usar a função **CAST**, tendo a string o formato MM-DD-YYYY:

```
1  INSERT INTO PRODUTO (  
2      → COD_PROD,  
3      → NOM_PROD,  
4      → PCO_PROD,  
5      → VEM_PROD  
6  ) VALUES (  
7      → 1,  
8      → 'EXTRATO TOMATE MAMUTE',  
9      → 1.35,  
10     → CAST('2008-03-31' AS DATE))  
11 );
```

- Nos valores decimais, deve-se utilizar ponto.

(W3SCHOOLS, 2019)

DML – INSERT



- Oracle: SYSDATE.
- MySQL, MariaDB ou PostgreSQL:
 - CURRENT_DATE
 - CURRENT_TIMESTAMP
- FireBird:
 - NOW,
 - CURRENT_DATE ou
 - CURRENT_TIMESTAMP
- SQL Server:
 - GETDATE()
 - CURRENT_TIMESTAMP

EXERCÍCIOS

Prática

Loja

Criação das tabelas

EXERCÍCIOS

Prática

Loja

Inserção de dados nas tabelas

SELECT

DQL – Data Query Language (SELECT)

- Pesquisa dados armazenados em uma tabela.
- Permite recuperar:
 - ✓ Todas as colunas (SELECT * FROM NOME_TABELA)
 - ✓ Colunas específicas, indicando o nome da coluna que se deseja pesquisar (SELECT NOM_DEPARTAMENTO FROM DEPARTAMENTO)
 - ✓ Expressões aritméticas, compostas somente por colunas ou colunas e constantes
 - ✓ Colunas indicadas por apelidos (ALIAS) ou com títulos, indicados entre aspas duplas “”.

DQL – Data Query Language (SELECT)

Cláusula	Expressão	Descrição
SELECT	<lista de seleção>	Quais colunas (atributos) serão devolvidos.
FROM	<tabela de origem>	Tabelas envolvidas na consulta.
WHERE	<condição da pesquisa>	Filtra as linhas (tuplas) desejadas.
GROUP BY	<lista de grupos>	Agrupa as linhas por grupos.
HAVING	<condição do agrupamento>	Filtra as linhas pelo agrupamento.
ORDER BY	<ordem da lista>	Ordena a resposta da consulta.

DQL – Data Query Language (SELECT)

- Ordem de escrita:

- 1) SELECT
- 2) FROM
- 3) WHERE
- 4) GROUP BY
- 5) HAVING
- 6) ORDER BY

- Ordem de execução:

- 1) FROM
- 2) WHERE
- 3) GROUP BY
- 4) HAVING
- 5) SELECT
- 6) ORDER BY

DQL – Data Query Language (SELECT)

- Sintaxe:

```
SELECT [ALL | DISTINCT]
      * | COL1 [, COL2, COL3, ... COLn]
      [, FUNCAO(nome_funcao)]
      [, TEXTO]
FROM TABELA1 [, TABELA2, ..., TABELAn]
[JOIN TABELA ON condição_lógica]
[LEFT JOIN TABELA ON condição_lógica]
[RIGHT JOIN TABELA ON condição_lógica]
[WHERE condição_logica]
[GROUP BY COL1 [, COL2, ..., COLn]]
[HAVING condição_lógica]
[ORDER BY COL1 [, COL2, ..., COLn]]
```

(W3SCHOOLS, 2019)

A cláusula FROM

SELECT – CLÁUSULA FROM

- A cláusula FROM é a primeira cláusula a ser avaliada logicamente em uma consulta SELECT.
- Dois papéis:
 - ✓ Indicar as tabelas que você quer consultar.
 - ✓ Aplicar operadores de tabelas como junções à tabelas de entrada.




(W3SCHOOLS, 2019)

SELECT – CLÁUSULA FROM

- Exemplo:

Liste todos os GRUPOS

6 • `SELECT * FROM LOJA.grupo;`

< Result Grid   Filter Rows: Edit: 

	COD_GRUPO	NOM_GRUPO	IDF_ATIVO
▶	1	ALIMENTOS E BEBIDAS	S
	2	INFORMÁTICA E TECNOLOGIA	S
	3	CASA E ESCRITÓRIO	S
	4	CULTURA E DIVERSÃO	S
	5	MODA E BELEZA	S
	6	FERRAMENTAS E AUTOMOTIVO	S
	7	ESPORTE E SAÚDE	S
	8	BEBÊS E BRINQUEDOS	S
*	NULL	NULL	NULL

SELECT – CLÁUSULA FROM

- Observe o nome de duas partes:

- ✓ Nome do esquema.

- ✓ Nome da tabela.

```
SELECT * FROM LOJA.grupo;
```

- Em alguns casos pode-se omitir o nome do esquema.

- ✓ Boa prática sempre se referir ao nome do esquema.

- ✓ Pode-se terminar em um esquema não desejado se não explicitar.

(W3SCHOOLS, 2019)

SELECT – CLÁUSULA FROM

6 • **SELECT *** FROM LOJA.grupo;

< **Result Grid** | Filter Rows: | Edit:

	COD_GRUPO	NOM_GRUPO	IDF_ATIVO
▶	1	ALIMENTOS E BEBIDAS	S
	2	INFORMÁTICA E TECNOLOGIA	S
	3	CASA E ESCRITÓRIO	S
	4	CULTURA E DIVERSÃO	S
	5	MODA E BELEZA	S
	6	FERRAMENTAS E AUTOMOTIVO	S
	7	ESPORTE E SAÚDE	S
	8	BEBÊS E BRINQUEDOS	S
*	NULL	NULL	NULL

8 • **SELECT COD_GRUPO, NOM_GRUPO** FROM LOJA.grupo;

< **Result Grid** | Filter Rows: | Edit:

	COD_GRUPO	NOM_GRUPO
▶	1	ALIMENTOS E BEBIDAS
	2	INFORMÁTICA E TECNOLOGIA
	3	CASA E ESCRITÓRIO
	4	CULTURA E DIVERSÃO
	5	MODA E BELEZA
	6	FERRAMENTAS E AUTOMOTIVO
	7	ESPORTE E SAÚDE
	8	BEBÊS E BRINQUEDOS
*	NULL	NULL

SELECT – CLÁUSULA FROM

- Podemos renomear (apelidar) a tabela em uma cláusula FROM:

```
SELECT *  
FROM LOJA.GRUPO G;
```

OU

```
SELECT *  
FROM LOJA.GRUPO AS GRU;
```

Recomendável

(W3SCHOOLS, 2019)

SELECT – CLÁUSULA FROM

- A segunda forma é a recomendável: `SELECT * FROM LOJA.GRUPO AS GRU`
 - ✓ Mais legível.
 - ✓ Nome da tabela.
- Por que renomear?
 - ✓ Ficará mais claro quando vermos junção entre tabelas.
 - ✓ No momento: Sintaxe permite que façamos isso.
- O novo nome é válido somente **durante a duração da consulta.**

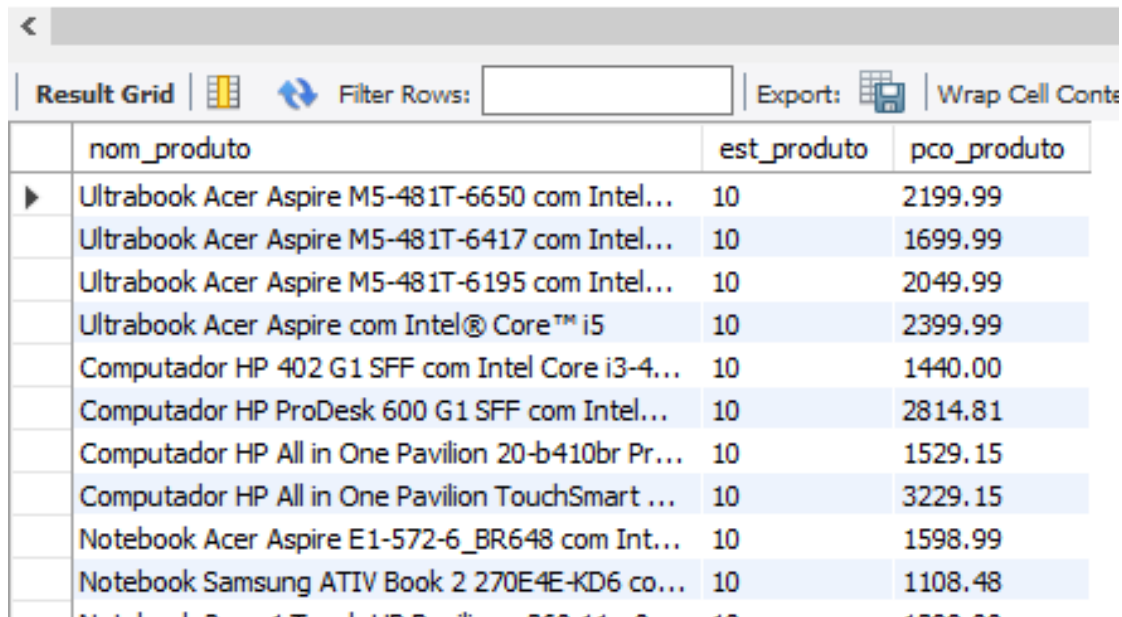
SELECT – CLÁUSULA SELECT

- Exemplo:
 - ✓ Liste o nome, o estoque e o preço dos produtos ativos.

SELECT – CLÁUSULA SELECT

```
SELECT nom_produto, est_produto, pco_produto  
FROM LOJA.produto  
WHERE idf_ativo = 'S';
```

```
2 • SELECT nom_produto, est_produto, pco_produto  
3     FROM LOJA.produto  
4     WHERE idf_ativo = 'S';
```



The screenshot shows a database query result grid. The grid has four columns: 'nom_produto', 'est_produto', and 'pco_produto'. The first column contains product names, the second column contains stock status (all '10'), and the third column contains prices. The grid is titled 'Result Grid' and includes a 'Filter Rows' input field and an 'Export' button. The data is as follows:

nom_produto	est_produto	pco_produto
Ultrabook Acer Aspire M5-481T-6650 com Intel...	10	2199.99
Ultrabook Acer Aspire M5-481T-6417 com Intel...	10	1699.99
Ultrabook Acer Aspire M5-481T-6195 com Intel...	10	2049.99
Ultrabook Acer Aspire com Intel® Core™ i5	10	2399.99
Computador HP 402 G1 SFF com Intel Core i3-4...	10	1440.00
Computador HP ProDesk 600 G1 SFF com Intel...	10	2814.81
Computador HP All in One Pavilion 20-b410br Pr...	10	1529.15
Computador HP All in One Pavilion TouchSmart ...	10	3229.15
Notebook Acer Aspire E1-572-6_BR648 com Int...	10	1598.99
Notebook Samsung ATIV Book 2 270E4E-KD6 co...	10	1108.48

SELECT – CLÁUSULA FROM

- Exercício 1 : Faça as seguintes consultas ao Banco Loja:
 - ✓ Liste os clientes.
 - ✓ Liste os fabricantes.
 - ✓ Liste as promoções.

A cláusula **SELECT**

SELECT – CLÁUSULA SELECT

- Avalia expressões que definem os atributos do resultado da consulta, os renomeando se necessário.
- Caso seja necessário eliminar linhas duplicadas, usar a cláusula **DISTINCT**.
- Partição vertical.



SELECT – CLÁUSULA SELECT

- Exemplo:

✓ Quais são os DEPARTAMENTOS das CATEGORIAS?

```
SELECT COD_DEPARTAMENTO  
      FROM LOJA.CATEGORIA;
```

18 • **SELECT** COD_DEPARTAMENTO
19 **FROM** LOJA.CATEGORIA;

< Result Grid   Filter Rows:

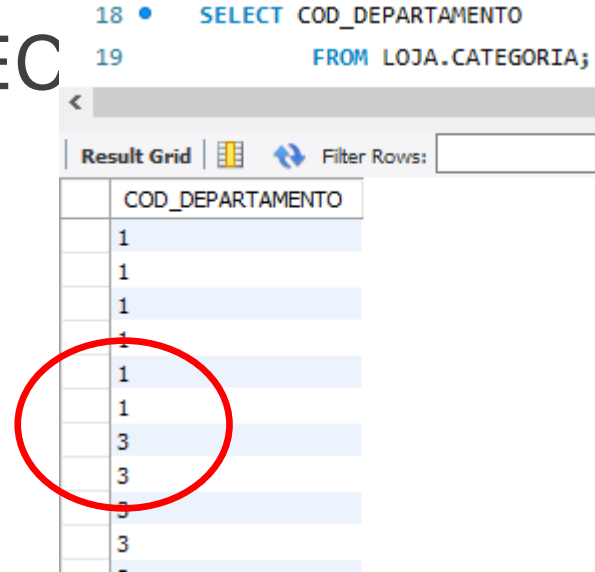
	COD_DEPARTAMENTO
	1
	1
	1
	1
	1
	1
	3
	3
	3
	3
	-

SELECT – CLÁUSULA SELEC

- Observe o resultado:

✓ Mesmo COD_DEPARTAMENTO em mais de uma linha no resultado.

```
18 • SELECT COD_DEPARTAMENTO
19 FROM LOJA.CATEGORIA;
```

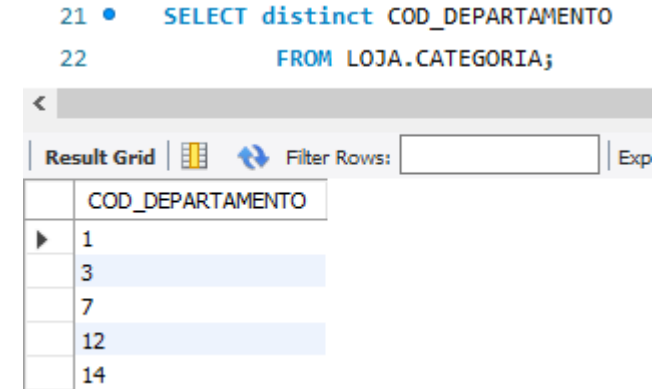


COD_DEPARTAMENTO
1
1
1
1
1
1
1
3
3
3
3

✓ Para selecionar somente os COD_DEPARTAMENTO distintos: usar **DISTINCT**.



```
21 • SELECT distinct COD_DEPARTAMENTO
22 FROM LOJA.CATEGORIA;
```



COD_DEPARTAMENTO
1
3
7
12
14

SELECT – CLÁUSULA SELECT

- Usar SELECT * ...
 - ✓ Lista todos os atributos das tabelas de entrada.
 - ✓ Considerada uma **má prática**.
 - ❖ Frequentemente só é necessário um subconjunto dos atributos.
 - * é só preguiça.
 - ❖ Devolver mais atributos que o que é necessário pode diminuir desempenho da consulta – utilização de índices.
 - ❖ Envia mais dados pela rede – impacto no desempenho do sistema.
 - ❖ Se definição da tabela mudar: todos os atributos eram necessários antes, mas não agora.

SELECT – CLÁUSULA SELECT

- Boa prática: Sempre explicitar os atributos que são necessários.

SELECT – CLÁUSULA SELECT

- Exercício 2:
 - ✓ O diretor solicitou um relatório com todos os produtos, contendo nome, preço e estoque.

SELECT – CLÁUSULA SELECT

- Na cláusula SELECT podemos renomear as expressões que definem os atributos resultantes.

✓ <expressão> AS <novo_nome>

Pnome **AS** PrimeiroNome

Recomendável

✓ <expressão> <novo_nome>

Pnome PrimeiroNome

✓ <expressão> = <expressão>

Pnome = PrimeiroNome

SELECT – CLÁUSULA SELECT

- Ao aplicar uma operação ou função na cláusula SELECT, não há nome de coluna:

```
SELECT NOM_CLIENTE, CONCAT(END_CLIENTE, ' - CEP:', CEP_CLIENTE)  
FROM LOJA.CLIENTE;
```

	NOM_CLIENTE	CONCAT(END_CLIENTE, ' - CEP:', CEP_CLIENTE)
▶	EMERSON FITIPALDI	RUA LOTUS, 2 - CEP:14100000
	NELSON PIQUET	AV DA WILLIAMS, 3 - CEP:14200000
	AIRTON SENNA	AV MACLAREN, 3 - CEP:14300000
	RUBENS BARRICHELLO	TRAVESSA DA FERRARI, 2 - CEP:14400000
	FELIPE MASSA	TRAVESSA DA SAUBER, 2 - CEP:14500000

SELECT – CLÁUSULA SELECT

- Ao aplicar uma operação ou função na cláusula SELECT, não há nome de coluna:

```
SELECT NOM_CLIENTE, CONCAT(END_CLIENTE, ' - CEP:', CEP_CLIENTE) AS ENDERECO  
FROM LOJA.CLIENTE;
```

	NOM_CLIENTE	ENDERECO
▶	EMERSON FITIPALDI	RUA LOTUS, 2 - CEP:14100000
	NELSON PIQUET	AV DA WILLIAMS, 3 - CEP:14200000
	AIRTON SENNA	AV MACLAREN, 3 - CEP:14300000
	RUBENS BARRICHELLO	TRAVESSA DA FERRARI, 2 - CEP:14400000
	FELIPE MASSA	TRAVESSA DA SAUBER, 2 - CEP:14500000

SELECT – CLÁUSULA SELECT

- Atenção:
 - ✓ A concatenação (soma de strings) é feita de diferentes formas em diferentes SGBDs:
 - **SQL Server** : +
 - **MySQL**: função CONCAT(Str1,Str2,...,Strn)
 - **Oracle, PostgreSQL, Firebird**: ||

SELECT – CLÁUSULA SELECT

- Ao aplicar uma operação ou função na cláusula SELECT, não há nome de coluna:

```
SELECT      lower(NOM_CLIENTE)      FROM LOJA.CLIENTE;
```

	LOWER(NOM_CLIENTE)
▶	emerson fitipaldi
	nelson piquet
	airton senna
	rubens barrichello
	felipe massa

SELECT – CLÁUSULA SELECT

- Precedência:

```
SELECT 20 + 20 / 5 ...;
```

```
SELECT (20 + 20) / 5 ...;
```

```
SELECT 20 + (20 / 5) ...;
```

```
SELECT (((10 + 2) / 2) * 0.3) % 2 ...;
```

Referências

Referências

- ELMASRI, Ramez; NAVATHE, Shamkant B.. Parte 7 – Estruturas de arquivo, indexação e hashing: Capítulo 17 – Armazenamento de disco, estruturas de arquivos básicas e hashing. In: ELMASRI, Ramez; NAVATHE, Shamkant B.. Sistema de Banco de Dados. 6. ed. São Paulo: Pearson Education, 2011. Cap. 4. p. 57-75.
- RAMAKRISHNAN, Raghu; GEHRKE, Johannes. SQL, Consultas, Restrições, Gatilhos. In: RAMAKRISHNAN, Raghu; GEHRKE, Johannes. Sistemas de Gerenciamento de Banco de Dados. 3. ed. São Paulo: Mc Graw Hill, 2008. Cap. 5. p. 110-147. Tradução da Terceira Edição.
- RANGEL, Alexandre Leite. DML: Data Manipulation Language. In: RANGEL, Alexandre Leite. InterBase 7: Desenvolvendo e Administrando Bancos de Dados. Rio de Janeiro: Alta Books, 2003. Cap. 2. p. 78-102.

Referências

- RANGEL, Alexandre Leite. Manipulação de Dados em MySQL. In: RANGEL, Alexandre Leite. MySQL - Projeto, Modelagem e Desenvolvimento de Bancos de Dados. Rio de Janeiro: Alta Books, 2004. Cap. 6. p. 74-96.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUNDARSHAN, S.. Introdução à SQL. In: SILBERSCHATZ, Abraham; KORTH, Henry F.; SUNDARSHAN, S.. Sistemas de Banco de Dados. 5. ed. Rio de Janeiro: Campus, 2006. Cap. 3. p. 37-67.
- W3SCHOOLS. SQL INSERT INTO Statement. Disponível em: <https://www.w3schools.com/sql/sql_insert.asp>. Acesso em: 19 jun. 2019.
- _____. SQL INSERT INTO SELECT Statement. Disponível em: <https://www.w3schools.com/sql/sql_insert_into_select.asp>. Acesso em: 19 jun. 2019.

Referências

- W3SCHOOLS. MySQL String Functions. Disponível em: <https://www.w3schools.com/sql/sql_ref_mysql.asp>. Acesso em: 27 jun. 2019.
- _____. SQL SELECT Statement. Disponível em: <https://www.w3schools.com/sql/sql_select.asp>. Acesso em: 27 jun. 2019.
- _____. SQL Server Functions. Disponível em: <https://www.w3schools.com/sql/sql_ref_sqlserver.asp>. Acesso em: 27 jun. 2019.

Obrigado!

Prof. Clóvis Ferraro

(Adaptado de Alexandre Rangel)