# Banco de Dados

**"A confiança em si mesmo é o primeiro segredo do sucesso."**

Ralph W. Emerson

# NOTAS DE AULA

(Laboratório)

Arquivo 2

Rafael Dias Ribeiro, M.Sc

rafaeldiasribeiro@gmail.com.br

**Sócio Efetivo da:**

Sociedade Brasileira de Computação

## Inserindo Dados em uma Tabela

Inserindo dados  em uma tabela utilizando o INSERT:

SINTAXE:

INSERT [LOW_PRIORITY | DELAY | HIGH_PRIORITY] [IGNORE]
[INTO] <nome-tabela> [( <nome-atributo>,...)]
VALUES ({expr | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE <nome-atributo>=expr,...]

## Inserindo Dados em uma Tabela

Inserindo dados  em uma tabela utilizando o INSERT:

SINTAXE:

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]

LOW_PRIORITY: A inserção só será realizada no momento em que não houver nenhum acesso a tabela referida. Caso esteja trabalhando na interface cliente, ela ficará aguardando a confirmação da inserção, o que pode demorar algum tempo.

DELAYED: Tem o comportamento similar ao LOW_PRIORITY porém a interface cliente recebe um OK e é liberada de qualquer tipo de espera.

HIGH_PRIORITY: O SGBD irá priorizar o comando em relação a qualquer outro comando referente a tabela.

IGNORE: O insert não mostrará eventuais mensagens de erro, caso exista. (ex: qualquer inserção duplicada ou erro no tipo de dados inseridos)

[ON DUPLICATE KEY UPDATE <nome-atributo>=expr,...]

## Inserindo Dados em uma Tabela

Inserindo dados  em uma tabela utilizando o INSERT:

SINTAXE:

[ON DUPLICATE KEY UPDATE <nome-atributo>=expr,...]

Se você especificar se uma cláusula ON DUPLICATE KEY UPDATE, e uma linha que causasse a duplicação de um valor fosse inserida em uma chave PRIMARY ou UNIQUE, um UPDATE da linha antiga seria realizado.

Exemplo:

mysql> INSERT INTO table (a,b,c) VALUES (1,2,3) -> ON DUPLICATE KEY UPDATE c=c+1;

Exemplo:

```
mysql> INSERT INTO carros
-> VALUES (‘LTP0251','FIAT',‘UNO',‘2002-05-19',NULL);
```

OBS: Perceba que os valores de string e datas são especificados aqui como strings com aspas.
Com o INSERT você também pode inserir NULL diretamente para representar um valor em falta.

Insira na tabela os seguintes registros:

| PLACA | MARCA | MODELO | COMPRA | VENDA |
|---|---|---|---|---|
| LNC0211 | Fiat | Marea | 2002-08-13 | NULL |
| LTP0343 | Ford | Ranger | 2000-05-14 | NULL |
| LNC3512 | Gurgel | Carajás | 1999-07-15 | 2002-02-19 |
| XYS8888 | Fiat | Uno | 1998-11-16 | NULL |
| XAB7788 | Peugeot | 206 | 2003-04-17 | 2006-12-29 |

## Inserindo Dados em uma Tabela

Inserindo dados a partir de uma fonte externa em uma tabela utilizando o LOAD DATA:

SINTAXE:

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE ‘nome-arquivo.txt‘
[REPLACE | IGNORE]
INTO TABLE nome-tabela
[FIELDS
[TERMINATED BY '\t']
[[OPTIONALLY] ENCLOSED BY ''] [ESCAPED BY '\\' ] ]
[LINES [STARTING BY ''] [TERMINATED BY '\n'] ]
[IGNORE n LINES] [(nome_coluna,...)]
```

## Inserindo Dados em uma Tabela

Inserindo dados a partir de uma fonte externa em uma tabela utilizando o LOAD DATA:

SINTAXE:

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'nome-arquivo.txt'

LOW_PRIORITY: Similar a regra apresentada no comando INSERT

CONCURRENT: O comando deve ser executado de forma concorrente a outros processos ou usuários que estejam acessando a tabela referida.

[REPLACE | IGNORE]

REPLACE: As linhas inseridas substituirão as linhas existentes (em outras palavras, linhas que tiverem o mesmo valor de um índice primário ou único como linhas existentes).

IGNORE: Registros inseridos que duplicam uma linha existente em um valor de chave única será ignorados.

OBS: Se você não especificar nenhuma das opções, o comportamento depende de se a palavra chave LOCAL é especificada ou não. Sem LOCAL, um erro ocorre quando um valor de chave duplicada é encontrado, e o resto do arquivo texto é ignorado. Com LOCAL o comportamento padrão é o mesmo de quando IGNORE for especificado, isto é porque o servidor não tem como parar no meio da operação.

## Inserindo Dados em uma Tabela

Inserindo dados a partir de uma fonte externa em uma tabela utilizando o LOAD DATA:

SINTAXE:

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE ‘nome-arquivo.txt‘

INFILE ‘nome-arquivo.txt‘: Se o utilizado o parâmetro LOCAL,o arquivo ‘nome-arquivo.txt‘ é lido na máquina client que executa o comando e nâo o servidor (default). Em função desta possibilidade, que pode a problemas de segurança,é possível desabilitar o uso do comando LOAD DATA, que pode ser feito através do arquivo de configuração (my.ini), onde se altera o valor da variável local_infile = 0 (o valor default é 1)

[FIELDS

[TERMINATED BY '\t']

TERMINATED BY '\t‘: Indica qual caracter será o delimitador de campos do arquivo.
Se for o 'TAB', não precisamos declará-lo (por ser padrão). Aqui podemos definir outro padrão, como por exemplo o ' ; ';

## Inserindo Dados em uma Tabela

Inserindo dados a partir de uma fonte externa em uma tabela utilizando o LOAD DATA:

SINTAXE:

ENCLOSED BY '']

ENCLOSED BY '': Indica o caracter que será usado para envolver os campos no arquivo. Se não colocarmos o parâmetro OPTIONALLY, todos os campos começarão e terminarão com o caracter especificado. Colocando o OPTIONALLY, apenas os campos CHAR, VARCHAR e TEXT começarão e terminarão com este caracter especificado.

[ESCAPED BY '\\' ] ]

ESCAPED BY : Controla a forma como determinados caracteres foram gravados no arquivo. É válido quando queremos que a consulta informe um valor alternativo a determinados valores (por exemplo campos com conteúdo NULL).

[LINES [STARTING BY ''] [TERMINATED BY '\n'] ]

[LINES [STARTING BY ''] [TERMINATED BY '\n'] ]: Este parâmetro, opcional, indica qual deve ser o caracter que indica oinicio e o  fim de linha.

[IGNORE n LINES] [(nome_coluna,...)] : Ignora as n primeiras linhas do arquivo lido.

## Inserindo Dados em uma Tabela

Inserindo dados a partir de uma fonte externa em uma tabela utilizando o LOAD DATA:

Ex:

```
mysql> LOAD DATA INFILE 'carros.txt'  INTO TABLE carros
-> FIELDS TERMINATED BY ' , '
-> OPTIONALLY ENCLOSED BY ' " '
-> LINES TERMINATED BY '\n' ;
```

Formatação do arquivo de entrada c:\mysql\data\aula\carros.txt

LNC0211, "FIAT","MAREA",2002-02-13, NULL
KXP0201, "GM","VECTRA",2003-09-13, NULL
MMM0001, "GURGEL","CARAJAS",1998-02-13, 2002-02-12

## Alterando registros

Para alterar registros inseridos em uma tabela, usamoso comando UPDATE.

Sintaxe:

UPDATE [LOW_PRIORITY] [IGNORE] nome-tabela
SET nome_coluna1=expr1 [, nome_coluna2=expr2 ...]
[WHERE definição_where]
[ORDER BY ...] [LIMIT row_count]

[LOW_PRIORITY] [IGNORE]: Mesma funcionalidade apresentada no INSERT

SET : Indica quais atributos terão seu conteúdo atualizado

ORDER BY: Se esta é utilizada, as linhas serão atualizadas nesta ordem. Isto só é útil em conjunto com LIMIT.

LIMIT: Pode ser utilizado para assegurar que apenas um determinado número de linhas será atualizado

## Alterando registros

Exercício:
Suponhamos agora que você vendeu seu Fiat Marea. A primeira providência, é alterar o campo venda de sua tabela para que você não possua dados desatualizados.

mysql> UPDATE carros SET venda = '2006-12-29' WHERE placa = 'LNC0211';

Usando o SELECT, verifique se o registro foi atualizado.
venda = '2006-12-29' campo que será substtuido. Podemos atualizar diversos campos simultaneamente utilizando vírgula

ex:
UPDATE carros SET venda = '2006-12-29', placa='LNC9999 ' WHERE marca = 'Gurgel ';

## Exluindo Registros

Quando um registro não nos é mais útil, podemos apagá-lo, para que a tabela não contenha dados obsoletos.

SINTAXE:

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM nome-tabela
[WHERE definição_where]
[ORDER BY ...] [LIMIT row_count]

OBSERVAÇÕES:

DELETE deleta linhas de nome-tabela que satisfaçam a condição dada por definição_where, e retorna o número de registros deletados.

**Se você executar um DELETE sem cláusula WHERE, todas as linhas são deletadas.**

## Exercício:

Exclua o registro de placa LNC3512

mysql> DELETE FROM carros WHERE placa = 'LNC3512 ';

Usando o SELECT, verifique se o registro foi excluido.
WHERE placa = 'LNC3512 ' é a condição para se deletar o registro

## Utilizando Condições

Para fazermos o uso de condições, devemos usar o comando WHERE juntamente com os operadores.

Os operadores de comparação lógica estão divididos em duas classes:

Operadores de linha única:

- = igual a
- ! diferente de
- > maior que
- >= maior ou igual a
- < menor que
- <= menor ou igual a

Operadores de várias linhas

- AND - e
- OR - ou
- NOT - não

## Utilizando condições

Precedência de operadores:

Uma cláusula WHERE pode combinar vários operadores AND e OR.

O operador AND tem maior precedência que o operador OR.

Os operadores de comparação de linha única tem maior precedência que os operadores AND e OR.

Todos os operadores de comparação de linha única têm a mesma precedência.

Operadores de igual precedência são calculados da esquerda para a direita.

A precedência de operadores pode ser cancelada através de parênteses;