

LINGUAGEM SQL BÁSICO



Wagner Bianchi

Certified MySQL 5.0 Developer

Certified MySQL 5.0 Database Administrator

Certified MySQL 5.1 Cluster Database Administrator



Introdução

- 🐞 **Structured Query Language**, ou Linguagem de Consulta Estruturada ou SQL, é uma linguagem de pesquisa declarativa para bancos de dados relacional (base de dados relacional). Muitas das características originais do SQL foram inspiradas na álgebra relacional.
- 🐞 O SQL foi desenvolvido originalmente no início dos anos 70 nos laboratórios da **IBM** em San Jose, dentro do projeto **System R**, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional proposto por **E. F. Codd**. O nome original da linguagem era **SEQUEL**, acrônimo para "**Structured English Query Language**" (Linguagem de Consulta Estruturada em Inglês), vindo daí o fato de, até hoje, a sigla, em inglês, ser comumente pronunciada "síquel" ao invés de "és-kiú-él", letra a letra. No entanto, em português, a pronúncia mais corrente é a letra a letra: "ése-quê-éle".

Introdução

- ✧ **A linguagem SQL é um grande padrão de banco de dados.** Isto decorre da sua simplicidade e facilidade de uso. Ela se diferencia de outras linguagens de consulta a banco de dados no sentido em que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele. Ela é uma linguagem declarativa em oposição a outras linguagens procedurais. Isto reduz o ciclo de aprendizado daqueles que se iniciam na linguagem.
- ✧ Embora o SQL tenha sido originalmente criado pela **IBM**, rapidamente surgiram vários "dialetos" desenvolvidos por outros produtores. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela **American National Standards Institute (ANSI)** em 1986 e **ISO** em 1987.

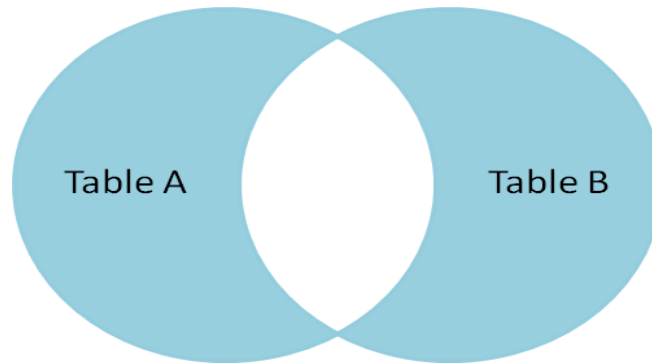
Introdução

- ❧ O **SQL** foi revisto em 1992 e a esta versão foi dado o nome de **SQL-92**. Foi revisto novamente em 1999 e 2003 para se tornar **SQL:1999** (SQL3) e **SQL:2003**, respectivamente.
- ❧ O **SQL:1999** usa ***expressões regulares*** de emparelhamento, *queries* recursivas e gatilhos (*triggers*). Também foi feita uma adição controversa de tipos não-escalados e algumas características de ***orientação a objeto***.
- ❧ O **SQL:2003** introduz características relacionadas ao **XML**, seqüências padronizadas e colunas com valores de auto-generalização (inclusive colunas-identidade).
- ❧ Tal como dito anteriormente, o **SQL**, embora padronizado pela ANSI e ***ISO***, possui muitas variações e extensões produzidos pelos diferentes fabricantes de sistemas gerenciadores de bases de dados. Tipicamente a linguagem pode ser migrada de plataforma para plataforma sem mudanças estruturais principais.

Subdivisões

🐞 A linguagem SQL se subdivide basicamente em três tipos:

- **DML – Data Manipulation Language;**
- **DDL – Data Definition Language;**
- **DCL – Data Control Language;**
- **DTL – Data Transaction Language;**



Subdivisões: DDL

- ↷ Conhecida como Data Manipulation Language ou, em português, como Linguagem de Manipulação de Dados;
- ↷ É a subdivisão da linguagem mais importante de todas por conter os elementos mais utilizados no funcionamento das aplicações;
- ↷ A **DML** é um subconjunto da linguagem usada para selecionar, inserir, atualizar e apagar dados.
- ↷ Dentro da **DML** temos os seguintes elementos:

- **SELECT**
- **DELETE**
- **INSERT**
- **UPDATE**

Para a linguagem **SQL** utilizada no MySQL, temos mais elementos dentro da DML, os quais veremos mais à frente!

Subdivisões: DDL

↻ O segundo grupo é a **DDL** (Data Definition Language - Linguagem de Definição de Dados). Uma **DDL** permite ao usuário definir tabelas novas e elementos associados. A maioria dos bancos de dados de **SQL** comerciais tem extensões proprietárias no **DDL**.

↻ Alguns sistemas de banco de dados usam o comando **ALTER**, que permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente. outros comandos **DDL**:

- **ALTER**
- **DROP**
- **CREATE**
- **RENAME**

Para a linguagem **SQL** utilizada no MySQL, temos mais elementos dentro da DDL, os quais veremos mais à frente!

Subdivisões: DDL

- ↪ CREATE TABLE LIKE ...
- ↪ ALTER TABLE ... ADD | DROP
- ↪ RENAME TABLE TO ...

Subdivisões: DCL

↗ O terceiro grupo é o **DCL** (Data Control Language - Linguagem de Controle de Dados). **DCL** controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

↗ No MySQL, tais elementos tem o nome de *Account Management Statements*, provê muito mais comandos do que a **SQL** original, definida e por **Codd** e padronizada pelo **ANSI**.

- **GRANT**
- **REVOKE**

Para a linguagem **SQL** utilizada no MySQL, temos mais elementos dentro da DCL, os quais veremos mais à frente!

Subdivisões: DTL

- ✧ A **DTL** é uma parte da linguagem **SQL** que abriga elementos utilizados em transações de bancos de dados, geralmente utilizados em meio a programas armazenados no SGBD, vinculados ou não a um banco de dados. Tais programas são chamados também de procedimentos armazenados ou **Stored Procedures*;
- ✧ O **MySQL** dá suporte em 100% ao modelo **ACID**, no tocante à transações e por consequência, à todos os elementos definidos na **SQL** padrão **ANSI**, que são:
 - **BEGIN WORK** (ou **START TRANSACTION**, dependendo do dialeto **SQL**) pode ser usado para marcar o começo de uma transação;
 - **COMMIT** torna permanentes os dados modificados em meio a uma transação.
 - **ROLLBACK** faz com que as mudanças nos dados existentes desde que o último **COMMIT** ou **ROLLBACK** sejam descartadas.

DML: SELECT

- ↪ O primeiro e mais importante elemento da **DML** que veremos é o elemento **SELECT**. Muito utilizado por ser o elemento que nos possibilita recuperar dados em meio a uma query – se pronuncia “*kuiêri*” - ou consulta, no português;

```
SELECT campo1, campo2, campon...  
FROM tabela1, tabela2, tabelan  
WHERE condição  
GROUP BY campo1, campo2, campon...  
HAVING condição sobre o agrupamento  
ORDER BY campo1 { ASC | DESC }, campo2 { ASC | DESC }...  
LIMIT number_start, count_register
```

A seguir, estudaremos cada cláusula que forma um **SELECT**.

SELECT

🐞 **SELECT**: aqui são abordadas as colunas da tabela, das quais queremos recuperar os dados. Os nomes dos campos devem ser escritos separados por vírgula, como o exemplo abaixo:

SELECT campo1, campo2, campon...

...ainda existe uma maneira de utilizar o * - *asterisco* – para selecionarmos todas as colunas da tabela, como no exemplo abaixo:

SELECT *

...vale salientar que não é uma boa utilizar o * - *asterisco* - quando se deseja obter valores de apenas algumas colunas de todo o conjunto.

FROM

- Esta cláusula da sintaxe do elemento **SELECT** é responsável por passar ao *parser* do SGDB, qual ou quais são as tabelas onde estão os campos mencionados na lista de seleção;

```
SELECT * FROM tabela;
```

- O **FROM** pode apresentar *subqueries;

```
SELECT LENGTH(data) AS CountCharData  
FROM (SELECT NOW() AS data) AS Tmp;
```

WHERE

- Na cláusula **WHERE** são aplicados filtros em forma de condição para recuperação da informação;
- Condições podem ser expressões;
- Condições não aceitam **alias*es;
- Condições podem apresentar funções;
- WHERE** pode ser utilizado em **SELECT**, **UPDATE** e **DELETE**. Para evitar acidentes, inicie o `mysqld` com a opção `--safe-updates`;
- Após obter o conjunto ou os conjuntos que respondem à consulta, aplica o filtro e os dados são retornados;

WHERE

Utilizando o banco de dados World, disponibilizado para download pela MySQL AB, suponhamos que temos a seguinte consulta:

```
SELECT Code, Name, Population, Capital  
FROM Country
```

...podemos combinar, por exemplo, os seguintes filtros para retorno dos dados:

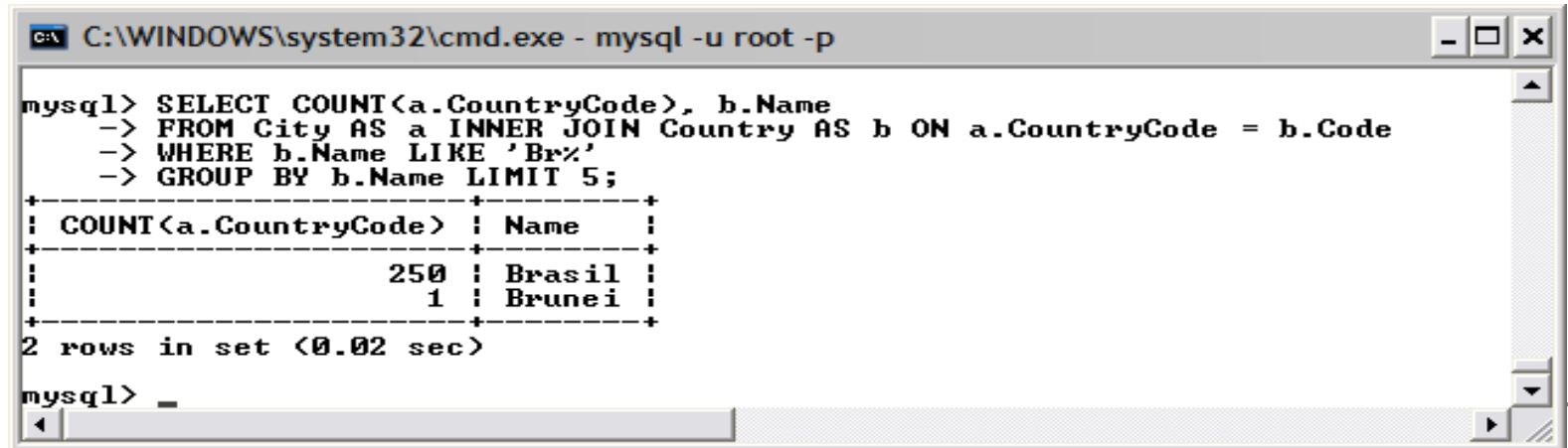
```
WHERE Name = 'Brazil' ;  
WHERE Continent = 'South America' ;  
WHERE (Population > 1000) AND (Population < 10000) ;  
WHERE ISNULL(Capital) ;  
WHERE LifeExpectancy IS NULL;
```

GROUP BY

- ↯ A cláusula **GROUP BY** é utilizada quando desejamos obter dados sumarizados, ou seja, quando queremos obter dados agrupados por uma ou mais colunas;
- ↯ Nesta cláusula, podemos abordar uma ou mais colunas;
- ↯ As colunas que deverão estar presentes em **GROUP BY** podem estar ou não na lista de colunas da cláusula **SELECT**;
- ↯ Deve-se abordar em **GROUP BY** pelo menos uma coluna que não faz parte da *função agregada*, utilizada na cláusula **SELECT**;
- ↯ Funções de agregação são:
 - **COUNT(*)**, **COUNT(field)**, **AVG()**, **SUM()**, **MAX()**, **MIN()**, **GROUP_CONCAT()**, **STD()** e **STDEV()**.

GROUP BY

Digamos que, utilizando o banco de dados World, disponibilizado para download pela MySQL AB, queiramos selecionar quantas cidades temos para cada país, utilizando somente a tabela país:



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> SELECT COUNT(a.CountryCode), b.Name
-> FROM City AS a INNER JOIN Country AS b ON a.CountryCode = b.Code
-> WHERE b.Name LIKE 'Br%'
-> GROUP BY b.Name LIMIT 5;
+-----+-----+
| COUNT(a.CountryCode) | Name   |
+-----+-----+
| 250                  | Brasil |
| 1                    | Brunei |
+-----+-----+
2 rows in set (0.02 sec)

mysql> _
```

veremos todos os detalhes.

GROUP BY

- ↗ A cláusula **GROUP BY** ainda apresenta uma outra cláusula que é **GROUP BY ... WITH ROLLUP**, que pode totalizar os dados para cada combinação de colunas na cláusula **GROUP BY** para produzir dados resumidos ou ainda *super-agregados*;
- ↗ No resultado de uma consulta com **GROUP BY ... WITH ROLLUP**, **NULL** representa o fim de cada resumo;

```
SELECT a.Continent, a.Name, SUM(a.Population)
FROM Country a
WHERE a.Continent = 'South America'
GROUP BY a.Continent, a.Name
WITH ROLLUP;
```

GROUP BY

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> SELECT a.Continent, a.Name, SUM(a.Population)
-> FROM Country a
-> WHERE a.Continent = 'South America'
-> GROUP BY a.Continent, a.Name
-> WITH ROLLUP;
```

Continent	Name	SUM(a.Population)
South America	Argentina	37032000
South America	Bolivia	8329000
South America	Brasil	170115000
South America	Chile	15211000
South America	Colombia	42321000
South America	Ecuador	12646000
South America	Falkland Islands	2000
South America	French Guiana	181000
South America	Guyana	861000
South America	Paraguay	5496000
South America	Peru	25662000
South America	Suriname	417000
South America	Uruguay	3337000
South America	Venezuela	24170000
South America	NULL	345780000
NULL	NULL	345780000

```
16 rows in set (0.00 sec)
```

HAVING

- A cláusula **HAVING** é utilizada para filtrar dados de agrupamento;
- Podemos abordar um *alias* de uma coluna em **HAVING**;
- Somente poderá ser utilizado quando o **SELECT** apresentar **GROUP BY**;

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> SELECT SUM(a.Population) AS SumOfPopulation, a.Continent, a.Name
-> FROM Country a
-> WHERE a.Continent = 'South America'
-> GROUP BY a.Continent, a.Name
-> HAVING SumOfPopulation > 300000000;

+-----+-----+-----+
| SumOfPopulation | Continent | Name      |
+-----+-----+-----+
| 37032000        | South America | Argentina |
| 170115000        | South America | Brasil    |
| 42321000         | South America | Colombia  |
+-----+-----+-----+

3 rows in set (0.00 sec)

mysql>
```

ORDER BY

- ✧ **ORDER BY** é a penúltima cláusula de um **SELECT**, aplicada na ordenação do conjunto recuperado. Antes que o conjunto seja retornado, a ordenação solicitada é aplicada;
- ✧ Dependendo do *conjunto-resultado* a ser recuperado, pode apresentar problemas graves de performance;
- ✧ Aceita vários campos;
- ✧ Os campos abordados não precisam fazer parte da lista de colunas do **SELECT**;
- ✧ O padrão sempre será **ASC** – ascendente – caso contrário, podemos abordar **DESC** para resultados ordenados de forma decrescente;
- ✧ Podem ser abordados *alias* de colunas;
- ✧ Podemos identificar colunas para a ordenação pelos sua posição na lista de seleção – 1, 2, 3...;

ORDER BY

Utilizando o banco de dados World, disponibilizado para download pela MySQL AB, suponhamos que temos a seguinte consulta:

```
SELECT Name, (Population) AS QtdPopulation  
FROM City
```

...temos os seguintes exemplos de ordenação do resultado da consulta com a cláusula **ORDER BY**:

```
ORDER BY CountryCode ASC;
```

```
ORDER BY Name ASC, QtdPopulation DESC;
```

```
ORDER BY ID; -- coluna não abordada na seleção
```

LIMIT

- ↪ A cláusula **LIMIT** é utilizada determinada a quantidade de linhas a serem exibida a partir do resultado obtido pela consulta e já ordenado;
- ↪ Pode ser utilizada com um ou opcionalmente com dois parâmetros. O primeiro diz *quantas linhas* retornar e o segundo diz, em conjunto com o primeiro, *quantas linhas retornar a partir do registro x*;
- ↪ Bom para sistemas e/ou aplicações que utilizam paginação de resultados;
- ↪ A utilização de **LIMIT** não impede que todos os registros da tabela sejam lidos, o que pode apresentar problemas de performance quando há muitas leituras;

LIMIT

Utilizando o banco de dados World, disponibilizado para download pela MySQL AB, suponhamos que temos a seguinte consulta:

```
SELECT CountryCode, Name, Population  
FROM City  
WHERE CountryCode = 'BRA'
```

...podemos ter os seguintes exemplos de utilização da cláusula **LIMIT**:

```
LIMIT 5;
```

```
LIMIT 0, 5;
```

```
LIMIT 10, 5;
```


Operadores

Operadores Lógicos

- **AND**

E lógico. Avalia as condições e devolve um valor verdadeiro caso ambos sejam corretos.

- **OR**

OU lógico. Avalia as condições e devolve um valor verdadeiro se algum for correto.

- **NOT**

Negação lógica. Devolve o valor contrário da expressão.

Operadores

Operadores Relacionais

- < Descrição – Menor que
- > Descrição – Maior que
- <> Descrição – Diferente de
- <= Descrição – Menor ou Igual que
- >= Descrição – Maior ou Igual que
- = Descrição – Igual a

BETWEEN

Utilizado para especificar um intervalo de valores.

LIKE

Utilizado na comparação de um modelo e para especificar registros de um banco de dados. "**Like**" + extensão % vai significar buscar todos resultados com o mesmo início da extensão.

Operadores

🐞 O operador **LIKE** pode ser utilizado para buscas específicas, onde utilizamos o sinal % como coringa de busca. A posição do coringa determina aonde deverá constar a string especificada;

```
SELECT Name FROM City WHERE Name
```

...podemos fazer as seguintes buscas com **LIKE**:

```
LIKE '%P' ; -- todas os nomes iniciados com P
```

```
LIKE 'P%' ; -- todas os nomes finalizados com P
```

```
LIKE 'P_u%' ; -- todos os nomes que tem a primeira letra P e a  
terceira letra u
```

Operadores

IS NULL

Seleciona registros que tem o valor de uma coluna igual a NULL

```
SELECT Capital FROM Country WHERE Capital IS NULL;
```

IS NOT NULL

Seleciona registros que não tem valor NULL em uma coluna

```
SELECT Capital FROM Country WHERE Capital IS NOT NULL;
```

Funções Agregadas

As funções de soma se usam dentro de uma cláusula **SELECT** em grupos de registros para devolver um único valor que se aplica a um grupo de registros.

- **AVG ()**

Utiliza para calcular a media dos valores de um campo determinado.

- **COUNT (field) ou COUNT (*)**

Utilizada para devolver o número de registros da seleção.

- **SUM ()**

Utilizada para devolver a soma de todos os valores de um campo determinado.

Funções Agregadas

- **MAX ()**

Utilizada para devolver o valor mais alto de um campo especificado.

- **MIN ()**

Utilizada para devolver o valor mais baixo de um campo especificado

Exercícios

- Utilizando o banco de dados [World](#), implementado e oferecido pela MySQL AB para download no site mysql.com para fins educacionais, resolva todas as questões da [LISTA 1](#).

DML: Sintaxe Geral

🐟 O próximo elemento é o **INSERT**, o qual nos permite inserir dados em tabelas de um banco de dados. Tal elemento nos permite duas sintaxes:

```
INSERT INTO tabela (campo1, campo2, campo3)
VALUES (valor1, valor2, valor3);
```

...ou ainda:

```
INSERT INTO tabela SET campo1 =valor1,
                      campo2 =valor2,
                      campo3 =valor3;
```


DML: Sintaxe Geral

Utilizamos o database chamado world, fornecido pela MySQL AB para inserir uma língua chamada 'MySQL' na tabela CountryLanguage, cujo país tem código igual a 'BRA', não é uma língua oficial e 1% da população fala tal língua.

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> desc CountryLanguage;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| CountryCode | char(3)       | NO   | PRI |          |       |
| Language    | char(30)      | NO   | PRI |          |       |
| IsOfficial  | enum('I','F') | NO   |     |          |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> insert into CountryLanguage SET CountryCode='BRA', Language = 'MySQL',
-> IsOfficial = 'F', Percentage = '1';
Query OK, 1 row affected (0.00 sec)

mysql>
```

DML: Sintaxe Geral

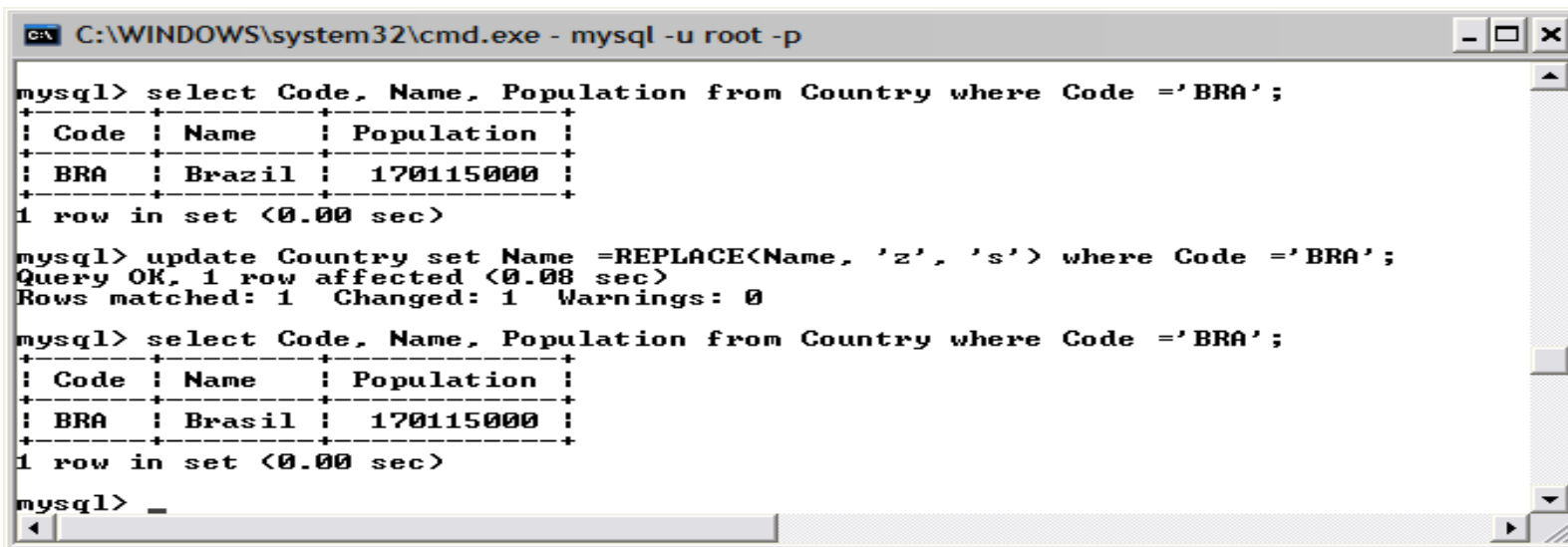
Podemos ainda, dentro da **DML**, atualizar os registros contidos em uma tabela, em um banco de dados através do elemento **UPDATE**. Tal elemento nos permite acessar uma ou mais linhas para atualizar uma ou mais colunas que compõem os registros. Sua sintaxe geral é:

– Single Table Syntax;

```
UPDATE tabela SET coluna1 =valor1, coluna2 =valor2
WHERE condicao
ORDER BY campo1 { ASC | DESC }, campo2 { ASC | DESC }
LIMIT number_start, count_register
```

DML: Sintaxe Geral

Utilizamos o database chamado world, fornecido pela MySQL AB para atualizar o nome do país 'Brazil' para 'Brasil'. Note que primeiro fazemos um **SELECT** para mostrar como está o nome do país antes e depois de **UPDATE**.



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select Code, Name, Population from Country where Code = 'BRA';
+-----+-----+-----+
| Code | Name   | Population |
+-----+-----+-----+
| BRA  | Brazil | 170115000 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> update Country set Name = REPLACE(Name, 'z', 's') where Code = 'BRA';
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select Code, Name, Population from Country where Code = 'BRA';
+-----+-----+-----+
| Code | Name   | Population |
+-----+-----+-----+
| BRA  | Brasil | 170115000 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

DML: Sintaxe Geral

Outro elemento importante de dentro da **DML** é o elemento **DELETE**, que por sua vez, nos possibilita excluir registros. A sintaxe geral do elemento é:

DELETE

FROM *tabela*

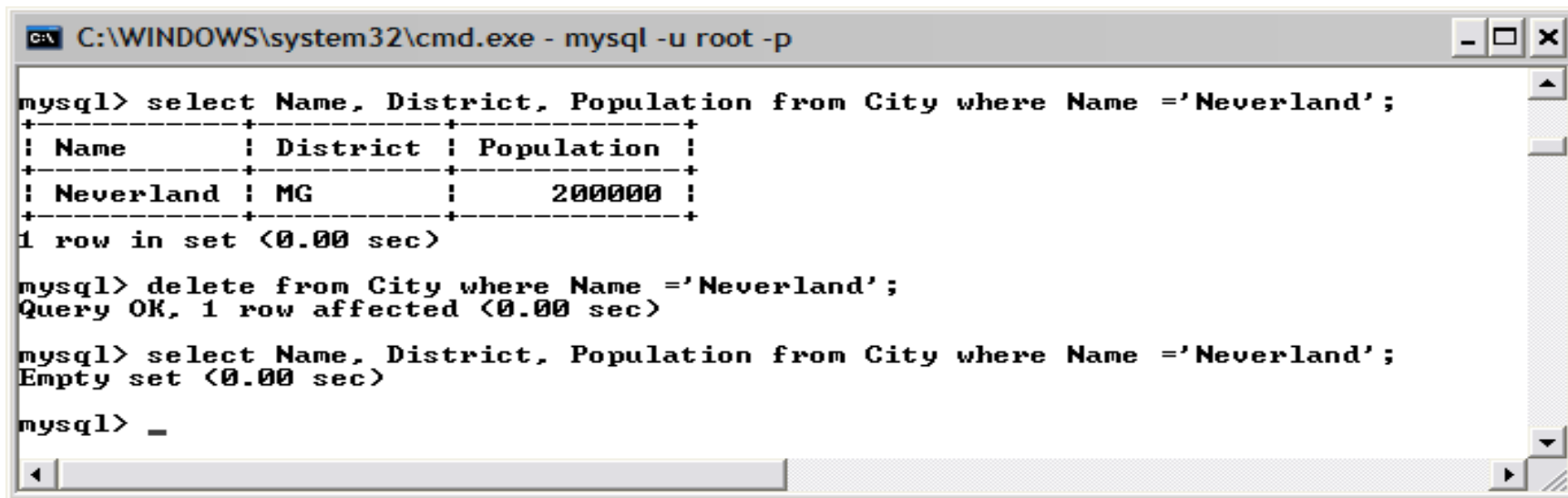
WHERE *condicao*

ORDER BY *campo1* { **ASC** | **DESC** }, *campo2* { **ASC** | **DESC** }

LIMIT *row_count*

DML: Sintaxe Geral

Utilizamos o database chamado world, fornecido pela MySQL AB, inserimos uma nova cidade na tabela City chamada Neverland, e em seguida excluimos a mesma com o elemento **DML DELETE**.



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select Name, District, Population from City where Name = 'Neverland';
+-----+-----+-----+
| Name      | District | Population |
+-----+-----+-----+
| Neverland | MG       | 2000000    |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> delete from City where Name = 'Neverland';
Query OK, 1 row affected (0.00 sec)

mysql> select Name, District, Population from City where Name = 'Neverland';
Empty set (0.00 sec)

mysql> _
```