

SQL – Structured Query Language

Introdução ao SQL

Antes de nos aprofundarmos no conhecimento do SQL, é fundamental entendermos o que é e como é estruturado um banco de dados. Neste tópico, veremos algumas noções importantes para criação de um banco de dados eficiente e bem planejado.

Apesar de o termo “banco de dados” parecer um tanto quanto técnico para a maioria das pessoas, trata-se de um conceito amplamente conhecido e empregado por quase toda população mundial. De fato, a grande maioria das pessoas hoje tem acesso a equipamentos, cuja função (principal ou secundária) é o armazenamento de informações. Quem, hoje em dia, não usa, por exemplo, um telefone celular? Desde o seu surgimento, esse tipo de aparelho possui uma agenda, na qual podemos gravar nomes e telefones para, em um segundo momento, acessá-los. Uma lista telefônica impressa também é um exemplo válido de banco de dados, pois nela são relatados todos os nomes, endereços e números de telefone das empresas e dos moradores da sua cidade e, eventualmente, dos arredores.

Introdução ao SQL

Tudo isso remete ao conceito de banco de dados, ou seja, um local no qual é possível armazenar informações, para consulta ou utilização, quando necessário.

Independentemente do aplicativo que se deseja usar para o armazenamento e manipulação das informações, todos os bancos de dados são constituídos por três elementos básicos: campos, registros e tabelas.

O que são campos?

Num banco de dados, é chamado de campo o espaço reservado para a inserção de um determinado dado.

O que são registros?

Um registro nada mais é que um conjunto de campos, ou seja, de dados sobre um determinado assunto.

O que são tabelas?

Todos os registros, isto é, as fichas que contêm os dados que armazenamos, são agrupados nas tabelas do banco de dados.

Introdução

Introdução ao SQL

Como vimos no tópico anterior, todos os sistemas de computação para o gerenciamento de informações são baseados em registros (ou *record*, em inglês). Um registro é um conjunto simples de dados com o qual se identifica determinado elemento (uma pessoa, um objeto, um evento etc.), associando a ele uma série de atributos que o caracterizam de maneira unívoca, por exemplo: nome, sobrenome e endereço, e número de telefone, no caso de uma pessoa; local, data e hora em que certo evento será realizado; código, peso, dimensões e cor, caso se trate de um objeto ou produto. Cada informação é inserida separadamente dentro de um espaço a ela reservada, chamado de campo (ou *field*, em inglês). Um exemplo de estrutura de dados pode ser vista na tabela do exemplo a seguir: cada célula da tabela é um campo, que contém apenas um tipo de informação; cada linha da tabela constitui um registro:

Introdução ao SQL

Augusto	J. Vésica	(62)9876-5432	Goiânia	GO
Nayana	Couto	(34)3216-5498	Patos de Minas	MG
Cláudio	Vésica	(62)3789-6545	Ap. de Goiânia	GO
Daniela	Savoi	(11)3558-9899	São Paulo	SP

Tabela 1.1: Registros.

A década de 1970 foi um marco na evolução dos bancos de dados digitais, pois viu o nascimento dos bancos de dados relacionais, definidos como conjuntos auto-explicativos de registros integrados. Antes disso, os bancos de dados utilizavam estruturas hierárquicas, o que tornava sua estrutura muito articulada e complexa de se consultar.

Introdução ao SQL

Para saber um pouco mais...

O modelo relacional para gerência de bancos de dados (SGBD) é um modelo de dados baseado em lógica de predicados e na teoria de conjuntos.

Historicamente, ele é o sucessor do modelo hierárquico e do modelo em rede. Estas arquiteturas antigas são até hoje utilizadas em alguns Data Centers com alto volume de dados, em que a migração é inviabilizada pelo custo que ela demandaria; existem ainda os novos modelos baseados em orientação ao objeto, que, na maior parte das vezes, são encontrados como kits de construção de SGBD, ao invés de um SGBD propriamente dito. O modelo relacional foi o primeiro modelo de banco de dados formal. Somente depois, seus antecessores, os bancos de dados hierárquicos e em rede, passaram a ser também descritos em linguagem formal.

Introdução ao SQL

O modelo relacional permite ao projetista criar um modelo lógico consistente da informação a ser armazenada. Este modelo lógico pode ser refinado através de um processo de normalização. Um banco de dados construído puramente baseado no modelo relacional estará inteiramente normalizado. O plano de acesso, outras implementações e detalhes de operação são tratados pelo sistema DBMS, e não devem ser refletidos no modelo lógico. Isto se contrapõe à prática comum para DBMSs SQL nos quais o ajuste de desempenho freqüentemente requer mudanças no modelo lógico.

Os blocos básicos do modelo relacional são: o domínio, ou tipo de dado; uma tupla, que é um conjunto de atributos que são ordenados em pares de domínio e valor; uma relvar (variável relacional), que é um conjunto de pares ordenados de domínio e nome que serve como um cabeçalho para uma relação; e uma relação, que é um conjunto desordenado de tuplas. Apesar destes conceitos matemáticos, eles correspondem basicamente aos conceitos tradicionais dos bancos de dados. Uma relação é similar ao conceito de tabela e uma tupla é similar ao conceito de linha.

Introdução ao SQL

O princípio básico do modelo relacional é o princípio da informação: toda informação é representada por valores em relações.

Assim, as relvars não são relacionadas umas às outras no momento do projeto. Entretanto, os projetistas utilizam o mesmo domínio em vários relvars, e se um atributo é dependente de outro, esta dependência é garantida através da integridade referencial.

Introdução ao SQL

Como todos os bancos de dados, o relacional também tem sua estrutura baseada em registros relacionados e organizados em tabelas. Essas relações tornam os registros integrados.

Um subconjunto específico de dados constitui um dicionário de dados, que determina a maneira em que os registros de dados são relacionados, quais vínculos existem para a sua utilização e outras características. Esses dados descritivos são chamados “metadados” e é graças a eles que um banco de dados é autodescritivo, ou seja, contém uma descrição de sua própria estrutura.

Introdução ao SQL

A presença de todos esses elementos (dados e metadados) transforma o banco de dados em uma estrutura informática acessível por qualquer aplicação que seja capaz de explorar os metadados, extrair informações sobre os dados e tratá-los para copiá-los, alterá-los e criar novos.

A característica autodescrevente de um banco de dados faz com que a aplicação que o acessa não precise gerenciar a estrutura dos registros, e sim, limite-se a utilizá-los, pois o próprio banco de dados se encarregará de criar espaço para novos registros, alterando seu conteúdo de acordo com as solicitações da aplicação que está acessando-o.

Introdução ao SQL

Assim, os bancos de dados com essas características não são simplesmente conjuntos de dados, mas contêm mecanismos automatizados que se encarregam da gestão dos registros, por esse motivo são chamados **Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR)**, ou **Relational Database Management Systems (RDMS)**.

Em um banco de dados relacional, as informações são estruturadas em tabelas, divididas em linhas e colunas: as linhas são os registros e as colunas constituem os campos.

Há diversas razões para que o modelo de banco de dados relacional substituisse o modelo hierárquico: primeiramente porque o SGBDR permite adicionar com facilidade novas tabelas em um banco de dados, relacionando seus dados e sem afetar substancialmente a estrutura do banco de dados e sem a necessidade de alterar os aplicativos que trabalharão com as tabelas; em segundo lugar, num

Introdução ao SQL

banco de dados relacional é muito simples alterar a estrutura de uma ou mais tabelas, inserindo ou excluindo linhas e colunas de acordo com as necessidades sem comprometer a funcionalidade do banco de dados; outra razão relevante do sucesso dos SGBDR é que todos eles podem ser acessados utilizando uma única ferramenta, chamada Structured Query Language (SQL), disponível praticamente para todas as plataformas de hardware, desde os mainframes até computadores portáteis.

THE SEQUEL TO SQL

rito

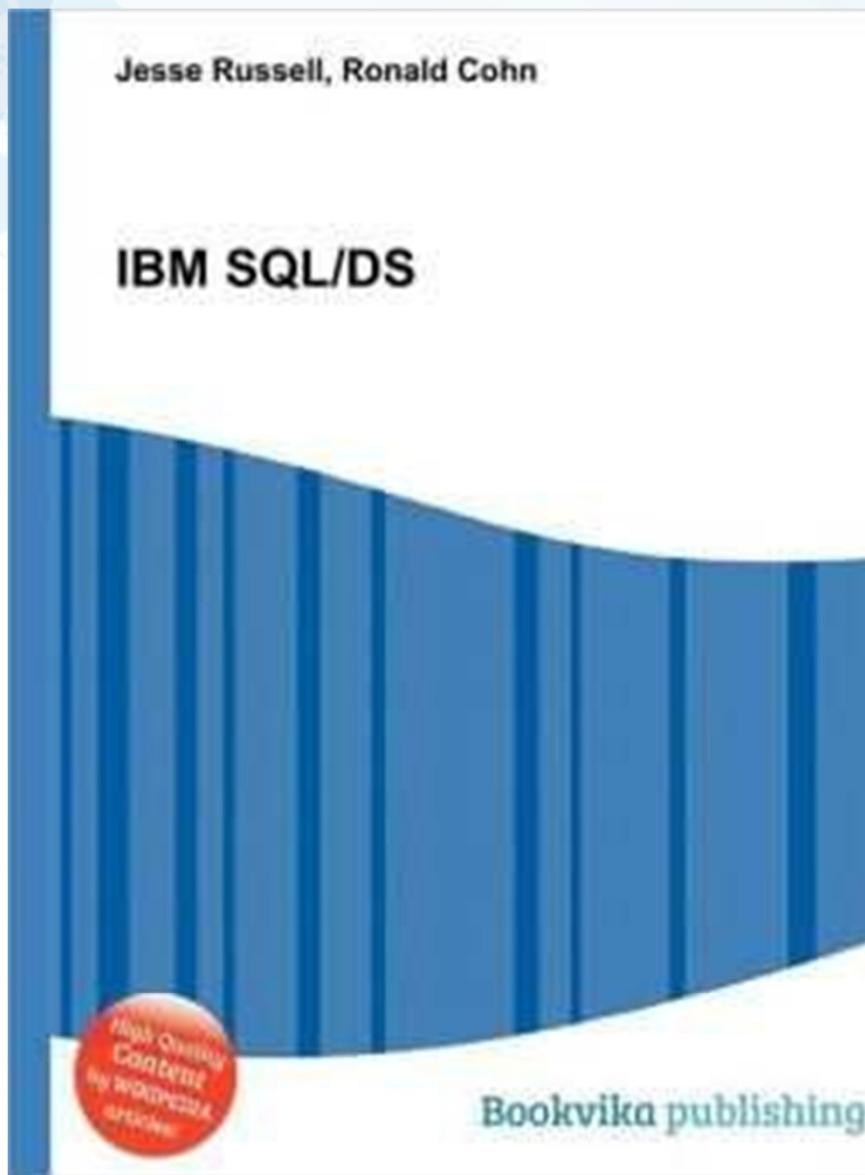


Introdução ao SQL

A linguagem SQL deriva diretamente do SEQUEL (*Structured English Query Language*).

O SEQUEL tinha como objetivo se tornar uma linguagem universal (cujos comandos originaram do idioma inglês), usada para acessar os dados armazenados em um banco de dados, ignorando as complicações derivadas da estrutura interna de funcionamento de cada um.

A primeira versão a ser usada foi distribuída em 1981 pela IBM e foi denominada Structured Query Language / Data System, ou simplesmente SQL/DS.



Introdução ao SQL

Esta nova linguagem despertou o interesse dos maiores construtores de computadores de grande porte (mainframes) e das empresas que desenvolviam softwares para aqueles equipamentos, pois, graças ao SQL/DS, seria possível simplificar os processos de manipulação de grandes quantidades de dados, que naquela época ainda empregavam estruturas hierárquicas.

Nos anos seguintes, o SQL deixou de ser um produto exclusivo da IBM e se tornou, de fato, uma linguagem de acesso a bancos de dados muito articulada e funcional, que pode ser empregada em computadores de arquiteturas totalmente diferentes.

Introdução ao SQL

Conceitos básicos do SQL

Os computadores trabalham executando programas escritos em uma linguagem de programação específica. As linguagens de programação são conjuntos de comandos e palavras-chave que devem ser utilizados respeitando regras de composição predefinidas, que constituem a sintaxe.

No decorrer dos anos, desde o surgimento da informática, foram criadas inúmeras linguagens de programação, cada qual com seus comandos e sintaxe, para tornar mais viável a interação do usuário

Introdução ao SQL

com o computador. Entre as linguagens mais conhecidas, podemos citar: FORTRAN, CLIPPER, COBOL, PASCAL, BASIC, VB, DELPHI, C (e seus derivados), JAVA etc. Em todos os casos, estamos falando de linguagens que permitem ao computador realizar qualquer tipo de operação (cálculos, armazenar e manipular dados, desenhar, digitar textos etc.). Para isso, nos oferecem comandos por meio dos quais o programador deve instruir o computador sobre como cada tarefa deve ser realizada.

Introdução ao SQL

No caso do SQL, estamos falando de uma linguagem um pouco diferente, antes de tudo porque pode ser usada exclusivamente para acessar dados em um banco de dados, isto é, trata-se de uma linguagem específica para a manipulação de tabelas de dados. Além disso, o SQL não serve para criar rotinas de procedimentos a serem executados pelo computador, e sim, para informar quais dados (ou conjuntos de dados) queremos manipular. De fato, a finalidade do SQL é acessar dados, independentemente do tipo de hardware ou software que estamos usando.

Introdução ao SQL

Tabelas, consultas e views

A linguagem SQL é baseada no conceito de views (visões, em português). Uma view é uma tabela virtual cujos registros são constituídos por campos que, no banco de dados físico, podem pertencer a registros localizados em tabelas diferentes.

Utilizando o SQL, podemos criar views realizando queries, ou seja, pesquisas, nas tabelas físicas existentes no banco de dados. O termo query significa: pergunta, interrogação e, na linguagem SQL, indica uma requisição precisa de dados a serem extraídos a partir de um banco de dados. Em outras palavras, uma query é uma lista de elementos (campos) que contém os dados que desejamos obter.

Introdução ao SQL

A partir da extração de dados de uma ou mais tabelas, é gerada uma nova tabela virtual (view), que contém os campos filtrados de acordo com os critérios especificados. Definimos uma view como uma tabela virtual porque, quando é gerada, não é armazenada em disco, mas apenas visualizada. Essa característica permite imprimi-la em papel ou torná-la uma tabela física pertencente ao banco de dados, gravando-a junto com as demais tabelas que constituem a estrutura do banco de dados. O SQL permite também operar não apenas com dados existentes: é possível criar novas tabelas criando novos campos ou, ainda, formá-las por campos obtidos a partir de

Introdução ao SQL

cálculos ou operações feitas em campos de outras tabelas. Essas tabelas são chamadas de “tabelas de base” e são físicas, ou seja, ao serem geradas são gravadas fisicamente no banco de dados.

A linguagem SQL é um conjunto de comandos (ou instruções) que permitem gerar enunciados, ou seja, linhas de comandos compostas por uma ou mais instruções. Alguns comandos permitem ou até mesmo exigem o uso de parâmetros adicionais, chamados de cláusulas e predicados. Basicamente, todos os comandos do SQL são verbos em inglês (`select`, `create`, `alter` etc.), e as cláusulas são preposições do mesmo idioma (`from`, `as`, `by`, `in` etc.).



Introdução ao SQL

Grupos de comandos SQL

Os comandos do SQL são classificados em três grupos, de acordo com suas principais funções:

- **DDL (Definition Data Language)**: são todos aqueles comandos usados para criar e alterar tabelas que compõem o banco de dados, ou seja, os comandos que definem a estrutura dos dados;
- **DML (Data Manipulation Language)**: pertencem a este grupo todos os comandos usados para extrair informações das tabelas, ou seja, para manipular os dados existentes;
- **DCL (Data Control Language)**: trata-se de um conjunto de comandos usado em sistemas multiusuário para definir os privilégios de acesso aos dados a cada usuário. Os comandos de controle de acesso aos dados são usados para implementar segurança e privacidade em bancos de dados.

Introdução ao SQL

Os dialetos do SQL

Como já dissemos, o SQL é uma linguagem que possui seus comandos, seus parâmetros e sua sintaxe específicos. A integridade dessas características é constantemente monitorada e mantida por um comitê, o *American National Standards Institute* (algo como Instituto Americano de Padrões Nacionais), mais conhecido como ANSI. O primeiro padrão para o SQL foi publicado em 1986, e em 1989 teve sua primeira atualização. Em 1992, houve outra atualização, que se tornou conhecida como SQL92, ou simplesmente SQL2. A última atualização é de 1999, mas foi publicada oficialmente em 2000, com o nome SQL99 (ou SQL3).

Introdução ao SQL

Essa última versão foi assumida como padrão também por outra instituição, a ISO (*International Standards Organization – Organização de Padrões Internacionais*).

A definição de padrões para os comandos, os parâmetros e a sintaxe do SQL é, sem dúvida alguma, um fato positivo, pois evita divergências entre as atualizações lançadas que devem respeitar as diretrizes estabelecidas pelo ANSI e pela ISO. Contudo, os produtores de softwares lançam com freqüência versões próprias desta linguagem, adicionando ao SQL novos comandos e funcionalidades.

Introdução ao SQL

Essas versões são chamadas “dialetos” e, além dos comandos básicos do SQL (cerca de 50 instruções), trazem outros que, em alguns casos, fogem do propósito inicial do SQL, que nasceu como linguagem para a manipulação de dados em bancos de dados. Assim, é possível encontrar dialetos SQL que trazem comandos para a criação de estruturas condicionais (IF...THEN...ELSE...), típicos das linguagens de programação para desenvolvimento de softwares, ou instruções para estruturas de controle (WHILE...) etc.

Para os puristas do SQL, tratam-se de alterações que descaracterizam a linguagem, pois tentam transformá-la em algo para o qual não foi criada. De qualquer forma, os dialetos apresentam novas funcionalidades que visam sempre aprimorar a manipulação de bancos de dados, desde alterações simples até mais complexas.

Introdução ao SQL

Introdução

Por incrível que pareça, a linguagem SQL não trazia, originalmente, nenhum comando para a criação de um banco de dados. Isto pode parecer ilógico, mas existe uma explicação: o SQL é uma ferramenta para criar, gerenciar e utilizar tabelas, enquanto o banco de dados é o objeto (software) que as contêm. De fato, não faz parte das finalidades do SQL definir características e funcionamento de um banco de dados, mas apenas manipular dados contidos dentro de tabelas.

Todavia, precisamos de um banco de dados para, em seguida, estruturar as tabelas e inserir os dados a serem armazenados, por isso, todas as implementações do SQL trazem pelo menos um comando para criar um banco de dados vazio e algumas ferramentas para excluir bancos de dados existentes.

Introdução ao SQL

Ao criar um banco de dados como MySQL em ambiente Windows, é criada apenas uma pasta vazia, dentro da qual serão armazenados os arquivos gerados utilizando os comandos para criar tabelas.

Em outros SGBDR (Sistemas para o Gerenciamento de Bancos de Dados Relacionais), como o Oracle ou o Microsoft SQL Server, por exemplo, o comando para a criação de um novo banco de dados gera estruturas de dados complexas. Todos os procedimentos e exemplos dos tópicos deste capítulo serão baseados no software MySQL.

Introdução ao SQL

Na medida em que digitamos uma linha de comando, a interface do MySQL Control Center destaca na cor azul e em negrito as palavras reconhecidas como palavras-chave do SQL, deixando as demais palavras em estilo normal e na cor cinza. As palavras-chave podem ser digitadas em maiúsculo ou minúsculo, sem problemas, tanto no MySQL como nos demais softwares para o uso desta linguagem.

No decorrer dos exemplos deste livro, usaremos tipos de fonte diferentes para representar as linhas de código para realizarmos operações em bancos de dados com o Structured Query Language (SQL). Veja alguns exemplos:

- Todas as palavras-chave das instruções SQL serão escritas em maiúsculo, como mostrado a seguir:

Criar Tabela

```
CREATE TABLE
```

- Os elementos, ou parâmetros, especificados para cada comando aparecerão em itálico, como no exemplo a seguir:

```
CREATE TABLE nome_da_tabela
```

- Parâmetros ou comandos opcionais, isto é, instruções adicionais dadas aos comandos, aparecerão em maiúsculo e dentro de colchetes, assim:

```
CREATE TABLE [IF NOT EXISTS] nome_da_tabela
```

- Quando, em uma instrução, podemos utilizar um entre vários parâmetros alternativos, todas as opções serão apresentadas e separadas por barras verticais (|). Caso esses parâmetros sejam opcionais, serão listados entre colchetes, como no exemplo a seguir:

Deletar Tabela

```
DROP TABLE nome_da_tabela [RESTRICT | CASCADE]
```

- Nos casos em que é obrigatório escolher um parâmetro dentro de uma série de opções, a lista das alternativas será exibida dentro de chaves, e cada parâmetro disponível será separado dos outros por barras verticais, como mostrado a seguir:

Alterar Tabela

```
UPDATE {nome_da_tabela | nome_da_view}
```

- Em algumas circunstâncias, é possível que um mesmo elemento possa ser repetido diversas vezes; nesses casos, usaremos a seguinte representação:

Col1, col2, ...

Quando, no meio de uma linha de comando, encontrarmos parênteses e vírgulas, esses elementos constituem parte integrante da linha de comando e devem ser escritas exatamente como apresentado no livro. Já os colchetes [] e as chaves { } não deverão ser digitados (veja o significado desses elementos nos parágrafos anteriores).

Criando um banco de dados

Criando um novo banco de dados

Vejamos, então, como se cria um novo banco de dados utilizando o MySQL. Siga os passos a seguir:

1. Ative o **MySQL Command Line Client** e, ao ser solicitado, digite a sua senha de acesso.
2. No Prompt do MySQL, digite a seguinte linha de comando:

```
mysql> CREATE DATABASE Teste; [Enter]
```

Lembre-se de finalizar sempre as linhas de comando com o caractere ponto-e-vírgula.

O novo banco de dados foi criado e o MySQL nos informa disso exibindo uma mensagem (em inglês) logo depois da instrução que digitamos:

Criando um banco de dados

```
mysql> CREATE DATABASE Teste;  
Query OK, 1 row affected (0.08 sec)  
  
mysql>
```

3. Podemos verificar rapidamente a existência do banco de dados recém-criado, bem como a de todos os outros criados anteriormente, utilizando a instrução SHOW DATABASES (mostrar bancos de dados); para isso, digite a seguinte linha de comando:

```
mysql> SHOW DATABASES;      [Enter]  
+-----+  
| Databases      |  
+-----+  
| information _ schema |  
| mysql          |  
| test           |  
| teste          |  
+-----+  
4 rows in set (0.05 sec)
```

Criando um banco de dados

O comando `SHOW DATABASES` é muito útil para sabermos os nomes dos bancos de dados já existentes, visto que não podemos criar dois bancos de dados com o mesmo nome. Por exemplo, se tentássemos criar novamente o banco de dados teste, receberíamos a seguinte mensagem de erro:

```
Can't create database 'Teste'. Database exists
```

A mensagem nos informa que não é possível criar um banco de dados com o nome teste, pois ele já existe.

Este tipo de erro pode ser prevenido acrescentando ao comando `CREATE` a cláusula `if not exists` (em inglês “se não existe”). Veja um exemplo:

```
mysql> CREATE DATABASE IF NOT EXISTS Teste; [Enter]
```

Excluindo um banco de dados

Excluindo um banco de dados

No tópico anterior, vimos como criar um banco de dados com o comando `CREATE DATABASE`. Vejamos agora como excluir um banco de dados existente.

É preciso ressaltar que, ao apagar um banco de dados, todas as suas tabelas e os dados nelas contidos também serão apagados e, portanto, perdidos de maneira irreversível.

Para excluir um banco de dados, usa-se o comando `DROP DATABASE`; para excluir, por exemplo, o banco de dados `Teste` criado no tópico anterior, faça o seguinte:

1. No prompt do MySQL, digite a linha de comando como mostrada a seguir:

```
mysql> DROP DATABASE Teste;      [Enter]
```

Excluindo um banco de dados

2. Será exibida a mensagem de confirmação a seguir (o valor entre parênteses é o tempo que o computador levou para realizar a operação):

```
mysql> DROP DATABASE Teste;  
Query OK, 1 row affected (0.05 sec)
```

Selecionando um banco de dados

Selecionando um banco de dados

Como vimos, podemos criar vários bancos de dados, porém, podemos manipular apenas um por vez. Assim, antes de começar, é preciso selecionar qual será o banco de dados que queremos alterar. Isso é feito utilizando o comando `USE` (“usar” em inglês), seguido pelo nome do banco de dados em questão.

Então, façamos o seguinte: vamos criar um novo banco de dados, listar os bancos de dados existentes e, por fim, selecionar o nosso para que possa ser manipulado. Siga os passos a seguir:

1. No prompt do MySQL, digite:

```
mysql> CREATE DATABASE MeuBD;      [Enter]
```

2. Recebida a mensagem de confirmação, digite:

```
mysql> SHOW DATABASES;      [Enter]
```

Selecionando um banco de dados

Que resultará na seguinte lista:

```
+-----+  
| Databases          |  
+-----+  
| information_schema |  
| meubd              |
```

3. Note que o segundo item da lista é o banco de dados recém-criado; então, digite:

```
mysql> USE MeuBD; [Enter]
```

A mensagem `Database changed` nos informa que o banco de dados foi ativado.

Criar uma Tabela no banco de dados

O comando CREATE pode ser associado também ao objeto TABLE (tabela) para criar novas tabelas no banco de dados ativo. De fato, o comando CREATE TABLE é um dos mais importantes da linguagem SQL, pois é a partir de uma nova tabela que começaremos a manipulação dos dados.

A sintaxe deste comando é a seguinte:

```
CREATE [GLOBAL TEMPORARY | LOCAL TEMPORARY] TABLE nome_da_tabela  
[ON COMMIT {PRESERVE ROWS | DELETE ROWS}] (nome_da_coluna tipo_de_dados especificações, [nome_da_coluna, tipo_de_dados especificações2,...]) | [LIKE nome_da_tabela] |  
[vínculo_tabela],...n]
```

Criar uma Tabela no banco de dados

Uma tabela deve ter um nome e pelo menos uma coluna, também identificada com um nome próprio e obrigatoriamente diferente do nome da tabela. Visto que as colunas são, na verdade, recipientes de dados, para cada tipo de coluna é preciso definir que tipo de dados conterá (números, textos etc.), especificando uma ou mais características (especificações) que identifiquem aquele determinado tipo de dados. Assim, considerando apenas as opções essenciais, a sintaxe do comando CREATE TABLE é a seguinte:

```
CREATE TABLE nome_da_tabela (nome_coluna1 tipo_de_dados especificações, [nome_coluna2 tipo_de_dados especificações,...])
```

Criar uma Tabela no banco de dados

Com base nas informações que possuímos até o momento, vamos criar uma tabela, que chamaremos de Cadastro, cuja estrutura é composta por duas colunas (Nome e sobrenome).

Observação: lembre-se que para criar uma tabela é indispensável selecionar o banco de dados que a conterá. No tópico **Selecionando um banco de dados**, criamos e selecionamos um banco de dados vazio chamado MeuBD. Se desejar acompanhar os procedimentos descritos a seguir, você deverá estar com um banco de dados criado e ativo.

Então, siga os passos mostrados:

1. No prompt do MySQL, digite esta linha de comando:

```
mysql> CREATE TABLE Cadastro (Nome CHAR (15), Sobrenome  
CHAR (20)); [Enter]
```

Criar uma Tabela no banco de dados

2. Após termos recebido a mensagem que confirma a execução do comando (Query OK, 0 rows affected (0.16 sec)), e de voltar ao prompt do MySQL, digite a instrução a seguir para verificar a existência da tabela criada:

```
mysql> SHOW TABLES; [Enter]
```

Será exibida a lista mostrada a seguir:

```
+-----+
| Tables_in_meuBD |
+-----+
| cadastro         |
+-----+
1 row in set (0.01 sec)
```

Criar uma Tabela no banco de dados

Visualizando a estrutura das tabelas

Podemos também analisar a estrutura de uma tabela de maneira aprofundada usando o comando `DESCRIBE` (“descrever”, em inglês), seguido pelo nome da tabela. Vejamos como aplicá-lo à tabela que criamos no tópico anterior:

1. No prompt do MySQL, digite:

```
mysql> DESCRIBE Cadastro; [Enter]
```

O resultado será como mostrado a seguir:

```
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+
| Field      | | Type       | | Null     | | Key      | | Default   | | Extra    |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+
| Nome       | | char(15)  | | YES      | |          | | NULL     | |          |
| Sobrenome  | | char(20)  | | YES      | |          | | NULL     | |          |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+
2 rows in set (0.04 sec)
```

Inserindo dados em uma Tabela

Inserindo dados em uma tabela

A tabela Cadastro, que criamos nos tópicos anteriores, ainda não contém dados, pois definimos a sua estrutura, mas ainda não realizamos a inserção de valores nas colunas. Para inserir dados nesta e em outras tabelas é preciso usar o comando `INSERT` ("inserir", em inglês), cuja sintaxe é basicamente a seguinte:

```
INSERT nome_da_tabela VALUES (valor1, valor2,...)
```

Inserindo dados em uma Tabela

Veja um exemplo para entender melhor:

Vamos inserir na nossa tabela Cadastro o primeiro item, ou seja, a primeira pessoa:

1. Lembre-se que a nossa tabela é constituída por dois campos (Nome e Sobrenome), ambos do tipo CHAR (texto); assim, para inserir, respectivamente, o nome e o sobrenome da primeira pessoa, digite o enunciado mostrado a seguir:

```
mysql> INSERT Cadastro VALUES ('Augusto', 'J. Vésica'); [Enter]
```

2. Em instantes, o MySQL nos retorna a mensagem de confirmação para o comando digitado:

```
Query OK, 1 row affected (0.06 sec)
```

Selecionando dados em uma Tabela

Visualizando o conteúdo de uma tabela

No tópico anterior, inserimos o primeiro registro na nossa tabela Cadastro, mas o que fazer se desejarmos ver esse conteúdo para verificar se foi digitado corretamente? É para isso que o SQL oferece o comando `SELECT`, considerado um dos comandos fundamentais da Structured Query Language (SQL). Este comando pertence ao grupo chamado *Data Manipulation Language* (DML) – Linguagem de Manipulação de Dados – e, como o próprio nome deixa imaginar, é usado para selecionar dados de uma tabela.

O comando `SELECT` é, basicamente, a ferramenta principal para consultar informações de um banco de dados, por isso, é comumente chamado de query (que, em inglês, significa “consulta”).

Sua sintaxe essencial é muito simples:

```
SELECT dados _ desejados FROM nome _ da _ tabela;
```

Selecionando dados em uma Tabela

Em que dados_desejados é aquilo que se deseja obter da tabela especificada como argumento da cláusula `FROM`; em outras palavras, dados_desejados é o critério com base no qual serão extraídos os dados da tabela.

Para definir esse critério, podemos usar uma palavra inteira, parte dela, ou, ainda, usar os caracteres especiais asterisco (*) e interrogação (?), cujos significados são os seguintes:

- **Asterisco (*)**: significa tudo, ou seja, todos os dados. Pode ser combinado com um ou mais caracteres para especificar conjuntos de dados com algo em comum, por exemplo, em geral, se digitarmos o critério A* significa que queremos ver todos os registros cujo conteúdo começa com a letra A;

Selecionando dados em uma Tabela

Vamos utilizar o comando `SELECT` para visualizar todo o conteúdo da nossa tabela. Para isso, digite a linha de comando a seguir:

```
mysql> SELECT * FROM Cadastro; [Enter]
```

Em poucos instantes, veremos o resultado, que no nosso exemplo será:

```
+-----+-----+
| Nome   | Sobrenome |
+-----+-----+
| Augusto | J. Vésica |
+-----+-----+
1 row in set (0.03 sec)
```

Tipo de dados em uma Tabela

Considerações sobre os tipos de dados para as colunas

É preciso tomar muito cuidado ao definirmos os tipos de dados para as colunas da tabela em que estamos trabalhando. De fato, se planejarmos de maneira errada as características dos dados que cada coluna irá conter, correremos o risco de perder dados sem perceber, ou então, o nosso banco de dados se tornará inviável para se trabalhar.

Tipo de dados em uma Tabela

Para entender melhor do que estamos falando, façamos um teste com a nossa tabela Cadastro: sabemos que dispomos de dois campos (Nome e Sobrenome), e que, para cada um deles, foi definido um limite de caracteres (respectivamente, 15 para o primeiro e 20 para o segundo), mas, o que aconteceria se tentássemos inserir na tabela um sobrenome com, por exemplo, 30 ou mais caracteres? Vejamos:

1. Com o banco de dados MeuBD ainda ativo e a tabela Cadastro ainda aberta, digite no prompt do MySQL o enunciado mostrado a seguir:

```
mysql> INSERT Cadastro VALUES (Luiz Henrique, Magalhães  
Figueiredo de Castro Júnior); [Enter]
```

Tipo de dados em uma Tabela

Imediatamente, receberemos do MySQL uma mensagem de erro como mostrado a seguir:

```
ERROR 1406 (22001): Data too long for column 'Sobrenome' at row 1
```

Isso significa que digitamos um valor longo demais (TOO LONG) para a coluna Sobrenome e, portanto, o registro não foi inserido.

É sabido que sobrenomes como o do nosso exemplo não são raros no nosso país, portanto, é evidente que um limite de apenas 20 caracteres para o sobrenome constitui um obstáculo para o nosso trabalho.

Para evitar erros de planejamento na estrutura das nossas tabelas, veremos neste tópico alguns detalhes importantíssimos sobre os tipos de dados que as colunas podem armazenar e as características de todos eles. A seguir, veremos um exemplo prático:

Tipo de dados em uma Tabela

Tipo de dados do MySQL	Tipo de dados do padrão SQL99	Descrição
Tinyint		Usado para números inteiros, positivos ou negativos, que vão de -128 a +127, ou entre 0 e 255, se não especificar o sinal (+/-).
VARCHAR(n)	CHARACTER VARYING(n)	Usado para seqüências alfanuméricas de comprimento variável, de até 255 caracteres.
YEAR[(2 4)]		Usado para valores de anos, com dois ou quatro dígitos, no intervalo de (19)70 até (20)69, para o formato de dois dígitos, ou de 1901 até 2155, para o formato de quatro dígitos - .

Tabela 2.1: Tipos de dados disponíveis na linguagem SQL e do MySQL, e suas respectivas descrições.

Tipo de dados em uma Tabela

Tipo de dados do MySQL	Tipo de dados do padrão SQL99	Descrição
LONGBLOB, LONGTEXT	Binary large object (BLOB)	Usado para blocos de caracteres binários ou de texto longo (até 4.294.967.295 caracteres).
MEDIUMBLOB, MEDIUMTEXT		Usado para blocos de caracteres binários ou de texto longo (até 16.444.216 caracteres).
mediumint		Usado para números inteiros, positivos ou negativos, que vão de -8.388.608 a +8.388.607.
NUMERIC(p,s)	NUMERIC(p,s)	O mesmo que DECIMAL.
REAL(p,s)	Double precision	O mesmo que DOUBLE PRECISION.
SET("Valor1", "Valor2",...)		Usado para uma seqüência de caracteres que pode ter zero ou mais valores, especificados na lista ("Valor1", "Valor2",...). A lista pode conter até 64 valores diferentes.
smallint	smallint	Usado para números inteiros, positivos ou negativos, que vão de -32.758 a +32.757.
TEXT	Linha 18 vazia	Usado para blocos de texto longo com até 65.536 caracteres.
TIME	TIME	Usado para valores de hora no formato hh:mm:ss.
TIMESTAMP(n)	Timestamp	Usado para valores de data e hora que vão de 1970-01-01 00:00:00 até 2037-12-31 23:59:59. O parâmetro <i>n</i> pode ter seu valor configurado em 14, 12, 8 ou 6 e se refere ao comprimento do valor armazenado.
TINYBLOB, TINYTEXT		Usado para valores de BLOB ou TEXT com até 255 caracteres de comprimento.

Tipo de dados em uma Tabela

Tipo de dados do MySQL	Tipo de dados do padrão SQL99	Descrição
bigint		Usado para números inteiros, positivos ou negativos, que vão de -9.223.372.036.854.775.808 a +9.223.372.036.854.775.807.
CHAR(n)	CHARACTER(n)	Usado para seqüências de caracteres de comprimento fixo, a quantidade de caracteres é definida no parâmetro <i>n</i> e pode variar de 1 a 255.
DATE	DATE	Usado para datas (no formato ano-mês-dia), dentro do intervalo de 1000-1-1 a 9999-12-31.
datetime	datetime	Usado para valores de data e hora que vão de 1000-1-1 00:00:00 a 9999-12-31 23:59:59.
DECIMAL(p,s)	DECIMAL(p,s)	Usado para números decimais não arredondados, definidos com <i>p</i> números antes da vírgula e <i>s</i> casas decimais.
DOUBLE(p,s) Double Precision	Double precision	Usado para valores numéricos com vírgula variável, com precisão dupla (até 308 posições).
ENUM("Valor1", "Valor2",...)		Usado para uma seqüência de caracteres que pode ter somente um valor escolhido dentro de uma lista de valores ("Valor1", "Valor2",...). Permite selecionar dentro de 65.535 valores diferentes.
Float	FLOAT(p)	Usado para valores numéricos com vírgula variável (até 38 posições).
INT, INTEGER	INT, INTEGER	Usado para números inteiros, positivos ou negativos, que vão de -2.147.483.548 a +2.147.483.547.