

## Interface Com o Usuário - MySQL

Edição nº1 - 2007



Francini Reitz Spanceski

---

### Apoio



### Gestão e Execução



### Conteúdo e Tecnologia



## Apresentação



Este livro didático contém a disciplina de **Interface Com o Usuário - MySQL**.

Este material tem por objetivo apresentar os conceitos que envolvem o desenvolvimento de um banco de dados. Procurará mostrar-lhe os conceitos que envolvem a modelagem de um banco de dados e uma tecnologia de gerenciamento de dados que lhe permitirá criar bancos de dados relacionados, gerenciá-los e manipulá-los.

Para sua melhor compreensão, o livro está estruturado em duas partes. Na primeira, são apresentados os conceitos que envolvem a modelagem de dados, a estrutura, os principais conceitos, sua utilização para o desenvolvimento de um Banco de Dados e exercícios que o ajudarão a fixar os conceitos apresentados e a aplicabilidade no desenvolvimento de um modelo Entidade Relacionamento. Na segunda parte, utilizaremos o MySQL para a criação de um banco de dados.

Nossa recomendação é que você verifique as datas importantes dentro do cronograma de estudos e elabore um plano de estudos pessoal. Assim você poderá aproveitar o curso ao máximo, garantindo o retorno de todo esforço investido. Estude as aulas virtuais, faça os exercícios, leia o livro-texto e faça todas as atividades propostas. Sempre que possível tire suas dúvidas com o monitor.

Lembre-se de que a sua passagem por esta disciplina será também acompanhada pelo Sistema de Ensino Tupy Virtual , seja por correio postal, fax, telefone, e-mail ou Ambiente Virtual de Aprendizagem.

Entre sempre entre em contato conosco quando surgir alguma dúvida ou dificuldade.

Toda a equipe terá a maior alegria em atendê-lo, pois a sua aquisição de conhecimento nessa jornada é o nosso maior objetivo. Acredite no seu sucesso e bons momentos de estudo!

Equipe Tupy Virtual.

## SUMÁRIO

|                                                                     |           |
|---------------------------------------------------------------------|-----------|
| <b>CARTA DO PROFESSOR.....</b>                                      | <b>4</b>  |
| <b>CRONOGRAMA DE ESTUDOS.....</b>                                   | <b>5</b>  |
| <b>PLANO DE ESTUDOS.....</b>                                        | <b>6</b>  |
| <b>1. INTRODUÇÃO A BANCO DE DADOS .....</b>                         | <b>7</b>  |
| <b>2. MODELO ENTIDADE RELACIONAMENTO.....</b>                       | <b>18</b> |
| <b>3. INSTALANDO O VERTRIGO.....</b>                                | <b>32</b> |
| <b>4. CRIANDO UM BANCO DE DADOS NO MySQL.....</b>                   | <b>40</b> |
| <b>5. CRIANDO, ALTERANDO E EXCLUINDO TABELAS NO MySQL.....</b>      | <b>53</b> |
| <b>6. MANIPULAÇÃO DE REGISTROS EM MySQL.....</b>                    | <b>65</b> |
| <b>7. CONSTRUINDO UM PROJETO.....</b>                               | <b>78</b> |
| <b>8. UTILIZANDO O PhpMyAdmin PARA CRIAR UM BANCO DE DADOS.....</b> | <b>89</b> |
| <b>REFERÊNCIAS.....</b>                                             | <b>96</b> |

## Carta do Professor



Caro aluno,

Você já imaginou como, em vários momentos do nosso dia-a-dia utilizamos ou acessamos informações em algum banco de dados? É bem provável que as empresas, atualmente, não conseguiram sobreviver sem armazenar suas informações em um banco de dados.

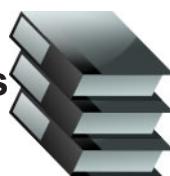
Imagine uma empresa com várias filiais em diversas partes do país ou até mesmo em outros países. É necessário que exista uma ferramenta para auxiliar no armazenamento das informações e no acesso a essas informações. Essa ferramenta já existe, é o banco de dados, que torna possível o acesso às informações sobre funcionários, clientes, produtos e vendas de uma empresa, auxiliando no gerenciamento de todas as informações.

Para a criação de um banco de dados, é importante que alguns conceitos sejam estudados, por isso, durante esta disciplina, conheceremos os conceitos que envolvem a criação de um banco de dados, assim como, conheceremos os comandos utilizados pelo banco de dados MySQL para a criação de um banco de dados e manipulação dos registros nele armazenados.

Iniciaremos compreendendo a importância de utilizarmos um banco de dados. Percebemos a utilização de banco de dados em nosso cotidiano, quando utilizamos um cartão de crédito ou quando consulta nosso saldo na conta bancária, entre tantas outras tarefas.

Professora Francini Reitz Spanceski

## Cronograma de Estudos



Acompanhem no cronograma abaixo os conteúdos das aulas, e atualize as possíveis datas de realização de aprendizagem e avaliações.

| Semanas | Horas/aula | Conteúdos                                            | Data/Avaliação |
|---------|------------|------------------------------------------------------|----------------|
| 1       | 5          | Introdução a Banco de Dados                          | /_ a _/_       |
|         | 10         | Modelo Entidade Relacionamento                       |                |
|         | 5          | Instalando o Vertrigo                                |                |
|         | 10         | Criando um Banco de Dados no MySQL                   |                |
| 2       | 10         | Criando, Alterando e Excluindo Tabelas no MySQL      | /_ a _/_       |
|         | 10         | Manipulação de Registros em MySQL                    |                |
| 3       | 20         | Construindo um Projeto                               | /_ a _/_       |
|         | 10         | Utilizando o PhpMyAdmin para Criar um Banco de Dados |                |

**PLANO DE ESTUDOS****Ementa**

Conceito de Sistemas Gerenciadores de Banco de Dados. Componentes. Modelos de SGBD. Criação de um banco de dados MySQL. Projetando e criando tabelas. Gerenciamento do banco de dados MySQL. Consultando o banco de dados MySQL. Otimização do banco de dados e segurança do banco de dados MySQL.

**Objetivos da Disciplina****• Geral**

Compreender as vantagens da utilização de um banco de dados para o desenvolvimento de aplicações voltadas para a Internet.

**• Específicos**

- Introduzir os conceitos fundamentais dos sistemas de bancos de dados;
- Desenvolver um modelo entidade relacionamento;
- Manipular as informações de um banco de dados, usando os recursos do MySQL;
- Demonstrar as melhores práticas para normalização e modelagem dos dados em um Projeto de Banco de Dados;
- Planejar soluções para aplicações WEB em internet e intranet;
- Ampliar os conhecimentos sobre o funcionamento e os recursos disponíveis pelos Sistemas Gerenciadores de Banco de Dados, usando os recursos do MySQL.

**Carga Horária: 80 horas/aula.**

## Aula 1

# INTRODUÇÃO A BANCO DE DADOS

## Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

1

- Conceituar Banco de Dados;
- Enumerar as vantagens de se utilizar um Banco de Dados;
- Conceituar SGBD e entender seu funcionamento básico.

## Conteúdos da aula



Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Introdução a Banco de Dados;
- Conceitos Básicos de Banco de Dados;
- Vantagens da utilização de um Banco de Dados;
- O que é um SGBD?
- Exercícios propostos.



Olá! Seja bem-vindo(a) a nossa primeira aula. Estudaremos os conceitos básicos de banco de dados, para entendermos os demais conceitos envolvidos no relacionamento entre tabelas e os tipos de relacionamentos. Aplicaremos esse conhecimento no desenvolvimento de um Modelo Entidade Relacionamento, que será utilizado para a criação de banco de dados durante as aulas desta disciplina. Quando falamos em Banco de Dados é importante lembrar que as informações armazenadas, em sua maioria, são extremamente importantes para a organização a que

TUPY Virtual

pertencem, sendo assim, estudaremos uma série de conceitos e técnicas que visam ao gerenciamento de Banco de Dados. Iniciaremos pela apresentação dos princípios dos sistemas de banco de dados.

Boa aula!



## 1 INTRODUÇÃO

Banco de dados é um conjunto de informações relacionadas entre si, sobre um determinado assunto ou entidade, armazenado por meio magnético.

A principal finalidade da criação de um banco de dados é o armazenamento organizado das informações, visando à otimização dos sistemas e facilitando: entrada de dados, alterações, consultas, geração de formulários e relatórios, bem como o uso adequado dessas informações.

Para haver um banco de dados, necessitamos armazenar e trabalhar com dados ou com informações armazenadas nesse banco de dados.

**Ex.** • Empresa de cartão de crédito: precisará armazenar as informações de seus clientes, bem como as informações de compras efetuadas, gerar fatura correspondente aos gastos do cliente. Essas informações devem ser armazenadas e manipuladas garantindo a integridade e a segurança das informações.

• Bancos: informações dos clientes e suas transações bancárias, depósitos, saques, solicitação de serviços, qualquer transação efetuada deve ser registrada de modo que o cliente possa consultar suas informações a qualquer momento.

• Escola: registros das informações dos cursos, alunos, notas e informações pessoais do aluno.

• Empresas: independente da área de atuação, terá necessidade de gerenciar as informações de seus funcionários referentes a salário, descontos/benefícios na folha de pagamento, dependentes e demais informações.

Nos exemplos mencionados, conseguimos perceber que, atualmente, os Bancos de Dados são extremamente importantes para as organizações, no que diz respeito ao armazenamento e manipulação das informações de uma organização.

Segundo Silberschatz, Korth e Sudarshan (2006, p 4), um sistema de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas que permitem aos usuários acessar e modificar os dados.

A principal finalidade da criação de um banco de dados é o armazenamento organizado das informações, visando à otimização dos sistemas e, desse modo, facilitar: entrada de dados, alterações, consultas, geração de formulários e relatórios, visando ao uso adequado dessas informações.

Para existir um banco de dados, precisamos ter a necessidade de armazenar e trabalhar com dados ou com as informações armazenadas no bando de dados.

O software usado para gerenciamento de banco de dados – gerenciamento de um conjunto de dados inter-relacionados – manipula dados de diversas maneiras. Um banco de dados pode ser útil para qualquer pessoa que precise controlar um grande número de fatos relacionados. Assim que os dados são introduzidos no banco, é possível fazer a manipulação desses dados.

O que são dados?

Veremos esse conceito a seguir.



## 2 CONCEITOS BÁSICOS DE BANCO DE DADO

### 2.1 Dado

É uma informação isolada qualquer, sem significado, ou seja, dado é parte de uma informação. Os dados são a matéria-prima – as notas em uma disciplina, os gols marcados ou um nome – a serem processados por um computador.

**Ex.** Gabriel ou Analista de Sistemas. Cada uma dessas informações, quando isoladas, não chega de fato a informar nada, por isso são dados. Quando os dados são agrupados de forma lógica, temos a informação. No exemplo, chegamos ao resultado quando esclarecemos que esses dados são o nome e o cargo.

### 2.2 Informação

É um conjunto lógico de dados com significado. Os dados processados são

transformados em informação – dados que estão organizados de maneira significativa e útil. Na escola, por exemplo, um professor poderia introduzir as notas (dados) de vários alunos, que seriam processadas para produzir as notas finais e, talvez, uma média (informação) da classe. Observe que a informação das notas dadas pelo professor poderia transformar-se em dados para o sistema de registro dos estudantes da escola, os quais produziriam relatórios de notas e coeficientes de rendimento como sua informação. Dados que, por si sós, talvez sejam pouco interessantes, mas de muita importância, quando transformados em informação.

### 2.3 Campo

São os dados agrupados em categorias de informação. Quando iniciarmos a criação de um banco de dados, criaremos tabelas, cujos campos são as colunas da tabela.

Exemplo: nome, endereço, cargo, salário, etc.

### 2.4 Registro

É um conjunto lógico de campos, ou um conjunto lógico de categorias de informação, que não forma um registro, mas um conjunto lógico de campos,..

Para formarmos um registro, é necessário juntarmos, o nome Gabriel com o endereço e com o cargo do Gabriel, pois a lógica desse conjunto é justamente o fato de que todos os campos estão mostrando informações do Gabriel.

São as linhas de uma tabela.

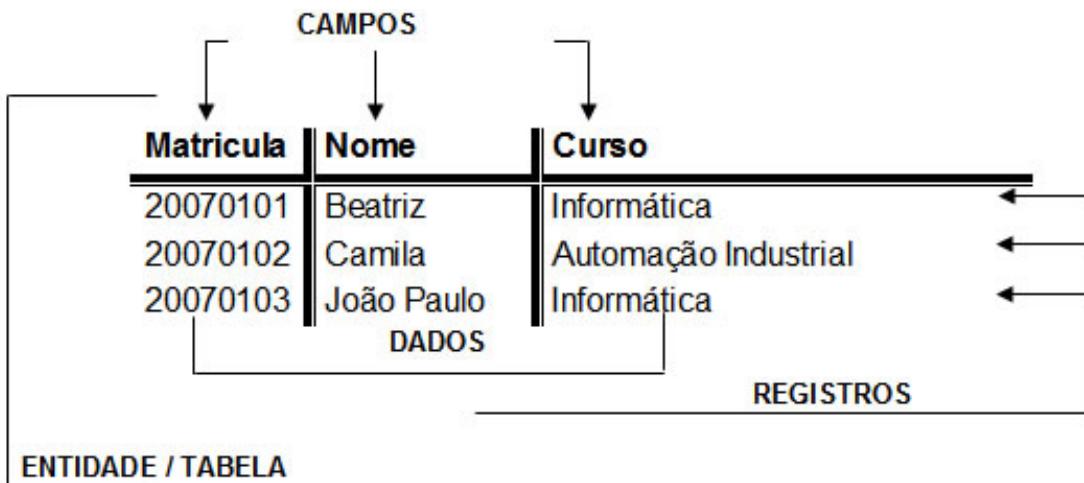
### 2.5 Entidades

As entidades podem também ser conhecidas como Arquivos, cadastro ou tabelas. São os conjuntos lógicos de registros. Se juntarmos um conjunto lógico de registros de clientes com um conjunto de registros de produtos não teremos uma tabela ou entidade, mas uma mistura de informações.

A lógica está no fato de que os registros serão informações de um mesmo

grupo lógico, ou seja, os registros de clientes formam a entidade ou tabela de clientes, os registros de produtos formam a entidade ou tabela de produtos e assim por diante. Veja a figura 1.

Voltando ao conceito de banco de dados, dizemos que é um conjunto de tabelas ou arquivos relacionados entre si.



**Figura 1 – Entidades de Banco de Dados**



### 3 POR QUE USAR UM SISTEMA DE BANCO DE DADOS?

Podemos perceber inúmeras vantagens na utilização de um sistema de banco de dados, em relação aos métodos tradicionais, baseados em papel, em manutenção de registros.

Vamos imaginar uma empresa com as informações de todos os seus funcionários em um arquivo, com várias fichas de papel, uma para cada funcionário, imagine a dificuldade de atualizar as informações de um funcionário, ou de fazer uma pesquisa se, por exemplo, tivéssemos interesse de saber os aniversariantes do dia, ou da semana ou do mês.

Encontraríamos grande dificuldade para fazer qualquer tipo de manipulação nas informações de cada um desses funcionários, porém, a utilização de um sistema de banco de dados proporciona à empresa o controle centralizado de seus dados. Tal situação contrasta com a encontrada em uma empresa que não utiliza um sistema de banco de dados, de modo que os dados ficam dispersos, sendo dificultando o controle de forma sistemática.

Vamos classificar um banco de dados, quanto ao tipo de ambiente, como tradicional (antes da tecnologia de redes se tornar comum) e atual (usando largamente a tecnologia de redes). Num ambiente tradicional, cada usuário ou departamento possuía os seus próprios cadastros.

**EX.** O Departamento de RH precisava ter um cadastro dos funcionários para gerar uma listagem dos aniversariantes, as informações de despesas, etc.; lá se iam 90 registros digitados de funcionários. O departamento de almoxarifado também precisava de um cadastro de funcionários, porque estes solicitavam material de estoque para desenvolver suas atividades. Lá se iam, de novo, os mesmos 90 registros dos funcionários, ocupando duas vezes memória secundária, e por aí afora.

A redundância é alta, mas o que é redundância?

Ocorre redundância quando os mesmos registros sobre as mesmas informações estão cadastrados em diferentes lugares. Se existe redundância, existe inconsistência, que, no ambiente tradicional, também é alta.

O que é inconsistência?

Ocorre quando os mesmos registros estão cadastrados em diferentes lugares, com conteúdos diferentes.

**EX.** A funcionária Francini está cadastrada no RH como Francini da Silva. A mesma funcionária Francini está cadastrada de novo (redundância) no almoxarifado como Francine da Silva. Existe uma inconsistência no conteúdo dos dados um está com I e ou outro está com E.

No ambiente tradicional, como os dados estão em diferentes lugares, cada um faz como quer, ocorrendo alta despadronização e difícil e cara manutenção (quando necessária) dos programas. Cada um chama os seus campos como bem entende.

Num ambiente de banco de dados atual, os dados estão centralizados em um só lugar e são acessados via rede de computadores. Dados em um só lugar, redundância inexistente.

### **ERRADO.**

Por questões de velocidade, performance ou por alguma necessidade específica, alguns dados podem ser armazenados repetidas vezes. Todo administrador de banco de dados precisa ter uma boa política de backup, portanto, existe aí uma condição de redundância.

Como o backup é obrigatório, a redundância existe no ambiente atual, mas é controlada. A inconsistência também existe no ambiente atual, mas é controlada.

Se os dados estão num só lugar, a estrutura dos dados é a mesma, portanto, há forte padronização dos dados, ou seja, chamamos o campo ‘nome de um cadastro de alunos’ de Nm-Aluno. No ambiente tradicional, um poderia chamar de Nm-Aluno, outro de Nome\_Do\_Aluno, outro de NomeDoAluno, e assim vai.

A manutenção de um banco de dados atual custa menos e é mais fácil, devido à padronização. No ambiente atual, os mesmos dados são compartilhados ou não por diversos usuários ao mesmo tempo. Como todos estão compartilhando dados, veja que nem todos podem acessar todas as informações. Ocorre no ambiente atual, o que chamamos de restrições de segurança, ou seja, cada usuário terá definido para si permissões de acesso ou não a um dado cadastrado.

Vamos explicar detalhadamente algumas vantagens da utilização de um banco de dados.

### 3.1 Redução ou eliminação da Redundância

Possibilita a eliminação de dados privativos de cada sistema. Os dados, que, eventualmente, são comuns a mais de um sistema, são compartilhados, permitem acesso a uma única informação e pode ser consultados por vários sistemas.

### 3.2 Eliminação da inconsistência

Com o armazenamento da informação em um único local, com acesso descentralizado e compartilhado à vários sistemas, os usuários utilizarão uma informação confiável. A inconsistência ocorre quando um mesmo campo tem valores diferentes em sistemas diferentes.

**EX.** O estado civil de uma pessoa é solteiro em um sistema e casado em outro. Isso ocorre porque a pessoa atualizou o campo em um sistema e não o atualizou em outro. Quando o dado é armazenado em um único local e compartilhado pelos sistemas, esse problema não ocorre.

### 3.3 Compartilhamento de dados

Permite a utilização simultânea e segura de um dado, por mais de uma aplicação ou usuário, independente da operação que esteja sendo realizada.

### 3.4 Restrição de segurança

Define para cada usuário o nível de acesso que lhe é concedido (leitura, leitura e gravação, sem acesso) ao arquivo e/ou campo. Esse recurso impede que pessoas não autorizadas utilizem ou atualizem um determinado arquivo ou campo.

### 3.5 Padronização dos dados

Permite que os campos armazenados na base de dados sejam padronizados segundo um determinado formato de armazenamento e nome dos campos, seguindo critérios, padrões pré-estabelecidos.

### 3.6 Manutenção da integridade

Impede que um determinado código ou chave de uma tabela tenha correspondência em outra tabela.

**EX.** Um código de uma determinada disciplina, na tabela “Histórico Escolar” , sem a sua descrição na tabela “Disciplina”.

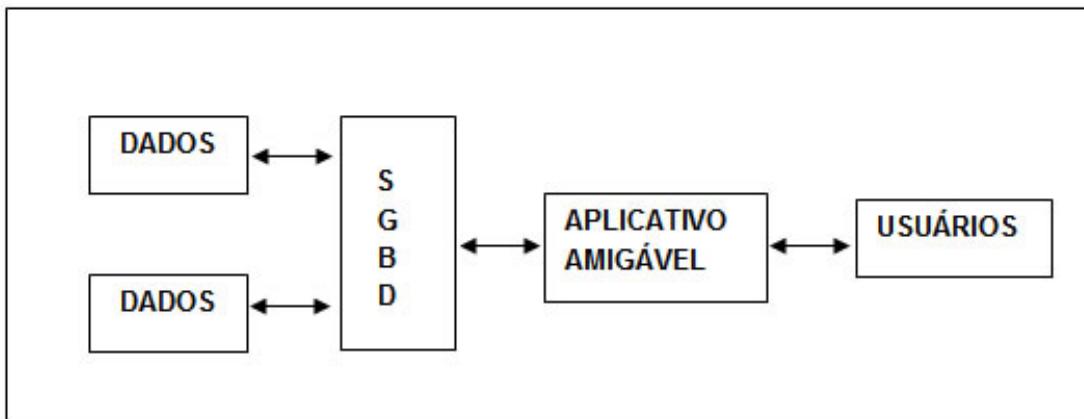


## 4 SISTEMA GERENCIADOR DE BANCO DE DADOS - SGBD

O SGBD – Sistema Gerenciador de Banco de Dados é o software, a ferramenta utilizada para acessar, gerenciar e manipular as informações organizadas e integradas de um sistema de banco de dados.

Na figura 2, podemos perceber o papel do SGBD num sistema de banco de dados, ressaltando que suas funções são:

- **Organização** - a estrutura organizacional dos dados deve refletir as múltiplas visões dos usuários;
- **Acesso** - deve permitir o armazenamento, recuperação e disseminação dos dados;
- **Controle** – garantia de integridade quanto à segurança, recuperação, edição e validação dos dados.



**Figura 2** – Representação do SGBD

Os componentes de um sistema gerenciador de banco de dados são:

- **DDL (Data Definition Language)** - Linguagem de Definição de Dados: conjunto de operações primitivas que permite a declaração de dados, áreas, etc. É o conjunto de comandos que permitem a definição da estrutura dos dados de um banco.
- **DCL (Data Control Language)** - Linguagem de Controle dos Dados: conjunto de comandos que permite a definição dos níveis de permissões de acesso aos dados.
- **DML (Data Management Language)** - Linguagem de Manipulação dos Dados: é um conjunto de comandos que permite a manipulação e/ou manuseio dos dados de um banco.
- **Aplicativo Amigável** - Não chega a ser um componente de um SGBD, mas é a interface que proporciona facilidade e interação no acesso aos dados do banco pelo SGBD.

**Síntese**

Nesta aula você conheceu os conceitos básicos de banco de dados, suas características e sua importância na manipulação e gerenciamento das informações em uma empresa. Verificamos as vantagens na utilização de um banco de dados, por exemplo, a diminuição ou eliminação da redundância, evitando que a informação seja armazenada repetidamente. Assim, a informação será cadastrada apenas uma vez e o acesso será compartilhado via banco de dados, permitindo o acesso atualizado e seguro às informações.

Apresentamos também o conceito de **SGBD – Sistema Gerenciador de Banco de Dados**, cuja principal função é acessar, gerenciar e manipular as informações organizadas e integradas de um sistema de banco de dados.

Na nossa próxima aula estudaremos os conceitos de modelagem de dados, objetivando o desenvolvimento de um Modelo Entidade Relacionamento.

**Exercícios Propostos****1) Assinale a Alternativa correta:**

- a. ( ) Banco de dados é um conjunto de informações que não podem estar relacionadas entre si.
- b. ( ) Banco de é um conjunto de informações relacionadas entre si, de modo que as informações não serão armazenadas.
- c. ( ) Banco de dados é um conjunto de informações relacionadas entre si, sobre um determinado assunto ou entidade que é armazenado por meio magnético.

**2) Coloque V – Verdadeiro ou F – Falso:**

- a. ( ) Dado é uma informação com significado, cada dado representa uma informação para o banco de dados.
- b. ( ) Informações são os dados não processados, para que os dados sejam transformados em informação não é necessário que estejam organizados de maneira ordenada.
- c. ( ) Os Dados são a matéria prima em um banco de dados, são parte de uma informação.
- d. ( ) Campos são os dados agrupados em categorias de informação, são as colunas de uma tabela.
- e. ( ) Registros são um conjunto qualquer de campos, não se faz necessário que estejam organizados de maneira lógica.

**3) Quais as vantagens da utilização de um Banco de Dados atualmente?**

---

---

---

---

**4) Coloque V – Verdadeiro ou F – Falso:**

- a. ( ) Podemos definir redundância como sendo a falta de repetição de registros com as mesmas informações em um banco de dados.
- b. ( ) A inconsistência em um ambiente de banco de dados tradicional é alta.
- c. ( ) A inconsistência ocorre quando os mesmos registros estão cadastrados em diferentes lugares com conteúdos diferentes.
- d. ( ) Redundância acontece quando os mesmos registros sobre as mesmas informações estão cadastrados em diferentes lugares.
- e. ( ) Nos bancos de dados atuais podemos dizer que a inconsistência foi eliminada e a redundância foi controlada ou em alguns casos eliminada.

## Aula 2

## MODELO ENTIDADE RELACIONAMENTO

### Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Desenvolver de um modelo de Banco de Dados;
- Identificar os principais conceitos que envolvem o Modelo Entidade Relacionamento;
- Identificar os relacionamentos utilizados na criação de um Modelo Entidade Relacionamento.

### Conteúdos da aula



Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Modelo Entidade Relacionamento;
- Definindo Entidades/Tabelas;
- Cardinalidade;
- Relacionamento entre Tabelas;
- Tipos de Relacionamentos;
- Exercícios propostos.



Seja bem-vindo(a) a nossa segunda aula!

Você já conheceu os conceitos de Banco de Dados, assim como a importância de sua utilização no gerenciamento das informações de uma instituição. Nesta aula estudaremos sobre a utilização da Modelagem de Dados no desenvolvimento de um Banco de Dados, apresentando o que é um Modelo Entidade

2

Tupu Virtual

Relacionamento e aplicando os conceitos para a criação do modelo. Desenvolveremos, com o auxílio dos conceitos, o relacionamento entre as entidades/tabelas do banco de dados e a representação dessas informações.

Boa aula!



## 1 MODELOS DE BANCO DE DADOS

Segundo Heuser (2002, p. 5), um modelo de (banco de) dados é uma descrição dos tipos de informações que estão armazenadas em um banco de dados.

A modelagem de dados é uma técnica que visa facilitar a comunicação entre o usuário e o analista de sistemas e que modela os dados de uma empresa, usando técnicas afins (Modelo Entidade Relacionamento), procurando refletir as atividades dessa empresa.

Imaginemos um banco de dados que armazene informações sobre os Alunos. O modelo de dados poderia informar que o banco de dados armazena informações sobre alunos e, para cada aluno, são armazenados matrícula, nome, endereço, telefone, data de nascimento, entre outras informações.

É necessário observar que o modelo de dados não informa quais alunos serão armazenados no banco de dados, mas apenas que o banco de dados contém informações sobre Alunos.

Para construir um modelo de dados, usa-se uma linguagem de modelagem de dados. Um mesmo modelo pode ser apresentado de várias formas e utilizando vários níveis de abstração.

**Ex.** Um modelo de dados utilizado para explicar a um usuário leigo sobre um determinado banco de dados, não deve conter detalhes sobre a representação em meio físico das informações, já que, provavelmente, essas informações não seriam compreendidas por esse usuário.

Vamos imaginar, porém, um modelo de dados utilizado por um técnico para otimizar a performance de acesso ao banco de dados. Certamente haverá mais detalhes de como as informações estão organizadas internamente, de modo a auxiliar na tarefa a ser desenvolvida.

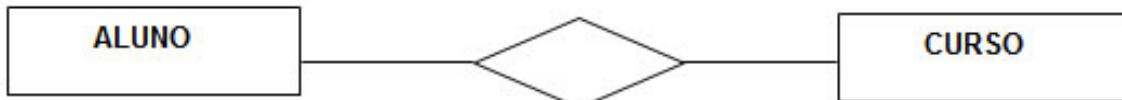
Quando falamos em modelos de banco de dados normalmente consideramos

dois níveis de abstração: o modelo conceitual e o modelo lógico.

### 1.1 Modelo Conceitual

Um modelo conceitual é uma descrição do banco de dados, independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como esses dados estão armazenados no SGBD.

Para a representação do modelo conceitual, visualizado na figura 3, a técnica mais difundida é o Modelo Entidade Relacionamento.



**Figura 3 – Exemplo de Modelo Conceitual**

### 2 MODELO LÓGICO



Um modelo lógico é a descrição do banco de dados, que deverá definir ou apresentar quais as tabelas contidas no banco e, para cada tabela deve ser apresentado o nome das colunas – os campos.

O modelo lógico representa a estrutura do Banco de Dados, conforme o usuário do SGBD o vê. Também define como o banco de dados será implementado.

A tabela 1 mostra a junção das tabelas curso e aluno.

**Tabela: Curso**

| SiglaCurso | DescCurso            |
|------------|----------------------|
| AT         | Automação Industrial |
| IN         | Informática          |
| MA         | Materiais            |
| TL         | Telecomunicações     |

**Tabela: Aluno**

| Matrícula | Nome          | Telefone      | SiglaCurso |
|-----------|---------------|---------------|------------|
| 20070101  | Beatriz       | (47)3422-0000 | IN         |
| 20070102  | Camila        | (47)3461-0102 | AT         |
| 20070103  | João Paluo    | (47)3465-0501 | IN         |
| 20070104  | Luis Fernando | (47)3427-0104 | TL         |

O modelo lógico das tabelas relacionadas acima é:

**Curso**(SiglaCurso, DescCurso)

**Aluno**(Matricula, Nome, Telefone, SiglaCurso)

Quando desenvolvemos um projeto de Banco de Dados, primeiramente desenvolvemos o modelo conceitual, na forma de diagrama entidade relacionamento. A partir desse modelo, desenvolvemos o projeto lógico, objetivando transformar o modelo conceitual desenvolvido em uma definição de como o banco de dados será implementado.

Esse processo é adequado para o desenvolvimento de um novo projeto de Banco de Dados. Caso já exista um banco de dados e se pretenda construir um novo, o processo acima precisará incorporar a etapa de engenharia reversa de banco de dados. No momento não trataremos dessa etapa. A seguir apresentaremos os conceitos para o desenvolvimento de um modelo Entidade Relacionamento.

### 3 ABORDAGEM OU MODELO ENTIDADE RELACIONAMENTO

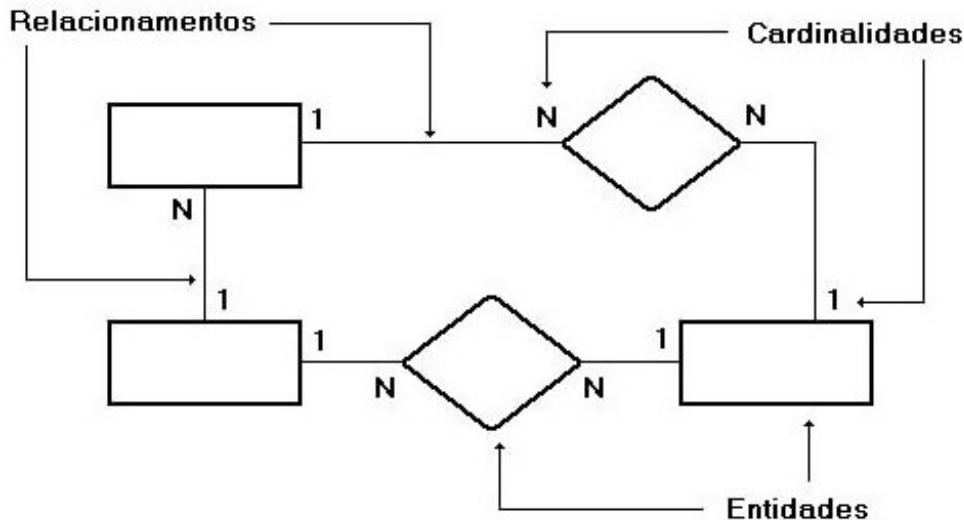


O modelo entidade relacionamento usa um conjunto de tabelas para representar tanto os dados quanto as relações (relacionamentos) entre eles. Esse conceito permite ampla utilização do modelo, fazendo com que a grande maioria dos produtos de banco de dados tenha como base o modelo relacional.

Segundo Silberschatz, Korth e Sudarshan (2006, p. 25), o modelo relacional é hoje o principal modelo de dados para aplicações comerciais de processamento de dados.

O modelo entidade relacionamento foi desenvolvido em 1976, por Peter Chen, podendo ser considerado um padrão para a modelagem conceitual.

Na figura, podemos visualizar os principais conceitos que envolvem o modelo Entidade Relacionamento, estes conceitos serão detalhados nos capítulos a seguir, objetivando o desenvolvimento da abordagem entidade relacionamento.



**Figura 04 – Modelo Entidade Relacionamento**

Seguem alguns conceitos que envolvem esse modelo, assim como uma notação gráfica para os diagramas ER.

### 3.1 Entidade

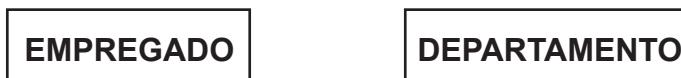
Segundo Heuser (2001, p.12) o conceito fundamental da abordagem ER é o conceito de entidade.

Uma entidade pode ser conhecida também como tabela, representando um conjunto de atributos da realidade modelada, sobre os quais se deseja manter informações no banco de dados, composta de campos e registros. Uma entidade sempre representa um assunto específico, que pode ser tanto um objeto como um evento.

Quando uma tabela representa um objeto, está representando alguma coisa tangível, como uma pessoa, lugar ou coisa. Este objeto possui várias características que podem ser os campos da tabela. Exemplos de objetos: produtos, clientes, alunos, pacientes, materiais, empregados, departamento, entre outros.

Quando uma tabela representa um evento, representará algo que ocorre em determinado espaço de tempo. Assim, os fatos de um evento podem ser os campos de uma tabela. Exemplos de eventos: entrevistas, vendas, visitas, transferências, entre outros.

Uma entidade ou tabela é representada em um modelo ER, por um retângulo que contém o nome da entidade. Veja a figura 5.



**Figura 05 – Exemplo de Entidades**

Na figura acima, o primeiro retângulo representa o conjunto de todos os empregados sobre os quais se deseja manter informações no banco de dados, enquanto o segundo representa o conjunto de todos os departamentos sobre os quais se deseja manter informações.

### 3.2 Atributo

O conceito de atributo serve para associar informações a ocorrências de entidades ou de relacionamentos. Um atributo é o dado associado a cada ocorrência de uma entidade ou de um relacionamento. No desenvolvimento de um modelo entidade relacionamento, utilizaremos o termo atributo quando estivermos desenvolvendo o banco de dados. Os atributos serão os campos de uma tabela no banco de dados.

Na prática, os atributos não são representados graficamente, para não sobre-carregar os diagramas, já que algumas vezes as entidades possuem um grande número de atributos. Utiliza-se uma representação textual que aparece separadamente do modelo ER.

### 3.3 Chaves

As chaves são fundamentais no desenvolvimento de um banco de dados. Garantem que cada registro de uma tabela possa ser corretamente identificado. São usadas para que seja possível desenvolver o relacionamento entre as entidades, auxiliando a impor a integridade no desenvolvimento de um banco de dados.

É necessário ter um a maneira de especificar como as entidades, dentro de um determinado conjunto de entidades, se distinguem, ou seja, é preciso que cada registro, dentro de um banco de dados seja identificado como único.

Vamos imaginar uma entidade aluno, onde são armazenadas as informações: nome, endereço, telefone, curso, entre outras. É preciso que cada aluno cadastrado

no banco de dados possa ser identificado como um registro único, para isso, será necessário armazenar a matrícula do aluno, podendo ser um número, porém não poderá existir no banco de dados dois alunos com a mesma matrícula. O atributo matrícula será a chave primária dentro da entidade aluno.

Desse modo, a chave primária será um atributo ou campo que irá identificar cada registro, unicamente. Outro tipo de chave é a chave estrangeira, cujo atributo ou campo é utilizado para estabelecer o relacionamento entre duas entidades.

### 3.3.1 Chave Primária

A chave primária identifica a tabela em toda a estrutura do banco de dados. Os dados armazenados no campo **chave primária** devem ser **únicos** para cada registro do banco de dados. Essa chave é usada para impor integridade à tabela, para ajudar a estabelecer relacionamentos com outras tabelas e para identificar precisamente e fazer referência a um registro específico dentro da tabela.

### 3.3.2 Chave Estrangeira

Utilizaremos uma chave estrangeira para estabelecer o relacionamento entre duas entidades ou tabelas. (O conceito de relacionamento será apresentado a seguir). Sempre que houver relacionamento do registro de uma entidade qualquer para muitos (N registros) de outra entidade qualquer, o campo que estiver definido como chave primária no lado um do relacionamento, será armazenado como chave estrangeira no lado N desse relacionamento.

Imaginemos um relacionamento entre uma entidade Produto e Tipo de Produto. Para cada Tipo de Produto pode haver vários (muitos ou N) registros de Produtos, porém, para cada produto haverá apenas um Tipo de produto, desse modo, o campo que for definido como chave primária, na entidade Tipo Produto, será definido como Chave Estrangeira na entidade Produto.

Trabalharemos com esse conceito quando definirmos os relacionamentos entre as entidades para o desenvolvimento do modelo Entidade Relacionamento.

### 3.4 Cardinalidade

A cardinalidade de um relacionamento, segundo Heuser (2001, p. 16), pode ser definida pelas ocorrências de uma entidade. Podem estar associadas a uma determinada ocorrência através de um relacionamento.

As cardinalidades dizem respeito às ocorrências do registro de uma entidade associada a outra. Podemos ter cardinalidade mínima – ocorrência do registro de uma entidade associado ao registro de outra entidade – ou cardinalidade máxima – ocorrência de muitos (N) registros de uma entidade associados a uma outra entidade.

### 3.5 Relacionamento

Segundo Hernandez (2000, p.44), a conexão estabelecida entre duas tabelas é conhecida como relacionamento.

Um relacionamento existe quando as tabelas ou entidades são conectadas por uma chave primária e uma chave estrangeira. A maneira pela qual as tabelas são conectadas depende do tipo de relacionamento que ocorre entre elas.

Um relacionamento é parte fundamental em um banco de dados, porém, para aproveitar as vantagens de um banco de dados relacional, é muito importante que os relacionamentos entre as entidades sejam estabelecidos de maneira correta. O erro no relacionamento entre as entidades pode dificultar o desenvolvimento e a utilização do banco de dados, de modo a dificultar a inserção, consulta, atualização e exclusão dos registros nas tabelas relacionadas.

É importante conhecermos quais os tipos de relacionamentos que podemos estabelecer entre as tabelas. A inter-relação de tabelas gerará um tipo específico de relacionamento. Saber reconhecer corretamente esses tipos de relacionamentos é fundamental para o desenvolvimento de um banco de dados. Veremos três tipos de relacionamentos possíveis entre as tabelas: um-para-um, um-para-muitos e muitos-para-muitos.

### 3.5.1 Relacionamento Um-para-Um (1-1)

Cada ocorrência de entidade corresponde a **uma** e somente **uma** ocorrência na outra entidade.

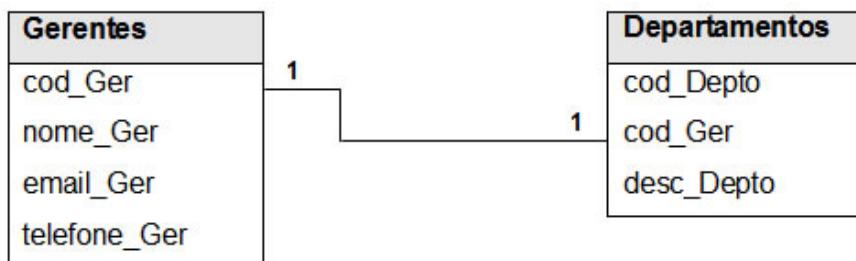
Desse modo, reconhecemos um relacionamento desse tipo quando um registro na primeira tabela estiver relacionado a **um** e somente **um registro** na segunda tabela e **um registro** na segunda tabela estiver relacionado a **um** e somente **um** na primeira tabela.



No exemplo acima podemos verificar um relacionamento um-para-um entre as entidades Gerente e Departamento, podemos concluir que UM Gerente gerencia UM e apenas UM Departamento, do mesmo modo que UM Departamento é Gerenciado por UM e apenas UM Gerente, reconhecemos então um relacionamento um-para-um entre estas duas entidades.

Para estabelecer um relacionamento desse tipo, deve-se pegar uma cópia da chave primária da tabela principal e inseri-la na tabela subordinada, para que se torne uma chave estrangeira, pois a tabela subordinada possui sua própria chave primária.

No relacionamento entre Gerente e Departamento, a tabela Gerente é a tabela principal, sendo assim, a tabela Departamento deverá receber um novo campo ou atributo que será uma cópia da chave primária da tabela Gerente. Verifique a figura 6.



**Figura 06** – Relacionamento um-para-um

O campo `cod_Ger`, chave primária na tabela Gerentes, é inserido na tabela Departamentos como chave estrangeira.

### 3.5.2 Relacionamento Um-para-Muitos (1 – N)

A cada ocorrência de uma entidade corresponde uma ou mais ocorrências na outra entidade.

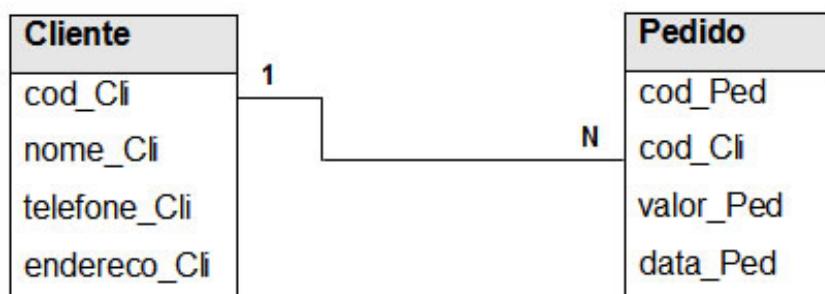
Sendo assim, um relacionamento desse tipo ocorre quando um registro na primeira tabela se relaciona a um ou mais registros na segunda tabela, mas um registro na segunda tabela pode estar relacionado a apenas um registro na primeira tabela.



No exemplo, verificamos um relacionamento um-para-muitos entre as entidades Cliente e Pedido. Podemos concluir que UM Cliente faz UM ou MUITOS Pedidos, porém UM Pedido é feito por UM e apenas UM Cliente, reconhecemos então um relacionamento um-para-muitos entre essas duas entidades.

Para desenvolver um relacionamento desse tipo, é necessário fazer uma cópia do campo ou atributo que é chave primária no lado UM do relacionamento e inseri-lo na tabela do lado N (muitos), onde esse campo será chave estrangeira.

No relacionamento entre Cliente e Pedido, a tabela do lado UM é a tabela Cliente, desse modo, o campo chave primária nesta tabela será inserido na tabela Pedido (lado N) como chave estrangeira, verifique a figura 7.



**Figura 07** – Relacionamento um-para-muitos

O campo cod\_Cli, chave primária na tabela Clientes (lado 1 do relacionamento), é inserido na tabela Pedido (lado N do relacionamento) como chave estrangeira.

### 3.5.3 Relacionamento Muitos-para-Muitos (N – N)

A cada ocorrência de uma entidade corresponde uma ou mais ocorrências na outra entidade e cada ocorrência dessa outra pode ocorrer uma ou mais ocorrências

na outra entidade e cada ocorrência dessa outra pode ocorrer uma ou mais ocorrências na primeira.

Um relacionamento desse tipo ocorre quando um registro na primeira tabela pode ser relacionado a um ou mais registros na segunda tabela e um registro na segunda tabela pode ser relacionado a um ou mais registros na primeira tabela.



No exemplo acima cada Paciente pode consultar com UM ou MUITOS (N) médicos e cada Médico pode consultar UM ou MUITOS Pacientes, desse modo, temos um relacionamento muitos-para-muitos (N – N).

Quando existir um relacionamento desse tipo entre duas entidades, é necessário utilizar uma terceira entidade, denominada entidade associativa, para que se torne possível a existência do relacionamento. Neste caso é necessário:

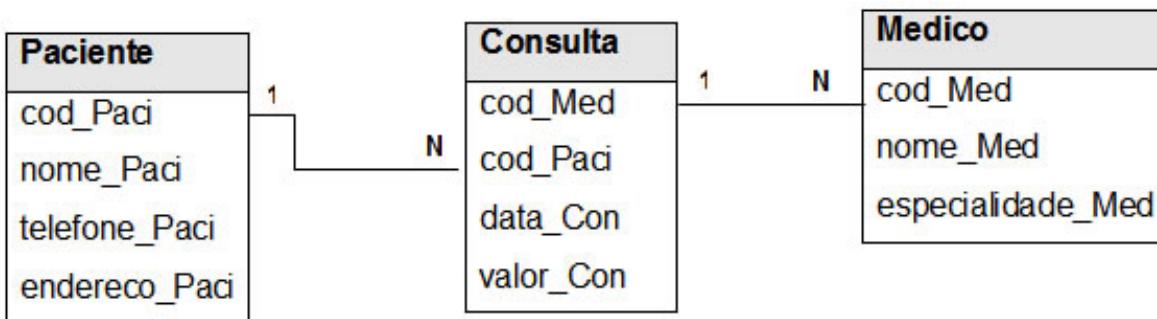
- Eliminar o relacionamento N-N;
- Criar uma entidade associativa (losango) entre as entidades para substituir o relacionamento N-N;
- Criar relacionamentos 1-N entre as entidades básicas (retângulo) e a entidade associativa (losango). As entidades básicas sempre serão o lado 1 e a entidade associativa será o lado N. Vamos verificar como ficaria nosso exemplo, veja a figura 8.



**Figura 08** – Criação da tabela associativa Consulta

Podemos observar que o relacionamento N – N foi quebrado, desse modo, para estabelecer o vínculo entre esse relacionamento, é necessário que uma cópia das chaves primárias das entidades básicas (Paciente e Médico) sejam inseridas na tabela associativa como chave estrangeira. A chave primária da tabela associativa pode ser formada pela junção das duas chaves estrangeiras ou pela criação de um novo campo como chave primária da tabela associativa. Quando houver necessidade, a tabela associativa pode receber outros campos que possam trazer informações

trazer informações importantes para esse relacionamento. Veja a figura 9.



**Figura 09** – Relacionamento muitos-para-muitos



Nesta aula você estudou sobre o desenvolvimento de um Modelo Entidade Relacionamento. Você definiu o modelo de banco de dados utilizando os conceitos de modelagem de dados, estabelecendo quais Entidades/Tabelas farão parte desse modelo, assim como os relacionamentos entre as entidades.

Verificamos os tipos de relacionamentos que podem existir entre duas tabelas e como será estabelecido esse relacionamento, com o uso do campo chave primária em uma das tabelas e campo chave estrangeira na outra tabela.

Vimos também o conceito de cardinalidade, que permite observarmos quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência da outra entidade.

Na próxima aula instalaremos o Vertrigo para que, em seguida, possamos iniciar o desenvolvimento de banco de dados utilizando o MySQL.

**Exercícios Propostos**

Após uma excelente leitura desta aula, você estará apto a responder as questões seguintes.

**1) Assinale a Alternativa correta:**

- a. ( ) A modelagem de dados é uma técnica que dificultou a comunicação entre o usuário e o analista de sistemas.
- b. ( ) Para construir um modelo de dados, usa-se uma linguagem de modelagem de dados. Um mesmo modelo pode ser apresentado de várias formas e utilizando vários níveis de abstração.
- c. ( ) Quando falamos em modelo de dados podemos utilizar apenas um modelo que é chamado de modelo lógico.

**2) Coloque V – Verdadeiro ou F – Falso:**

- a. ( ) O modelo entidade relacionamento foi desenvolvido em 1976 por Peter Chen, podendo ser considerado um padrão para a modelagem conceitual.
- b. ( ) As entidades são representadas em um modelo entidade relacionamento pela figura de um retângulo, porém o conceito de entidade não é de grande importância para o desenvolvimento de um modelo ER.
- c. ( ) A chave primária será um atributo ou campo que irá identificar unicamente cada registro. Outro tipo de chave é a chave estrangeira que é um atributo ou campo utilizado para estabelecer o relacionamento entre duas entidades.
- d. ( ) A cardinalidade em um modelo ER diz respeito ao número de entidades existentes no modelo a ser desenvolvido.
- e. ( ) Um relacionamento existe quando as tabelas ou entidades são conectadas por uma chave primária e uma chave estrangeira.

**3) Quais os três tipos de relacionamentos possíveis entre duas entidades?**

---

---

**4) Coloque V – Verdadeiro ou F – Falso:**

- a. ( ) O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados no SGBD.
- b. ( ) Para a representação do modelo Lógico a técnica mais difundida é o Modelo Entidade Relacionamento
- c. ( ) Um modelo lógico é descrição do banco de dados, que deverá definir ou apresentar quais as tabelas que o banco contém, e para cada tabela não é necessário apresentar os campos.
- d. ( ) Quando desenvolvemos um projeto de Banco de Dados primeiramente é desenvolvido o modelo conceitual, na forma de diagrama entidade relacionamento.
- e. ( ) O projeto lógico é desenvolvido baseando-se no desenvolvimento de outros projetos, não sendo necessário que seja desenvolvido a partir do modelo conceitual do banco de dados a ser desenvolvido.

**5) Faça o modelo entidade relacionamento e identifique os relacionamentos existentes entre as seguintes entidades:**

- a. Correntista - Conta Corrente
- b. Departamento - Empregado
- c. Aluno - Curso
- d. Funcionário - Dependente
- e. Turma - Aluno

## Aula 3

## INSTALANDO O VERTIGO

**Objetivos da aula**

Ao final desta aula, você deverá ser capaz de:

- Instalar adequadamente Vertrigo;
- Identificar os componentes do Vertrigo.

**Conteúdos da aula**

Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- O que é o Vertrigo;
- Instalando o Vertrigo;
- Exercícios Propostos.



Olá! Seja bem-vindo(a) a nossa terceira aula. Na aula anterior conhecemos os conceitos que envolvem a modelagem de um banco de dados. antes, porém, de iniciarmos a criação de um banco de dados na prática, precisamos instalar a ferramenta que utilizaremos. Assim, faremos a instalação do VertrigoServ. Acompanhe os passos apresentados nesta aula e vamos instalar o MySQL para que possamos criar nosso banco de dados em MySQL nas próximas aulas.

Boa Aula!!

3

Tupu Virtual

## 1 O QUE É O VERTRIGO



VertrigoServ foi desenvolvido para disponibilizar um pacote altamente profissional e de fácil instalação do Apache (servidor web HTTP), PHP (linguagem de programação), MySQL (Sistema Gerenciador de Base de dados SQL), SQLite (sistema gerenciador de base de dados relacional ACID-compliant), SQLiteManager (ferramenta web multi-linguagem para gerenciar base de dados SQLite), PhpMyAdmin (ferramenta escrita em PHP que objetiva manusear a administração do MySQL) e Zend Optimizer (para aumentar o desempenho do runtime até 40%) para plataforma Windows.

Possui um instalador, tornando o processo de instalação dos componentes citados muito simples, todos os componentes são instalados em um único diretório e podem ser usados imediatamente ao término do processo de instalação. Um desinstalador permite remover o VertrigoServ do disco rígido.

## 2 INSTALANDO O VERTIGO



Para instalar o Vertrigo, você pode fazer o download no site <http://vertrigo.sourceforge.net/?lang=br> em seguida, vamos detalhar o processo de instalação.

Após o download, dê um duplo clique no arquivo para iniciar a instalação. É necessário escolher o idioma para continuar a instalação. Selecione o idioma **Português (do Brasil)** e clique no botão **OK**, conforme mostra a figura 10.



**Figura 10** – Selecionando o idioma para instalação do Vertrigo

A tela de boas-vindas será apresentada ao assistente, que auxiliará durante

todo o processo de instalação. Nessa tela, você deve apenas verificar as informações apresentadas e clicar no botão **Avançar**, conforme mostra a figura 11.

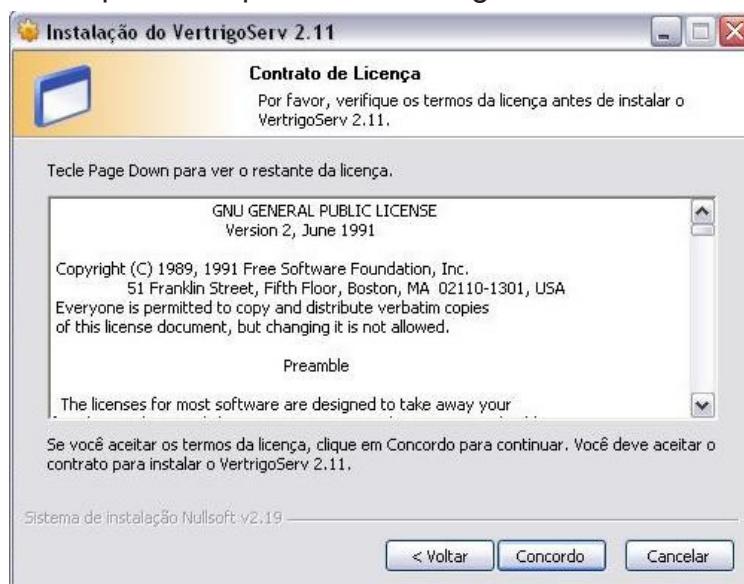


**Figura 11 – Tela de Boas Vindas ao assistente de instalação do Vertrigo**

A próxima tela apresentará o contrato de licença para que possa ser feita a leitura das informações.

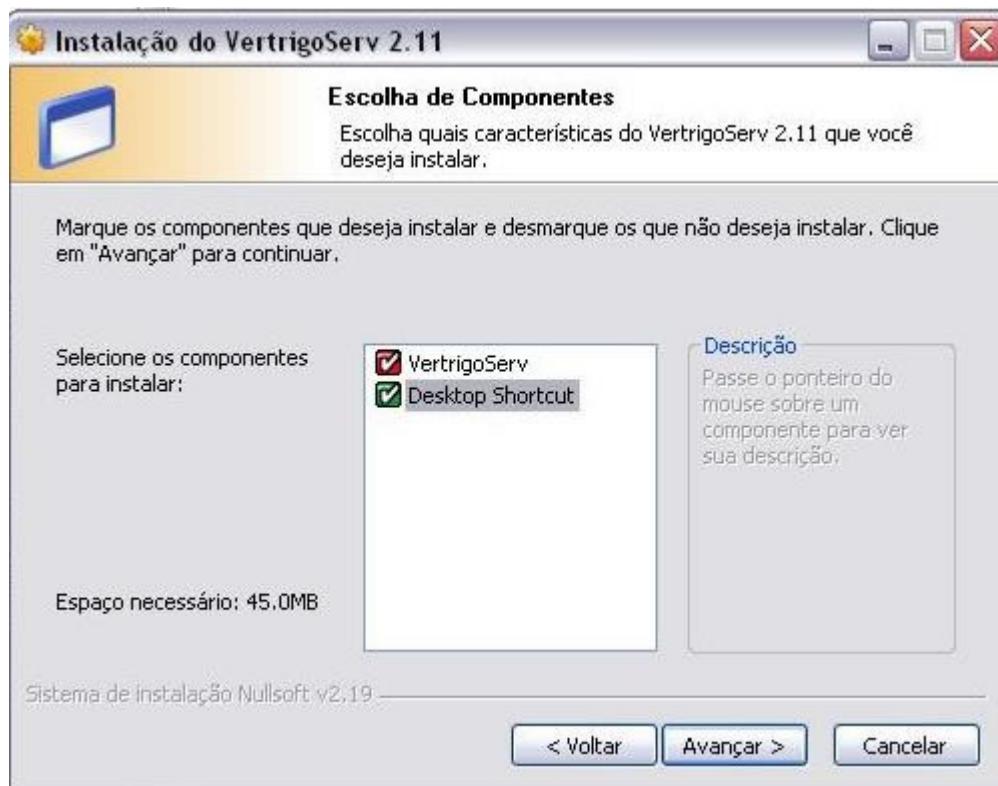
Clique no botão **Concordo** para continuar a instalação.

Verifique a tela que será apresentada na figura 12.



**Figura 12 – Tela do Contrato de Licença**

A próxima etapa é para escolhermos os componentes que serão instalados. Com a seleção na opção **VertrigoServ**, todos os itens que foram detalhados no início desse capítulo serão instalados, com a seleção na opção **Desktop Shortcut** será criado um ícone do VertrigoServ, na área de trabalho. Verifique que nessa tela também é apresentado o espaço necessário em disco para a instalação dos itens selecionados. Selecione os componentes para instalação e clique no botão **Avançar**, conforme mostra a figura 13.



**Figura 13 – Escolha dos Componentes para instalação**

Em seguida, devemos escolher o local onde os arquivos do VertrigoServ serão gravados em seu computador. Será mostrada a sugestão de **Pasta de Destino**, para alterar o local de instalação. Clique no botão **Procurar** e selecione a pasta onde deseja que as informações sejam salvas. Nessa tela também são mostradas outras duas informações importantes, o **espaço necessário** para a instalação e o **espaço disponível** no computador onde o VertrigoServ será instalado. Verifique essas informações para garantir que existe espaço disponível suficiente para que a instalação possa acontecer. Após definir a pasta de destino e verificar as informações de espaço para a instalação, clique no botão **Avançar**, conforme mostra a figura 14.



**Figura 14 – Escolha do Local de Instalação**

Agora é necessário definir uma pasta do Menu Iniciar para **criar um atalho** para o VertrigoServ. Você poderá selecionar uma pasta existente ou digitar o nome da pasta que deseja criar. Se a opção **Não Criar Atalhos** for selecionada não será criado um atalho do VertrigoServer no Menu Iniciar. Defina o nome da pasta a ser criada e, em seguida, clique no botão **Instalar**. Verifique esses itens na figura 15.



**Figura 15 – Definir pasta do Menu Iniciar para os atalhos do VertrigoServ**

Em seguida, será apresentada uma tela de acompanhamento da instalação, que fechará sozinha quando a instalação for concluída, verifique a figura 16.



**Figura 16** – Acompanhamento da instalação

A instalação do VertrigoServ foi concluída. Nesta tela você pode selecionar uma opção para **executar o VertrigoServ**, além de poder selecionar outra opção para **Mostrar Leiame**, arquivo com algumas informações bastante importantes em relação ao VertrigoServ, principalmente em relação às senhas *defaults* que são definidas durante a instalação. Verifique essas informações, selecionando a opção para executar Leiame. Selecione as opções e clique no botão **Terminar**. Verifique as informações na figura 17.



**Figura 17** – Finalizando a instalação

Ainda será apresentada uma tela antes de iniciar o VertrigoServ. Clique no botão **Hide this Window and start server**, para que a janela mostrada seja fechada e o VertrigoServer possa ser iniciado. Verifique o procedimento na figura 18.



**Figura 18** – Inicializando o VertrigoServer

Neste momento o VertrigoServer foi iniciado, podemos então utilizar o Banco de Dados MySQL que foi instalado para criar nossos banco de dados na prática.

**Síntese**

Nesta aula instalamos o VertrigoServ, que servirá para a instalação do banco de dados MySQL, além de outras ferramentas utilizadas no desenvolvimento de aplicações Web como, Apache, que é o servidor Web HTTP, a linguagem de programação, entre outras ferramentas comentadas anteriormente.

Para cada uma das etapas do processo de instalação do VertrigoServ, detalhamos os passos que devemos seguir de modo a permitir a criação de nosso Banco de Dados.

Na próxima aula utilizaremos o banco de dados MySQL.

## Aula 4

# CRIANDO UM BANCO DE DADOS NO MySQL

## Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Analisar as características do Banco de Dados MySQL;
- Identificar os comandos para criação e exclusão de um Banco de Dados;
- Utilizar as principais instruções do MySQL.

## Conteúdos da aula



Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Banco de Dados MySQL;
- Criando um Banco de Dados;
- Excluindo um Banco de Dados;
- Criando Tabelas;
- Excluindo Tabelas;
- Exercícios Propostos.



Olá! Seja bem- vindo(a) a nossa quarta aula. Você já tomou conhecimento dos conceitos de Banco de Dados e Modelagem de Dados. Agora já podemos iniciar o desenvolvimento de um Banco de Dados na prática. Primeiramente, nos familiarizaremos com o MySQL e, a partir daí, criaremos tabelas para este novo banco de dados. Acompanhe os conceitos apresentados nesta aula e vamos criar nosso banco de dados em MySQL.

Boa Aula!!

4

TUPLA Virtual



## 1 HISTÓRICO DO MYSQL

O MySQL foi criado na Suécia por dois suecos: David Axmark, Allan Larsson, e um finlandês: Michael “Monty” Widenius, que trabalham juntos desde a década de 1980.

O MySQL se tornou o mais popular banco de dados *open source* (código aberto) do mundo, porque possui consistência, alta performance, confiabilidade e é fácil de usar. É um sério competidor para os maiores sistemas de banco de dados existentes para aplicações de pequeno e médio porte.

O MySQL começou como uma ferramenta para atender a uma necessidade interna. Quando surgiu, era apenas um substituto para o ultrapassado sistema de banco de dados mSQL. Ao relacionarem tabelas usando rotinas ISAM – por sinal muito rápidas - no mSQL, os autores conseguiram uma versão atualizada a que chamaram **MySQL**.



ISAM é a abreviatura de *Indexed Sequential Access Method* (método de acesso seqüencial indexado) cujos dados, segundo Buyens (2002, p.36), são armazenados seqüencialmente em um arquivo de disco.

Desse modo, a recuperação dos dados é feita por meio de um índice que especifica o deslocamento no arquivo de dados, tornando as rotinas de leitura dos dados mais rápidas.

Atualmente, o MySQL é usado em mais de 6 milhões de instalações, em todos os continentes. Além disso, o MySQL se tornou a escolha de uma nova geração de aplicações, que utiliza o modelo LAMP (Linux, Apache, MySQL, PHP). Funciona em mais de 20 plataformas, incluindo Linux, Windows, HP-UX, AIX, Netware, permitindo que exista flexibilidade e controle.

O MySQL é desenvolvido, distribuído e tem suporte da MySQL AB, empresa comercial, fundada pelos desenvolvedores do MySQL, cujos negócios é fornecer serviços relacionados ao sistema de gerenciamento de banco de dados MySQL. O site do MySQL é <http://www.mysql.com> onde podem ser encontradas informações atualizadas sobre o MySQL e a empresa desenvolvedora – a MySQL AB.



A origem do nome MySQL é um mistério até mesmo para os autores. Segundo eles, os diretórios-base e um grande número de suas bibliotecas e ferramentas sempre tiveram o prefixo “my”, pelo menos por 10 anos. A filha de Monty também ganhou o nome My. Qual das duas originou o nome do MySQL continua sendo uma incógnita.

O golfinho foi escolhido como um símbolo para o banco de dados MySQL já que ele é esperto, rápido e um animal ágil, se esforçando em navegar pelos oceanos de dados. O nome do golfinho do MySQL (logo do MySQL) é Sakila, escolhido pelos fundadores da MySQL AB, a partir de uma enorme lista de nomes sugeridos pelos usuários em um concurso “Name the Dolphin”. De acordo com o vencedor, o nome Sakila tem as suas raízes em SiSwati, a língua local de Swaziland. Sakila é também o nome de uma cidade em Arusha, Tanzânia, próxima ao país de origem do vencedor do concurso, Uganda.



## 2 O QUE É O MYSQL?

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language - Linguagem de Consulta Estruturada) como interface. A SQL é um padrão de comunicação com banco de dados de qualquer tipo.

Vamos verificar algumas características do MySQL, segundo o WIKIPÉDIA:

- Portabilidade (suporta praticamente qualquer plataforma atual)
- Compatibilidade (existem drivers ODBC, JDBC e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, Python, Perl, PHP e Ruby)
- Excelente desempenho e estabilidade;
- Pouco exigente quanto a recursos de hardware;
- Facilidade de uso;
- É um Software Livre;
- Suporte a vários tipos de tabelas (como MyISAM e InnoDB), cada um específico para um fim;

- Faltam alguns recursos quando comparados a outros banco de dados, como o PostgreSQL.

De acordo com as características apresentadas, podemos perceber algumas vantagens na utilização desse banco de dados, principalmente no que diz respeito à compatibilidade com interfaces para várias linguagens, portabilidade de utilizar qualquer plataforma para desenvolvimento. É importante também percebermos que é um banco de dados que exige poucos recursos de hardware e tem grande facilidade de uso, tendo como base a linguagem SQL.

### 3 CRIANDO UM BANCO DE DADOS



Na aula anterior, instalamos o servidor Apache, o PHP e o MySQL. Agora iniciaremos a criação de um banco de dados utilizando o MySQL, que como qualquer outro SGBD, permite a criação de vários bancos.

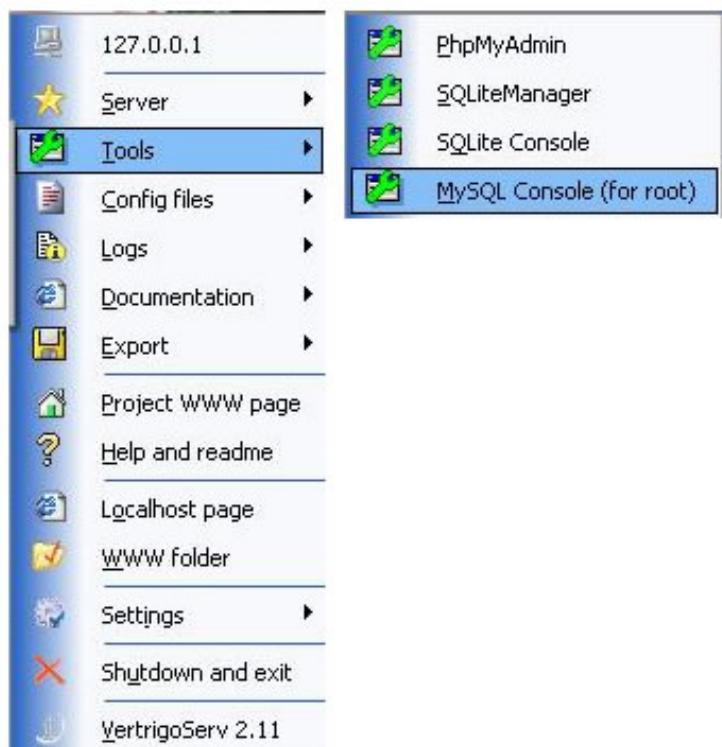
Há duas maneiras de administrarmos os bancos de dados criados: pelo console do MySQL ou utilizando a ferramenta mysqladmin. Utilizaremos o console do MySQL para criação do Banco de Dados.

Para acessar o console do MySQL você deve clicar no ícone do VertigoServ, que se encontra na barra de tarefas. A figura 19 mostra o ícone do Vertrigo.



**Figura 19** – Ícone do Vertrigo na barra de tarefas

Em seguida, escolha a opção **Tools** e clique em **MySQL Console (for root)**. Verifique a figura 20.



**Figura 20** – Abrindo o MySQL Console

Ao se abrir a tela do MySQL Console, é necessário digitar a senha de acesso, nesse caso, o usuário é o root (administrador), a senha padrão após a intalação é vertrigo. Digite a senha e será permitido o acesso para que possamos iniciar a criação de um banco de dados. Observe na figura 21a tela inicial do MySQL Console após o login.

```
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.24-community
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> _
```

**Figura 21** – Tela inicial do MySQL Console

Primeiramente, criaremos o banco de dados (figura 22), em seguida, adicionaremos as tabelas ao banco (figura 23). Com o MySQL Console iniciado, basta digitarmos o comando responsável por criar um banco de dados novo. Observe que, ao final de cada comando, é necessário colocar o ; (ponto e vírgula) para indicar o final do comando que pretendemos executar.

```
CREATE DATABASE nome_do_banco_de_dados;
```

```
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.24-community

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database produto;
Query OK, 1 row affected (0.01 sec)

mysql>
```

**Figura 22 – Criando Banco produto**

Agora criaremos o banco produto (figura 23) para podermos acrescentar as tabelas.

Em algum momento, pode ser que seja necessário mostrar todos os bancos de dados criados, para listá-los, utilize o comando:

```
SHOW DATABASES;
```

Verifique o exemplo de execução desse comando na figura 23, onde serão apresentados os nomes de todos os bancos criados.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| aluno |
| mysql |
| produto |
| test |
+-----+
5 rows in set (0.00 sec)
```

**Figura 23 – Mostrando os Bancos de Dados criados**

A lista de bancos de dados provavelmente será diferente na sua máquina, mas os bancos de dados MySQL e test provavelmente estarão entre eles. O banco de dados MySQL é necessário porque descreve privilégios de acessos de usuários. O banco de dados test é geralmente fornecido como um espaço para que os usuários possam fazer testes.

Verifique na figura 23 a listagem dos banco de dados existentes, se houver necessidade de excluir o banco de dados **aluno**, utilize o comando demonstrado a seguir:

```
DROP DATABASE nome_do_banco_de_dados;
```

Verifique a execução desse comando na figura 24. Observe que utilizamos o comando drop database para fazer a exclusão do banco. A seguir, utilize novamente o comando show databases. Verifique que o banco de dados aluno não mais aparece na listagem, portanto, confirmamos a exclusão deste banco de dados.

```
mysql> drop database aluno;
Query OK, 0 rows affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| produto |
| test |
+-----+
4 rows in set (0.00 sec)
```

**Figura 24** – Excluido Banco de Dados Aluno

Criado o banco de dados produto, iniciaremos a criação das tabelas que farão parte do banco.

Acesse o banco de dados produto (figura 24), executando o comando:

```
USE nome_do_banco_de_dados
```

```
mysql> use produto
Database changed
mysql>
```

**Figura 25** – Acessando o Banco de Dados produto

Perceba que o USE não necessita de um ponto e vírgula. (Você pode terminar tais declarações com um ponto e vírgula, se preferir; isso não irá ocasionar nenhum tipo de erro). A instrução USE é especial em outra maneira, também: deve ser usada em uma única linha. Para conseguir utilizar um banco de dados, é necessário ter permissão de acesso a este banco, neste caso, utilizaremos o usuário root, para termos acesso a todos os bancos.

Para iniciar a criação das tabelas para o banco de dados produto, é necessário conhecer os tipos de dados que poderemos utilizar para a definição dos campos de cada tabela.



## 4 TIPOS DE DADOS NO MYSQL

O MySQL suporta vários tipos de dados e podemos agrupá-los em três grandes grupos:

- Tipos Numéricos;
- Tipos de Data;
- Tipos de Cadeia.

### 4.1 Tipos numéricos

Os tipos de dados numéricos podem ser divididos em números com casas decimais ou números que não tem casas decimais. Verifique os principais tipos numéricos na tabela a seguir.

| Tipo de Dados       | Descrição                                                                                                                                                                                                      |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>TinyInt</b>      | é um número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -128 até 127. Sem sinal, a margem de valores é de 0 até 255.                                                               |
| <b>Bit ou Bool</b>  | um número inteiro que pode ser 0 ou 1.                                                                                                                                                                         |
| <b>SmallInt</b>     | número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -32768 até 32767. Sem sinal, a margem de valores é de 0 até 65535.                                                              |
| <b>MediumInt</b>    | número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -8.388.608 até 8.388.607. Sem sinal, a margem de valores é de 0 até 16777215.                                                   |
| <b>Integer, Int</b> | número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -2147483648 até 2147483647. Sem sinal, a margem de valores é de 0 até 429.496.295                                               |
| <b>BigInt</b>       | número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -9.223.372.036.854.775.808 até 9.223.372.036.854.775.807. Sem sinal, a margem de valores é de 0 até 18.446.744.073.709.551.615. |

| Tipo de Dados                | Descrição                                                                                                                                                                                        |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Float</b>                 | Número com casas decimais de precisão simples. Os valores válidos vão desde -3.402823466E+38 até -1.175494351E-38,0 ou desde 175494351E-38 até 3.402823466E+38.                                  |
| <b>xReal, Double</b>         | Número com casas decimais de dupla precisão. Os valores permitidos vão desde 1.7976931348623157E+308 até -2.2250738585072014E-308, 0 e desde 2.2250738585072014E-308 até 1.7976931348623157E+308 |
| <b>Decimal, Dec, Numeric</b> | Número com casas decimais desempacotado. O número é armazenado como uma cadeia.                                                                                                                  |

Observe na tabela abaixo o tamanho de armazenamento necessário para criação de um campo utilizando os tipos de dados listados:

| Tipo de Campo           | Tamanho de Armazenamento               |
|-------------------------|----------------------------------------|
| <b>TINYINT</b>          | 1 byte                                 |
| <b>SMALLINT</b>         | 2 bytes                                |
| <b>MEDIUMINT</b>        | 3 bytes                                |
| <b>INT</b>              | 4 bytes                                |
| <b>INTEGER</b>          | 4 bytes                                |
| <b>BIGINT</b>           | 8 bytes                                |
| <b>FLOAT(X)</b>         | 4 a 8 bytes                            |
| <b>FLOAT</b>            | 4 a 8 bytes                            |
| <b>DOUBLE</b>           | 8 bytes                                |
| <b>DOUBLE PRECISION</b> | 8 bytes                                |
| <b>REAL</b>             | 8 bytes                                |
| <b>DECIMAL(M,D)</b>     | M+2 bytes se D > 0, M+1 bytes se D = 0 |
| <b>NUMERIC(M,D)</b>     | M+2 bytes se D > 0, M+1 bytes se D = 0 |

## 4.2 Tipos Data

Quando existir a necessidade de armazenar datas, porém o MySQL não verifica se a data que foi armazenada é válida ou não. Simplesmente comprova que o mês está compreendido entre 0 e 12 e que o dia está compreendido entre 0 e 31.

Verifique os principais tipos data na tabela a seguir.

| Tipo de Dados    | Descrição                                                                                                                                                                                                                                              |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Date</b>      | tipo data, armazena uma data. A margem de valores vai desde o 1 de Janeiro de 1001 ao 31 de dezembro de 9999. O formato de armazenamento é de ano-mes-dia.                                                                                             |
| <b>DateTime</b>  | Combinação de data e hora. A margem de valores vai desde o 1 de Janeiro de 1001 às 0 horas, 0 minutos e 0 segundos ao 31 de Dezembro de 9999 às 23 horas, 59 minutos e 59 segundos. O formato de armazenamento é de ano-mes-dia horas:minutos:segundos |
| <b>TimeStamp</b> | Combinação de data e hora.                                                                                                                                                                                                                             |
| <b>Time</b>      | armazena uma hora. O formato de armazenamento é 'HH:MM:SS'.                                                                                                                                                                                            |
| <b>Year</b>      | armazena um ano. O campo pode ter tamanho dois ou tamanho 4 dependendo de se queremos armazenar o ano com dois ou quatro algarismos.                                                                                                                   |

O formato de armazenamento depende do tamanho do campo:

| Tamanho | Formato                                   |
|---------|-------------------------------------------|
| 14      | AnoMesDiaHoraMinutoSegundo aaaammddhhmmss |
| 12      | AnoMesDiaHoraMinutoSegundo aammddhhmmss   |
| 8       | AnoMesDia aaaammdd                        |
| 6       | AnoMesDia aammdd                          |
| 4       | AnoMes aamm                               |
| 2       | Ano aa                                    |

Verifique o tamanho de armazenamento para os campos criados utilizando os tipos de data:

| Tipo de Campo | Tamanho de Armazenamento |
|---------------|--------------------------|
| DATE          | 3 bytes                  |
| DATETIME      | 8 bytes                  |
| TIMESTAMP     | 4 bytes                  |
| TIME          | 3 bytes                  |
| YEAR          | 1 byte                   |

### 4.3 Tipos de Cadeia

Para armazenar valores de cadeia de caracteres, podemos utilizar os tipos de dados apresentados na tabela a seguir.

| Tipo de Dados                  | Descrição                                                                                                                                              |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Char(n)</b>                 | armazena valor de caracter de tamanho fixo. Valores inferiores ao tamanho definido serão deixados em branco. Poderá conter desde 0 até 255 caracteres. |
| <b>VarChar(n)</b>              | armazena valor de caracter de tamanho variável. Valores inferiores ao tamanho definido serão suprimidos. Poderá conter desde 0 até 255 caracteres.     |
| <b>Blob e Text</b>             | um texto com um máximo de 65535 caracteres                                                                                                             |
| <b>MediumBlob e MediumText</b> | um texto com um máximo de 16.777.215 caracteres.                                                                                                       |
| <b>LongBlob e LongText</b>     | um texto com um máximo de caracteres 4.294.967.295.                                                                                                    |
| <b>Enum</b>                    | campo que pode ter um único valor ou uma lista de valores. O tipo Enum aceita até 65535 valores diferentes.                                            |
| <b>Set</b>                     | um campo que pode conter nenhum, um ou vários valores de uma lista. A lista pode ter um máximo de 64 valores.                                          |

Dentro dos tipos de cadeia, podem-se distinguir dois subtipos, o tipo Test e o tipo Blob (Binary Large Object) A diferença entre um tipo e outro é o tratamento que recebem na hora de ordená-los e compará-los. No tipo test ordenam-se, sem ter importância, as maiúsculas e as minúsculas, e no tipo blob, ordenam-se tendo em conta as maiúsculas e minúsculas.

Diferença de armazenamento entre os tipos Char e VarChar

| Valor     | CHAR(4) | Armazenamento | VARCHAR(4) | Armazenamento |
|-----------|---------|---------------|------------|---------------|
| ''        | ''      | 4 bytes       | ''         | 1 byte        |
| 'ab'      | 'ab'    | 4 bytes       | 'ab'       | 3 bytes       |
| 'abcd'    | 'abcd'  | 4 bytes       | 'abcd'     |               |
| 'abcdefg' | 'abcd'  | 4 bytes       | 'abcd'     | 5 bytes       |

**Síntese**

Nesta aula você conheceu o histórico do banco de dados MySQL, principais características, utilização para o gerenciamento de aplicações que necessitem da utilização de um banco de dados de grande capacidade.

Utilizamos os recursos do VertigoServ, instalado anteriormente para a criação de um banco de dados. Também conhecemos os comandos necessários para a criação de um banco de dados, objetivando a criação de tabelas.

Finalmente, aprendemos como utilizar os comandos MySQL para excluir um banco de dados existente, além de conhecer os principais tipos de dados que serão utilizados para a criação das tabelas.

Na próxima aula adicionaremos as tabelas ao banco de dados, assim como, alteraremos e excluiremos tabelas, usando os comandos MySQL.

**Exercícios Propostos****1) Assinale a Alternativa incorreta:**

- a. ( ) O MySQL tornou-se o mais popular banco de dados open source do mundo e é um sério competidor para os maiores sistemas de banco de dados para aplicações de pequeno e médio porte.
- b. ( ) Quando surgiu o MySQL era um substituto para o sistema de banco de dados mSQL.
- c. ( ) O MySQL é desenvolvido por um conjunto de empresas, sendo administradas pelos desenvolvedores do MySQL, com o objetivo de fornecer serviços relacionados ao MySQL.
- d. ( ) O símbolo do MySQL é um golfinho, o nome do golfinho foi escolhido através

**2) Coloque V – Verdadeiro ou F – Falso em relação as características do MySQL:**

- a. ( ) É um banco de dados desenvolvido apenas para a plataforma Windows.
- b. ( ) Excelente desempenho e estabilidade além de ser pouco exigente quanto a

recursos de hardware.

- c. ( ) Facilidade de uso.
- d. ( ) É necessário comprar uma licença de uso para gerenciamento de um servidor de banco de dados.
- e. ( ) Suporte a vários tipos de tabelas (como MyISAM e InnoDB), cada um específico para um fim.

**3) Coloque V – Verdadeiro ou F – Falso:**

- a. ( ) O comando **show databases**; é utilizado para visualizar os bancos de dados existentes, será apresentada uma listagem com os bancos criados.
- b. ( ) Para iniciar a utilização de um banco de dados em MySQL é necessário apenas utizar o comando **create database** e logo em seguida iniciar a criação das tabelas para este banco.
- c. ( ) O comando **drop database nome\_do\_banco**; é utilizado para alterar o nome de um banco de dados já criado anteriormente.
- d. ( ) O comando **use nome\_do\_banco**; deve ser utilizado após a criação do banco de dados, deste modo estamos mostrando ao MySQL qual o banco de dados estaremos utilizando.
- e. ( ) Para excluir um banco de dados MySQL utilizamos o comando **delete**.

**4) Assinale a(s) Alternativa(s) correta(s) em relação a utilização dos comandos MySQL:**

- a. ( ) Para criar um novo banco de dados chamado exercício utilizamos o comando: **create new database exercicio;**
- b. ( ) Para visualizar os bancos de dados criados utilizamos o comando: **show databases nome\_do\_banco;**
- c. ( ) Para excluir um banco de dados é necessário utilizar o comando drop associado ao nome do banco de dados a ser excluído utilizando o comando a seguir para excluir o banco exercício: **drop database exercicio;**
- d. ( ) Após utilizar o comando **create database** para criar o banco de dados, utilizamos o comando **use** para iniciar a utilização do banco, para o banco criar o banco exercício e utiliza-lo usamos os comandos: **create database exercicio; e use exercicio.**

## Aula 5

**CRIANDO, ALTERANDO E EXCLUINDO TABELAS NO MySQL****Objetivos da aula**

Ao final desta aula, você deverá ser capaz de:

- Identificar os comandos utilizados para a criação de tabelas em um banco de dados MySQL;
- Identificar os comandos utilizados para exclusão de tabelas;
- Identificar os comandos utilizados para alterar, excluir ou acrescentar campos em uma tabela.

**Conteúdos da aula**

Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Criando tabelas;
- Visualizando tabelas e os campos;
- Alterando tabelas;
- Excluindo tabelas;
- Utilizando índices;
- Exercícios propostos.



Olá! Você aprimorou um pouco mais seus conhecimentos, criando um banco de dados utilizando o MySQL Console, além disso, você conheceu os tipos de dados que utilizaremos para definir o tipo de cada um dos campos da tabelas que criaremos. Neste momento, adicionaremos as tabelas ao banco

5

TUPY Virtual

de dados criado, além de verificar os comandos utilizados para visualizar e excluir as tabelas. Vamos continuar nosso estudo do banco de dados MySQL com a criação da tabelas.

Boa Aula!



## 1 CRIANDO TABELAS

Já criamos um banco de dados, porém é necessário criar as tabelas para esse banco. Criaremos duas tabelas para o banco produto com os campos descritos a seguir. Criaremos a tabela produto e a tabela tipo\_prod.

**Tabela tipo\_prod**

| Campos    | Tipos de Dados |
|-----------|----------------|
| Cod_tipo  | int(5)         |
| Desc_tipo | varchar(50)    |

Para a tabela tipo\_prod criaremos dois campos, um para armazenar o código e outro a descrição do tipo de produto.

**Tabela Produto**

| Campos     | Tipos de Dados |
|------------|----------------|
| cod_prod   | int(5)         |
| cod_tipo   | int(5)         |
| desc_tipo  | varchar(50)    |
| preco_prod | decimal(16,2)  |
| quant_prod | int(4)         |

Para a tabela produto, criaremos cinco campos, conforme descrito na tabela acima, o campo cod\_tipo é chave estrangeira na tabela produto, o relacionamento existente entre as duas tabelas é um relacionamento 1 – N.



**Figura 26** – Relacionamento entre Produto e Tipo\_prod

Para criar uma tabela no MySQL, utilizaremos o comando **CREATE TABLE**. Verifique o exemplo, na figura 27, onde apresentaremos a criação da tabela tipo\_prod.

```
mysql> create table tipo_prod(cod_tipo int<5> not null
-> primary key auto_increment,
-> desc_tipo varchar<50> not null);
Query OK, 0 rows affected (0.12 sec)
```

**Figura 27 – Criando a tabela tipo\_prod**

A sintaxe do comando **CREATE TABLE** é:

**CREATE TABLE nome\_da\_tabela(campo tipo(tamanho) null/not null, campo2 tipo(tamanho) null/not null ..., primary key (campo));**

Para cada tabela é necessário especificar um nome, em seguida deve ser detalhado o nome de cada campo a ser criado, assim como o tipo de dado e o tamanho do campo. A opção null/not null irá indicar se o campo aceita valores nulos ou não. O item primary key está especificando qual campo será a chave primária dessa tabela.

Para a criação da tabela produto, verifique o comando na figura 28, onde, do mesmo modo que na tabela tipo\_prod é necessário especificar o nome, tipo de dado e tamanho para cada campo a ser criado, além de especificar o campo que será chave primária. Verifique que, para o campo cod\_tipo (chave estrangeira) foi utilizado o comando **references** e mostrado que a referência para este campo é o campo da tabela tipo\_prod o campo cod\_tipo. Quando utilizamos esse comando, estamos fazendo a ligação do campo cod\_tipo, da tabela produto, chave estrangeira - com o campo cod\_tipo da tabela tipo\_prod - chave primária - no lado 1 do relacionamento entre produto e tipo\_prod. O relacionamento entre as duas tabelas é criado nesse momento. Verifique todas estas informações na figura 28.

```
mysql> create table produto(cod_prod int<5> not null
-> primary key auto_increment,
-> cod_tipo int<5> not null references tipo_prod(cod_tipo),
-> desc_prod varchar<50> not null,
-> preco_prod decimal<16,2> not null,
-> quant_prod int<4> not null);
Query OK, 0 rows affected (0.17 sec)
```

**Figura 28 – Criando a tabela produto**

A seguir, observe a criação de mais uma tabela, para que possamos aplicar os comandos de alteração e exclusão de tabelas. Crie a tabela teste, conforme mostra a figura 29. Observe, contudo, que esta tabela não está relacionada com as outras duas tabelas criadas.

```
mysql> create table teste(cod varchar(3) not null primary key,  
-> desc_teste varchar(35) not null,  
-> data_teste date not null);  
Query OK, 0 rows affected (0.19 sec)
```

Figura 29 – Criando tabela teste

Agora você já aprendeu a criar tabelas para um banco de dados em MySQL, a seguir, conhiceremos outros comandos para continuarmos trabalhando com tabelas e aprendendo um pouco mais sobre o MySQL.



## 2 VISUALIZANDO TABELAS E CAMPOS

Primeiramente vamos verificar como visualizar as tabelas de um banco de dados. Para isso, usaremos o comando **SHOW TABLES** de duas maneiras, conforme mostra a figura 30.

```
mysql> show tables;  
+-----+  
| Tables_in_produto |  
+-----+  
| produto          |  
| teste           |  
| tipo_prod        |  
+-----+  
3 rows in set (0.35 sec)  
  
mysql> show tables from produto;  
+-----+  
| Tables_in_produto |  
+-----+  
| produto          |  
| teste           |  
| tipo_prod        |  

```

Figura 30 – Visualizando tabelas do banco produto.

Ao criarmos o banco produto, na aula 4, utilizamos o comando **use** para acessarmos o banco criado. Ao utilizarmos apenas o comando **SHOW TABLES** serão mostradas as tabelas do banco em uso, ou seja, do banco produto. Porém, podemos

utilizar este comando especificando de qual banco de dados queremos visualizar as tabelas, usando a seguinte sintaxe:

**SHOW TABLES** from nome\_do\_banco\_de\_dados;

Para visualizar os campos de uma tabela podemos utilizar o comando **describe** ou **desc**, usando a seguinte sintaxe:

**DESCRIBE nome\_da\_tabela;**

| Field      | Type          | Null | Key | Default | Extra          |
|------------|---------------|------|-----|---------|----------------|
| cod_prod   | int(5)        | NO   | PRI | NULL    | auto_increment |
| cod_tipo   | int(5)        | NO   |     |         |                |
| desc_prod  | varchar(50)   | NO   |     |         |                |
| preco_prod | decimal(16,2) | NO   |     |         |                |
| quant_prod | int(4)        | NO   |     |         |                |

5 rows in set (0.04 sec)

**Figura 31 – Visualizando tabela produto usando comando describe**

Na figura 31, apresentamos o resultado da execução do comando **describe** para a tabela **produto**. Verifique que aparecem: o nome de cada campo, o tipo de dado. A opção null recebeu o valor NO, pois na criação do campo foi especificado como not null. No item KEY demonstra-se qual campo é a chave primária.

Existe outro comando que mostrará o mesmo resultado do comando **describe**, verifique na figura 32.

| mysql> show columns from produto; |               |      |     |         |                |
|-----------------------------------|---------------|------|-----|---------|----------------|
| Field                             | Type          | Null | Key | Default | Extra          |
| cod_prod                          | int(5)        | NO   | PRI | NULL    | auto_increment |
| cod_tipo                          | int(5)        | NO   |     |         |                |
| desc_prod                         | varchar(50)   | NO   |     |         |                |
| preco_prod                        | decimal(16,2) | NO   |     |         |                |
| quant_prod                        | int(4)        | NO   |     |         |                |

5 rows in set (0.27 sec)

**Figura 32 – Visualizando tabela produto usando comando show columns**

As informações mostradas na figura 32 são as mesmas da figura 31, porém com o comando **SHOW COLUMNS**, que deve respeitar a sintaxe.

**SHOW COLUMNS** from nome\_da\_tabela;

Para visualizarmos os campos de uma tabela, podemos utilizar ainda o comando

SHOW FIELDS, respeitando a sintaxe:

**SHOW FIELDS from nome\_da\_tabela;**

Verifique o exemplo na figura 33, para visualizar os campos da tabela produto.

| mysql> show fields from produto; |               |      |     |         |       |                |
|----------------------------------|---------------|------|-----|---------|-------|----------------|
| Field                            | Type          | Null | Key | Default | Extra |                |
| cod_prod                         | int(5)        | NO   | PRI | NULL    |       | auto_increment |
| cod_tipo                         | int(5)        | NO   |     |         |       |                |
| desc_prod                        | varchar(50)   | NO   |     |         |       |                |
| preco_prod                       | decimal(16,2) | NO   |     |         |       |                |
| quant_prod                       | int(4)        | NO   |     |         |       |                |
| imposto_prod                     | decimal(9,2)  | YES  |     | NULL    |       |                |

6 rows in set (0.01 sec)

**Figura 33 – Visualizando tabela produto usando comando show fields**

Agora que já aprendemos como visualizar as tabelas de um banco de dados e os campos de cada tabela, vamos verificar como alterar a estrutura de uma tabela.



### 3 ALTERANDO TABELAS

Quando existir a necessidade de fazer alterações na estrutura de uma tabela criada, utilizamos o comando ALTER TABLE. Podemos adicionar um novo campo utilizando este comando, observe o exemplo na figura 34 e a sintaxe do comando apresentada a seguir:

**ALTER TABLE nome\_da\_tabela add nome\_do\_novo\_campo tipo\_de\_dado(tamanho);**

| mysql> alter table produto add imp_prod decimal(10,2); |               |      |     |         |                |
|--------------------------------------------------------|---------------|------|-----|---------|----------------|
| Query OK, 0 rows affected (0.32 sec)                   |               |      |     |         |                |
| Records: 0 Duplicates: 0 Warnings: 0                   |               |      |     |         |                |
| mysql> describe produto;                               |               |      |     |         |                |
| Field                                                  | Type          | Null | Key | Default | Extra          |
| cod_prod                                               | int(5)        | NO   | PRI | NULL    | auto_increment |
| cod_tipo                                               | int(5)        | NO   |     |         |                |
| desc_prod                                              | varchar(50)   | NO   |     |         |                |
| preco_prod                                             | decimal(16,2) | NO   |     |         |                |
| quant_prod                                             | int(4)        | NO   |     |         |                |
| imp_prod                                               | decimal(10,2) | YES  |     | NULL    |                |

6 rows in set (0.09 sec)

**Figura 34 – Alterando tabela produto inserindo novo campo**

Para adicionar um novo campo, é necessário informar o nome do novo campo, o tipo de dado e tamanho. O item **add** está informando ao comando **ALTER TABLE** que será acrescentado um novo campo. No exemplo da figura 34 foi inserido um campo chamado `imp_prod`, verifique que logo a seguir foi utilizado o comando **describe** para mostrar os campos da tabela `produto`, mostrando, deste modo, o novo campo inserido.

Além de inserir um novo campo, o comando **ALTER TABLE**, permite alterar as informações de um campo já existente. Verifique o exemplo na figura 35 e a sintaxe do comando:

```
ALTER TABLE nome_da_tabela change nome_atual_do_campo novo_nome_
do_campo tipo_de_dado(tamanho);
```

```
mysql> alter table produto change imp_prod imposto_prod decimal(9,2);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe produto;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cod_prod | int(5) | NO | PRI | NULL | auto_increment |
| cod_tipo | int(5) | NO |   |   |   |
| desc_prod | varchar(50) | NO |   |   |   |
| preco_prod | decimal(16,2) | NO |   |   |   |
| quant_prod | int(4) | NO |   |   |   |
| imposto_prod | decimal(9,2) | YES |   | NULL |   |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Figura 35 – Alterando campo da tabela `produto`

Para alterar um campo é necessário informar o nome do novo campo, o tipo de dado e o tamanho. O item **change** informa ao comando **ALTER TABLE** que haverá alteração em um campo. No exemplo da figura 35 foi alterado o campo chamado `imp_prod`, o nome do campo foi alterado para `imposto_prod` e o tamanho do campo também foi alterado, poderíamos também ter alterado o tipo de dado desse campo. Verifique que, logo a seguir, foi utilizado o comando **describe** para mostrar os campos da tabela `produto` e alteração feita no campo `imp_prod`.

Com o comando **ALTER TABLE**, conseguimos excluir um campo da tabela, antes, porém, utilize o comando **ALTER TABLE** para inserir um novo campo chamado `teste_prod`, conforme mostra a figura 36, em seguida faremos a exclusão desse campo.

```
mysql> alter table produto add teste_prod int;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe produto;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cod_prod | int(5) | NO | PRI | NULL | auto_increment |
| cod_tipo | int(5) | NO | | | |
| desc_prod | varchar(50) | NO | | | |
| preco_prod | decimal(16,2) | NO | | | |
| quant_prod | int(4) | NO | | | |
| imposto_prod | decimal(9,2) | YES | | NULL | |
| teste_prod | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

**Figura 36 – Inserindo campo teste\_prod**

Verifique na figura 37 como utilizar o comando ALTER TABLE para excluir um campo. Veja, a seguir, a sintaxe do comando:

**ALTER TABLE nome\_da\_tabela drop nome\_do\_campo;**

```
mysql> alter table produto drop teste_prod;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe produto;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cod_prod | int(5) | NO | PRI | NULL | auto_increment |
| cod_tipo | int(5) | NO | | | |
| desc_prod | varchar(50) | NO | | | |
| preco_prod | decimal(16,2) | NO | | | |
| quant_prod | int(4) | NO | | | |
| imposto_prod | decimal(9,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

**Figura 37 – Excluindo campo da tabela**

Para excluir um campo, é necessário informar o nome do campo que será excluído. O item **drop** informa ao comando **ALTER TABLE** que será excluído um campo. No exemplo da figura 37 foi excluído o campo **teste\_prod**. Verifique que, logo a seguir, foi utilizado o comando **describe**, para mostrar os campos da tabela **produto** e o campo **teste\_prod** excluído.

Para finalizarmos essa aula, estudaremos o comando para excluir uma tabela do nosso banco de dados.



## 4 EXCLUINDO TABELAS

Para excluir uma tabela, utilizamos o comando DROPTABLE, porém é necessário muito cuidado na utilização desse comando, pois apagaremos todas as informações armazenadas, quando apagamos uma tabela. Verifique o exemplo de utilização deste comando na figura 38, a sintaxe do comando é apresentada a seguir:

```
DROP TABLE nome_da_tabela;
```

```
mysql> drop table teste;
Query OK, 0 rows affected (0.00 sec)
```

**Figura 38 – Excluindo tabela Teste.**

No figura 38 demonstramos o comando utilizado para excluir a tabela teste, você poderá utilizar o comando SHOW TABLES para mostrar as tabelas do banco e verificar a exclusão da tabela teste.



## 5 UTILIZANDO ÍNDICES

Um índice é um arquivo estruturado que facilita o acesso aos dados armazenados em um banco de dados. Um índice, em um campo correto, aumentará a velocidade de uma consulta, consideravelmente.

O objetivo de um índice é trabalhar da mesma forma que uma pasta com separadores alfabéticos em um arquivo. Se você estiver procurando por um nome em um arquivo organizado com separadores alfabéticos, poderá ir diretamente à letra correspondente ao nome que deseja encontrar, sem ter a necessidade de passar por todos os nomes que estiverem nas outras letras. Isso tornará sua pesquisa mais fácil de ser realizada, sem a necessidade de examinar registros desnecessários.

Os índices auxiliarão imensamente no momento de desenvolver uma consulta, porém, se a tarefa for inserir um registro, o processo será um pouco mais lento do que a mesma tarefa em uma tabela que não tenha índices, pois o registro deverá ser inserido na ordem correta, de acordo com o índice definido. Nesse caso, verificamos

que a criação de índices acelera o acesso a dados para consultas de SELECT, mas tornam lentas as consultas de INSERT, UPDATE e DELETE.

Por padrão, o MySQL cria um índice, quando declaramos uma coluna como chave primária. Podemos ter até 16 índices em uma tabela, não sendo aconselhável a utilização de um número muito grande de índices por tabela.

A sintaxe do comando para criar um índice é:

**CREATE INDEX nome\_do\_indice ON nome\_da\_tabela(lista\_de\_campos);**

Para a criação de um índice é necessário especificar-lhe o nome, a tabela para a qual o índice será criado e os campos que o formarão, lembrando que um índice poderá ser formado por mais de um campo.

Para excluir um índice utilizamos o comando drop. Observe a sintaxe do comando:

**DROP INDEX nome\_do\_indice ON nome\_da\_tabela;**

## Síntese



Nesta aula utilizamos os comandos MySQL para criação, alteração e exclusão de tabelas em um banco de dados.

Utilizamos o comando CREATE TABLE para criar uma tabela para o banco de dados. Verificamos que essa ação exige que especifiquemos os campos que farão parte da tabela, além das definições de chave primária e dos relacionamentos com o campo chave estrangeira.

Na utilização do comando ALTER TABLE foi apresentada a possibilidade de alterar um campo já criado na tabela, além de adicionar ou excluir um campo. Utilizamos o comando DROP TABLE para excluir uma tabela existente em nosso banco de dados. Finalmente abordamos a utilização e criação de índices em um banco de dados.

Na próxima aula utilizaremos os comandos MySQL para manipulação dos registros de um banco de dados.

**Exercícios Propostos**

**1) Coloque V – Verdadeiro ou F – Falso em relação às características do MySQL:**

- a. ( ) Para criar a tabela aluno, em um banco de dados, utilizamos o comando **insert table aluno;**
- b. ( ) Quando utilizamos o comando **show tables**, serão listadas as tabelas existentes no banco de dados que estiver em uso.
- c. ( ) Através do comando **alter table** podemos somente alterar ou incluir um campo em uma tabela. Para excluir um campo é necessário excluir a tabela, através do comando **drop table**.
- d. ( ) Para visualizar os campos de uma tabela podemos utilizar o comando **describe nome\_da\_tabela;**

**2) Assinale a alternativa correta para utilização do comando alter table para alterar o nome do campo valor para valor\_final e o tipo de dados para decimal, este campo pertence a tabela produto:**

- a. ( ) **ALTER TABLE** produto **change** valor decimal (10,2);
- b. ( ) **ALTER TABLE** produto **add** valor\_final decimal (10,2);
- c. ( ) **ALTER TABLE** produto **drop** valor\_final decimal (10,2);
- d. ( ) **ALTER TABLE** produto **change** valor valor\_final decimal (10,2);

**3) Assinale a Alternativa correta em relação à utilização do comando create table para criar uma tabela com os seguintes campos:**

|              |             |
|--------------|-------------|
| Funcionarios | ▼           |
| codFunc:     | INTEGER     |
| nomeFunc:    | VARCHAR(45) |
| endeFunc:    | VARCHAR(45) |
| codDept:     | INTEGER     |

**Dica:** Na tabela Funcionários o campo codFunc é chave primária e o campo codDept é chave estrangeira, mostrando que existe um relacionamento entre a tabela Funcionários e a tabela Departamento.

- a. ( ) **CREATE TABLE** Funcinarios(codFunc int(3) not null,  
nomeFunc varchar(45) not null, endeFunc varchar(45), codDepto int(3),  
**primary key**(codFunc));
- b. ( ) **CREATE TABLE** Funcinarios(codFunc int(3) not null,  
nomeFunc varchar(45) not null, endeFunc varchar(45),  
codDepto int(3) not null references Departamento(codDepto),  
**primary key**(codFunc));
- c. ( ) **CREATE TABLE** Funcinarios(codFunc varchar(3) not null,  
nomeFunc varchar(45) not null, endeFunc varchar(45),  
codDepto int(3) not null references Departamento(codDepto),  
**primary key**(codFunc));
- d. ( ) **CREATE TABLE** Funcinarios(codFunc int(3) not null,  
nomeFunc varchar(45) not null, endeFunc varchar(45),  
codDepto int(3) not null references Departamento(codDepto),  
**primary key**(codDepto));

## Aula 6

# MANIPULAÇÃO DE REGISTROS EM MySQL

## Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Identificar os comandos utilizados para a manipulação das informações em uma tabela;
- Identificar os comandos utilizados para inserir, alterar, excluir e mostrar informações em uma tabela.

## Conteúdos da aula



Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Inserindo registros;
- Alterando registros;
- Excluindo registros;
- Mostrando registros;
- Exercícios propostos.



Olá! Chegamos a nossa sexta aula. Seus conhecimentos em MySQL já lhe permitiram criar um banco de dados, criar as tabelas para estes banco de dados, fazer a alteração nos campos da tabela, excluir tabelas. Chegou o momento de manipular os registros armazenados no banco de dados. Primeiramente será necessário inserir registros no banco de dados, desse

# 6

Tutoria Virtual

modo, poderemos iniciar a utilização dos comandos para manipulação dos registros. Alteraremos um registro, mostrando as informações, excluindo registros e, deste modo, aprimorando nossos conhecimentos em MySQL. Vamos à manipulação dos registros.

Boa Aula!



## 1 INSERINDO REGISTROS

Criamos as tabelas para nosso banco de dados na aula anterior, porém precisamos inserir registros nessas tabelas. Para tanto utilizamos o comando INSERT, verifique a seguir a sintaxe deste comando:

```
INSERT INTO nome_da_tabela(campo1, campo2, ... , campoN)  
VALUES('valor_campo1', 'valor_campo2', ..., 'valor_campoN');
```

ou

```
INSERT INTO nome_da_tabela  
VALUES('valor_campo1', 'valor_campo2', ..., valor_campoN);
```

Para inserir registros em uma tabela podemos utilizar o comando de duas maneiras: verificando que não é necessário apresentar o nome de todos os campos; é possível inserir informações apenas sobre o nome da tabela e os valores que cada campo deverá receber. Podemos suprimir o nome dos campos, quando inserirmos valores para todos os campos da tabela. Se houver a necessida de inserir valores em apenas alguns campos, então será necessário utilizar o nome dos campos.

Nas figuras 39 e 40 apresentaremos o exemplo da utilização do comando INSERT para inserir registros nas tabelas do banco produto, verifique os exemplos.

```
mysql> insert into tipo_prod values('','alimentacao');  
Query OK, 1 row affected, 1 warning (0.17 sec)  
  
mysql> insert into tipo_prod values('','vestuario');  
Query OK, 1 row affected, 1 warning (0.00 sec)
```

**Figura 39** – Inserindo registros na tabela tipo\_prod

Os registros foram inseridos na tabela tipo\_prod, conforme mostra a figura 39, porém precisamos inserir registros na tabela produto. Na figura 40 apresentamos o comando para inserir registros na tabela produto, verifique o exemplo e insira registros nas duas tabelas.

```
mysql> insert into produto(cod_prod,cod_tipo,desc_prod,preco_prod,quant_prod
-> imposto_prod)
-> values(, 2, 'calça jeans', 56.70, 12, 5.32);
Query OK, 1 row affected, 1 warning (0.00 sec)
```

**Figura 40 – Inserindo registros na tabela produto**

Agora já conhecemos o comando para inserir registros em um banco de dados MySQL. Para que se torne possível a utilização de comandos de manipulação de registros, é importante inseri-los em cada uma das tabelas, nesse momento.



## 2 ALTERANDO REGISTROS

Para alterar o valor dos campos em um ou mais registros, usamos o comando UPDATE. Verifique a sintaxe do comando a seguir:

**UPDATE nome\_da\_tabela SET campo1 = valor1, campo2 = valor2, ..., campoN = valorN WHERE condição1 and condição2 and condiçãoN;**

Verificar na sintaxe apresentada, que podemos alterar o valor de um ou mais campos, especificando o nome da tabela e dos campos, cujos valores serão atualizados.

A cláusula WHERE especifica uma condição, mas é necessário que exista uma condição. O comando UPDATE será executado normalmente caso não haja a cláusula WHERE. Se não houver uma condição, todos os registros da tabela sofrerão a alteração que estiver descrita no comando UPDATE.

Na figura 41, mostramos um exemplo da utilização do comando UPDATE. Alteraremos o campo quan\_prod para receber o valor 30 e o campo preco\_prod para receber o valor 4.90, porém respeitamos a condição apresentada na cláusula WHERE. Fazendo essa alteração apenas para os produtos que tiverem código do tipo igual a 3. Verifique a utilização do comando na figura 41.

```
mysql> update produto
-> set quant_prod=30, preco_prod=4.90
-> where cod_tipo = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Figura 41 – Comando Update



### 3 EXCLUINDO REGISTROS

O comando utilizado para apagar registros é o DELETE, responsável por apagar um ou mais registros de uma tabela sempre que a condição for satisfeita. A sintaxe do comando DELETE é:

**DELETE FROM nome\_da\_tabela**

**WHERE condição and condição2 and condiçãoN;**

O comando DELETE encontrará a condição que foi definida junto a cláusula WHERE estará testando essa condição. Se a condição for verdadeira executará a exclusão dos registros que atenderam à condição. Deste modo, poderão ser excluídos um ou mais registros, de acordo com a condição testada.

Na figura 42 mostraremos a utilização do comando DELETE para excluir apenas um registro, pois excluiremos o tipo do produto que tenha o código do tipo igual ao valor 4. Se a condição definida tivesse sido cod\_tipo < 4 estaríamos excluindo todos os tipos de produtos que tivessem seu código menor que 4. Verifique o exemplo do comando DELETE na figura 42.

```
mysql> delete from tipo_prod where cod_tipo=4;
Query OK, 1 row affected (0.00 sec)

mysql> select * from tipo_prod;
+-----+-----+
| cod_tipo | desc_tipo |
+-----+-----+
| 1 | alimentacao |
| 2 | vestuario |
| 3 | brinquedo |
+-----+-----+
3 rows in set (0.00 sec)
```

Figura 42 – Excluindo registros



## 4 SELECIONANDO E MOSTRANDO REGISTROS

O comando SELECT é responsável por selecionar e mostrar os registros das tabelas, podendo ou não ter uma condição a ser testada. Esse comando é extremamente importante, pois mostra as informações cadastradas no banco. Observe a sintaxe desse comando apresentada a seguir:

**SELECT** campo1, campo2, ..., campoN

**FROM** tabela1, tabela2, ..., tabelaN

**WHERE** condição1 and condição2 and condiçãoN;

Na sintaxe apresentada verificamos que podemos definir vários campos para serem mostrados. Se for necessário mostrar a informação de todos os campos, substituiremos o nome dos campos e pelo símbolo \* (indica que queremos que sejam mostradas as informações de todos dos campos).

Verificamos também que podemos selecionar as informações de várias tabelas, cujas utilizações devem ser listadas após o FROM. Em seguida é possível colocar uma ou mais condições para serem testadas e então mostrar as informações.

Na figura 43 utilizamos o comando SELECT, porém não especificamos nenhuma condição, desse modo, mostraremos todos os produtos e apresentaremos

```
a: mysql> select * from produto;
+-----+-----+-----+-----+-----+
| cod_prod | cod_tipo | desc_prod | preco_prod | quant_prod | imposto_prod |
+-----+-----+-----+-----+-----+
|     2    |     2    | calça jeans |   56.70   |      12     |    5.32    |
|     1    |     2    | bermuda     |   35.40   |       2     |    2.75    |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

**Figura 43 – Selecinando e mostrando todos os Produtos**

Suponha que não queremos mostrar as informações de todos os campos da tabela produto. Assim, após o comando SELECT, listamos o nome dos campos que queremos mostrar. Nesse momento ainda não temos nenhuma condição fazendo com que todos os registros cadastrados na tabela produto sejam mostrados. Veja a figura 44.

```
mysql> select cod_prod, desc_prod, preco_prod, quant_prod from produto;
+-----+-----+-----+-----+
| cod_prod | desc_prod | preco_prod | quant_prod |
+-----+-----+-----+-----+
|     2   | calça jeans |      56.70 |       12 |
|     1   | bermuda    |      35.40 |        2 |
|     3   | carrinho hot wheels |      4.90 |       30 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

**Figura 44** – Selecionando todos os Produtos

Na figura 45, utilizamos o comando SELECT para mostrar todos os tipos de produtos, mostrando as informações de todos os campos da tabela tipo\_produto.

```
mysql> select * from tipo_prod;
+-----+-----+
| cod_tipo | desc_tipo |
+-----+-----+
|     1   | alimentacao |
|     2   | vestuario   |
|     3   | brinquedo   |
|     4   | higiene e limpeza |
+-----+-----+
4 rows in set (0.00 sec)
```

**Figura 45** – Mostrando os tipos de produto

Mostramos todos os registros da tabela tipo\_prod, pois não foi especificado nenhuma condição, além disso, serão apresentadas as informações de todos os campos da tabela.

Agora utilizamos a opção GROUP BY que definirá a forma como os dados serão agrupados na hora da visualização. Desse modo, na Figura 46 apresentamos o comando para listar as informações de todos os campos da tabela tipo\_prod que serão agrupadas ou organizadas pela informação do campo desc\_tipo. Quando não utilizamos a opção GROUP BY, as informações são organizadas pelo campo que é chave primária, nesse caso, seriam organizados pelo campo cod\_tipo.

```

mysql> select * from tipo_prod group by desc_tipo desc;
+-----+-----+
| cod_tipo | desc_tipo |
+-----+-----+
| 2 | vestuario |
| 3 | brinquedo |
| 1 | alimentacao |
+-----+
3 rows in set <0.00 sec>

mysql> select * from tipo_prod group by desc_tipo asc;
+-----+-----+
| cod_tipo | desc_tipo |
+-----+-----+
| 1 | alimentacao |
| 3 | brinquedo |
| 2 | vestuario |
+-----+
3 rows in set <0.00 sec>

```

**Figura 46** – Usando Group By para agrupar os dados

Para a opção GROUP BY é necessário informar o campo onde queremos agrupar as informações. Podemos também informar se queremos mostrar as informações organizadas, de forma ascendente (asc) ou descendente (desc), permitindo que os dados sejam organizados em ordem alfabética de A até Z ou em ordem alfabética de Z até A.

Quando utilizamos o comando SELECT podemos utilizar uma condição, para tanto precisamos da cláusula WHERE. No exemplo da figura 47 utilizaremos a cláusula WHERE junto com o comando BETWEEN, para comparar uma informação entre um intervalo de valores. No exemplo, serão mostrados os produtos cuja quantidade permanece (quant\_prod) entre 5 e 20. Veja:

```

mysql> select * from produto where quant_prod between 5 and 20;
+-----+-----+-----+-----+-----+-----+
| cod_prod | cod_tipo | desc_prod | preco_prod | quant_prod | imposto_prod |
+-----+-----+-----+-----+-----+-----+
| 2 | 2 | calça jeans | 56.70 | 12 | 5.32 |
+-----+-----+-----+-----+-----+-----+
1 row in set <0.19 sec>

```

**Figura 47** – Utilizando Between para definir uma condição

O mesmo exemplo da figura 47 pode ser desenvolvido sem utilizar o comando BETWEEN, utilizando os operadores relacionais para fazer a comparação, a sintaxe do comando será:

```
SELECT * from produto WHERE quant_prod > 5 and quant_prod < 20;
```

Neste exemplo, substituímos o comando BETWEEN pelos operadores > (maior que) e < (menor que), o resultado será o mesmo apresentado no comando da figura 47.

Podemos utilizar a função **AVG** para retornar a média de um conjunto de valores, junto com o comando SELECT. É necessário especificar o campo onde estão os valores que desejamos calcular a média. Verifique a utilização da função AVG na figura 48.

```
mysql> select avg as "Média Preços" from produto;
+-----+
! Média Preços !
+-----+
! 32.333333 !
+-----+
1 row in set <0.00 sec>
```

**Figura 48** – Utilizando função AVG para mostrar média

No exemplo, utilizamos a função AVG para mostrar a média dos preços dos produtos, e a cláusula **AS** para definir um rótulo para a coluna onde será apresentado o resultado da função AVG.

A função SUM é utilizada para retornar a soma de um conjunto de valores, também é necessário especificar o campo a ser utilizado para calcular a soma dos valores, verifique o exemplo de utilização desta função na figura:

```
mysql> select sum as "Total de Produtos" from produto;
+-----+
! Total de Produtos !
+-----+
! 44 !
+-----+
1 row in set <0.00 sec>
```

**Figura 49** – Utilizando função SUM para mostrar soma .

A cláusula **AS** está definindo o rótulo Total de Produtos para a coluna onde será apresentado o resultado da soma dos valores armazenados no campo `quant_prod`.

A função COUNT é utilizada para contar e mostrar o total de ocorrências de um grupo de registros. No exemplo da figura 50, conta-se o total de ocorrências do campo `cod_prod`, mostrando que existem três ocorrências, ou seja, três produtos cadastrados.

```
mysql> select count(cod_prod) as "Quantidade de Produtos" from produto;
+-----+
| Quantidade de Produtos |
+-----+
|            3           |
+-----+
1 row in set <0.00 sec>
```

**Figura 50** – Utilizando função COUNT para mostrar a quantidade de registros

A função MAX mostrará o maior valor entre um grupo de valores. Para utilizá-la é necessário especificar o campo a ser analisado. Verifique o exemplo na figura 51, onde a função MAX é utilizada para mostrar o maior valor do campo preco\_prod, ou seja, o maior preço entre os produtos cadastrados.

```
mysql> select max(preco_prod) as "Maior Preço" from produto;
+-----+
| Maior Preço |
+-----+
|      56.70   |
+-----+
1 row in set <0.00 sec>
```

**Figura 51** – Utilizando função MAX para mostrar o maior valor

A função MIN mostrará o menor valor entre um grupo de valores. Para utilizá-la será necessário especificar o campo a ser analisado. No exemplo da figura 52 a função MIN mostra o menor preço entre todos os produtos cadastrados.

```
mysql> select min(preco_prod) as "Menor Preço" from produto;
+-----+
| Menor Preço |
+-----+
|      4.90   |
+-----+
1 row in set <0.03 sec>
```

**Figura 52** – Utilizando função MIN para mostrar o menor valor



## 5 MOSTRANDO REGISTROS DE MÚLTIPLAS TABELAS: INNER JOIN

Em um banco dos dados, é comum dados armazenados em tabelas separadas, mas relacionados, garantindo, que o banco de dados não armazene dados redundantes.

No exemplo das tabelas produto e tipo de produto, se as informações estivessem armazenadas em uma mesma tabela para cada produto estaríamos armazenando as informações do tipo de produto, isso resultaria em um banco de dados armazenando informações duplicadas para os tipos que tivessem vários produtos.

No entanto, em algumas situações, há necessidade de combinar dados de múltiplas tabelas em um único conjunto de dados. Referido como junção (join) das tabelas, sendo feito por meio da operação INNER JOIN em uma consulta SELECT.

Uma operação INNER JOIN combina os registros de duas ou mais tabelas, testando os valores correspondentes em um campo que seja comum entre essas tabelas, ou seja, que fazem parte do relacionamento entre estas tabelas. A sintaxe do comando SELECT utilizando a cláusula INNER JOIN é:

```
SELECT campo1, campo2, ... , campoN  
FROM tabela1  
INNER JOIN tabela2  
ON tabela1.nomeCampo = tabela2.nomeCampo;
```

Na sintaxe do comando, verificamos que são relacionadas duas tabelas e, em seguida, os campos comuns entre estas tabelas são comparados, deste modo, serão apresentados os registros em que a informação no campo correspondente à tabela 1 seja igual a informação no campo correspondente à tabela 2. Verifique o exemplo apresentado a seguir para as tabelas produto e tipo\_produto:

```
SELECT desc_prod, preco_prod, desc_tipo  
FROM produto  
INNER JOIN tipo_prod  
ON produto.cod_tipo = tipo_prod.cod_tipo;
```

Mostramos no exemplo informações de campos da tabela produto e campos da tabela tipo\_prod, desse modo, utilizamos a cláusula INNER JOIN para relacionar as duas tabelas, usando o comando ON para relacionar os campos comuns entre estas tabelas.

**Síntese**

Nesta aula utilizamos os comandos do banco de dados MySQL para manipulação de registros. Utilizamos o comando INSERT para inserirmos nas tabelas as informações de acordo com cada um dos campos.

Utilizamos o comando SELECT para visualizar e manipular as informações armazenadas em cada tabela.

Agora você já está apto(a) para criar um banco de dados, usando os recursos do MySQL, aprendidos nas aulas anteriores.

**Exercícios Propostos**

**Utilize as tabelas apresentadas a seguir para responder os exercícios.**

| Clientes    | Estado   |
|-------------|----------|
| codCli      | siglaEst |
| nomeCli     | descEst  |
| sexoCli     |          |
| idadeCli    |          |
| siglaEstado |          |

**1) Assinale a alternativa correta comando INSERT:**

- a. ( ) **INSERT INTO** Estado(siglaEst, descEst);
- b. ( ) **INSERT** Estado(siglaEst, descEst) VALUES('SC', 'Santa Catarina');
- c. ( ) **INSERT** Estado( 'SC', 'Santa Catarina');
- d. ( ) **INSERT INTO** Estado(siglaEst, descEst) VALUES('SC', 'Santa Catarina');

**2) Assinale a alternativa correta UPDATE:**

Fazer uma consulta UPDATE que altere para 30, a idade dos clientes que tem a sigla do Estado igual a SC e o sexo igual a M (masculino).

- a. ( ) **UPDATE** Clientes **SET** idadeCli = 30 **WHERE** siglaEstado = 'SC' ;
- b. ( ) **UPDATE** Clientes **SET** idadeCli = 30 **WHERE** siglaEstado = 'SC' **AND** sexoCli = 'M' ;
- c. ( ) **UPDATE** Clientes **idadeCli = 30 WHERE** siglaEstado = 'SC' **AND** sexoCli = 'M' ;
- d. ( ) **UPDATE** Clientes **WHERE** siglaEstado = 'SC' **AND** sexoCli = 'M' ;

**3) Assinale a Alternativa correta comando DELETE:**

Fazer uma consulta DELETE para excluir todos os Clientes com mais de 45 anos.

- a. ( ) **DELETE FROM** Clientes **where** idadeCli = 45;
- b. ( ) **DELETE** Clientes **where** idadeCli = 45;

- c. ( ) DELETE FROM Clientes;
- d. ( ) DELETE FROM Clientes where idadeCli > 45;

**4) Assinale a Alternativa correta comando SELECT:**

Fazer uma consulta SELECT para mostrar o média das idades de todos os Clientes que moram no estado SP.

- a. ( ) SELECT SUM(idadeCli) FROM Clientes WHERE siglaEstado='SP';
- b. ( ) SELECT COUNT(idadeCli) FROM Clientes WHERE siglaEstado='SP';
- c. ( ) SELECT AVG(idadeCli) FROM Clientes WHERE siglaEstado='SP';
- d. ( ) SELECT MEDIA(idadeCli) FROM Clientes WHERE siglaEstado='SP';

**5) Assinale a Alternativa correta comando SELECT – INNER JOIN:**

Fazer uma consulta SELECT para mostrar o nome, e descrição do estado e todos os clientes.

- a. ( ) **SELECT** nomeCli, idadeCli, descEst  
**FROM** Clientes **INNER JOIN** Estado  
    **ON** Cliente.siglaEstado = Estado.siglaEst;
- b. ( ) **SELECT** nomeCli, idadeCli, descEst  
**FROM** Clientes, Estado  
    **ON** Cliente.siglaEstado = Estado.siglaEst;
- c. ( ) **SELECT** nomeCli, idadeCli, descEst  
Clientes **INNER JOIN** Estado  
    **ON** Cliente.siglaEstado = Estado.siglaEst;
- d. ( ) **SELECT** nomeCli, idadeCli, descEst  
**FROM** Clientes **INNER JOIN** Estado;

## Aula 7

# CONSTRUINDO UM PROJETO

**Objetivos da aula**

Ao final desta aula, você deverá ser capaz de:

- Aplicar os comandos de criação de banco de dados e tabelas para a criação de um projeto com diversas tabelas;
- Aplicar os comandos de manipulação de registro no banco de dados criado.

**Conteúdos da aula**

Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Apresentando o Modelo Entidade Relacionamento do Projeto;
- Criando o Banco de Dados;
- Criando as tabelas;
- Manipulando registros;
- Exercícios propostos.



Olá! Chegamos a nossa sétima aula. Agora já conhecemos os comandos necessários para fazer a criação de um banco de dados e os comandos utilizados para manipulação dos registros. Através dos conhecimentos adquiridos nas aulas anteriores criaremos um banco de dados em MySQL com diversas tabelas, onde visualizaremos e realizaremos o relacionamento entre as tabelas, estruturando um banco de

7

TUPY Virtual

dados para que, futuramente, você possa desenvolver uma aplicação que use como base de dados o banco desenvolvido em MySQL.

Boa Aula!

## 1 APRESENTANDO O MODELO ER DO PROJETO



Criaremos um banco de dados que contenha seis tabelas, todas relacionadas entre si. Podemos visualizar o modelo ER do projeto na figura 52, criaremos as seguintes tabelas: Funcionários, Operação, Clientes, Automóveis, Modelos e Marca.

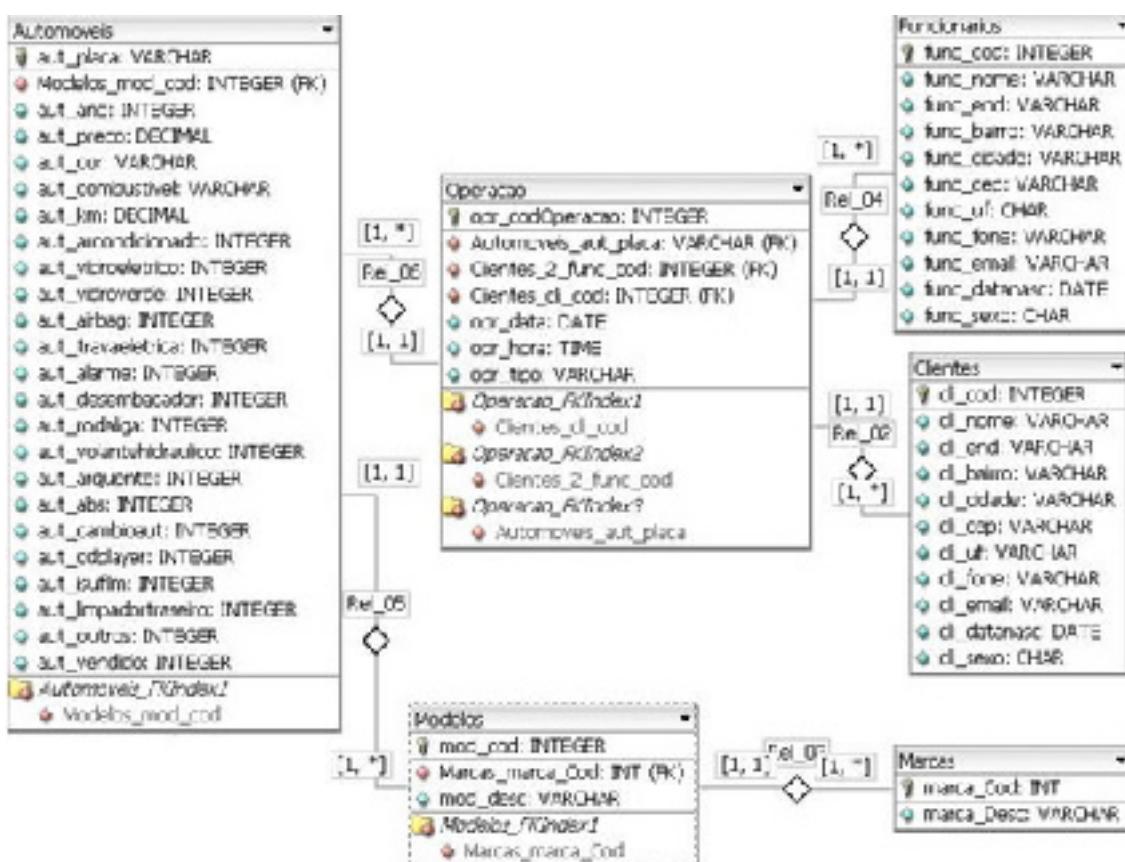


Figura 52 – Modelo ER do projeto

Vamos iniciar a criação do banco de dados ER. Na figura 53, demonstramos o login no MySQL Console, em seguida, a criação do banco. Após criar o banco, utilize o comando **USE** para acessar o banco de dados criado e para iniciar a criação das tabelas. Observe a figura 53.

```

Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 23 to server version: 5.0.24-community

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database revenda;
Query OK, 1 row affected (0.00 sec)

mysql> use revenda;
Database changed

```

**Figura 53 – Criando o banco revenda**

Nesse momento, criamos o banco **revenda**, agora é necessário criar as tabelas para esse banco. Iniciaremos pela criação da tabela **marcas**, observe na figura 54 o comando utilizado para criação da tabela, os campos criados, os tipos de dados para cada um dos campos, além do campo definido como chave primária.

```

mysql> create table marcas(marca_cod int<3> not null auto_increment,
-> marca_desc varchar<35> not null,
-> primary key(marca_cod));
Query OK, 0 rows affected (0.17 sec)

```

**Figura 54 – Criando tabela marcas**

O campo **marca\_cod** está sendo definido como chave primária, o comando **not null** está definindo que esses campos não poderão ficar sem preenchimento no momento em que se cadastrarem os registros na tabela **marcas**. O comando **auto\_increment**, para o campo **marca\_cod**, e que, para cada novo registro, será incrementado o código (somado 1 ao valor do código anterior).

Na tabela **modelos**, apresentamos o comando utilizado para criação da tabela. Veja figura 55.

```

mysql> create table modelos(mod_cod int<4> not null auto_increment,
-> mod_desc varchar<50> not null,
-> mod_marcacod int<3> not null references marcas(marca_cod),
-> primary key(mod_cod));
Query OK, 0 rows affected (0.08 sec)

```

**Figura 55 – Criando tabela modelos**

A tabela **modelos** está relacionada à tabela **marcas**, e o campo **mod\_marcacod** é o campo chave estrangeira que será relacionado com o campo **marca\_cod**, chave primária na tabela **marca**. O comando responsável por este relacionamento é:

**references marcas(marca\_cod)**

Para criar a tabela clientes, observe o comando utilizado na figura 56, onde demonstramos o tipo de dado de cada um dos campos e o campo definido como chave primária.

```
mysql> create table clientes(cli_cod int<4> not null,
-> cli_nome varchar<50> not null,
-> cli_end varchar<50> not null,
-> cli_bairro varchar<35> not null,
-> cli_cidade varchar<35> not null,
-> cli_cep varchar<9> not null,
-> cli_ue char<2> not null,
-> cli_fone varchar<15>,
-> cli_email varchar<50>,
-> cli_datanasc date not null,
-> cli_sexo char<1> not null,
-> primary key(cli_cod));
Query OK, 0 rows affected (0.06 sec)
```

**Figura 56** – Criando a tabela clientes

Mostramos o comando utilizado para criar a tabela funcinários na figura 57. O campo definido como chave primária será o func\_cod (código do funcinário).

```
mysql> create table funcionarios(func_cod int<4> not null,
-> func_nome varchar<50> not null,
-> func_end varchar<50> not null,
-> func_bairro varchar<35> not null,
-> func_cidade varchar<35> not null,
-> func_cep varchar<9> not null,
-> func_ue char<2> not null,
-> func_fone varchar<15>,
-> func_email varchar<50>,
-> func_datanasc date not null,
-> func_sexo char<1> not null,
-> primary key(func_cod));
Query OK, 0 rows affected (0.09 sec)
```

**Figura 57** – Criando a tabela funcionarios

Para criar a tabela automóveis, observe o comando utilizado. Esta tabela está relacionada com a tabela modelo. O campo aut\_modelocod é o campo chave estrangeira, sendo relacionado com o campo mod\_cod da tabela modelo através do comando **references**, conforme está detalhado na Figura 58.

```

mysql> create table automoveis(aut_placa varchar(8) not null,
-> aut_modelocod int<4> not null references modelos(mod_cod),
-> aut_ano int<4> not null,
-> aut_preco decimal<16,2> not null,
-> aut_cor varchar<35> not null,
-> aut_combustivel varchar<25> not null,
-> aut_km decimal<16,2> not null,
-> aut_arcondicionado int<1>,
-> aut_vidroeletrico int<1>,
-> aut_vidroverde int<1>,
-> aut_airbag int<1>,
-> aut_travaeletrica int<1>,
-> aut_alarme int<1>,
-> aut_desembacdador int<1>,
-> aut_rodaliga int<1>,
-> aut_volantehidraulico int<1>,
-> aut_arquente int<1>,
-> aut_abs int<1>,
-> aut_cambioaut int<1>,
-> aut_cdplayer int<1>,
-> aut_isufilm int<1>,
-> aut_limpadortraseiro int<1>,
-> aut_outros int<1>,
-> aut_vendido int<1>;
-> primary key(aut_placa));
Query OK, 0 rows affected (0.16 sec)

```

**Figura 58** – Criando a tabela automoveis

A placa do automóvel será o campo chave primária da tabela. Verifique os campos e tipos de dados na figura 58.

A próxima tabela a ser criada será a tabela operacao, estará relacionada com as tabelas automóveis, funcionarios e clientes. Para cada um desses relacionamentos teremos um campo chave estrangeira na tabela operacao. Os relacionamentos serão feitos ligando a chave estrangeira com o campo chave primária em cada uma destas tabelas. Observe na figura 59 os relacionamentos entre estas tabelas.

```

mysql> create table operacao(opr_codoperacao int<5> not null,
-> opr_placa varchar<8> not null references automoveis(aut_placa),
-> opr_funccod int<4> not null references funcionarios(func_cod),
-> opr_clientecod int<4> not null references clientes(cli_cod),
-> opr_data date not null,
-> opr_hora time not null,
-> opr_tipo varchar<25>;
-> primary key(opr_codoperacao));
Query OK, 0 rows affected (0.08 sec)

```

**Figura 59** – Criando a tabela operação

Após criar a tabela operação concluímos a criação do banco revenda. Para visualizar as tabelas criadas, podemos utilizar o comando show tables, conforme mostra a figura 60.

```
mysql> show tables;
+-----+
| Tables_in_revenda |
+-----+
| automoveis
| clientes
| funcionarios
| marcas
| modelos
| operacao
+-----+
6 rows in set (0.03 sec)
```

**Figura 60** – Visualizando as tabelas do banco revenda

Para realizarmos as consultas no banco revenda, é necessário que sejam inseridos registros em cada uma das tabelas, utilize o comando **INSERT** para inserir registros para a tabela do banco revenda. Insira vários registros para cada uma das tabelas. Se existir alguma dúvida em relação à utilização do comando **INSERT** você deverá verificar o item Inserindo Registros, na aula 6 deste livro.



## 2 MANIPULANDO REGISTROS

Vamos desenvolver algumas consultas para o banco revenda utilizando o comando **SELECT**, devemos lembrar que o banco revenda possui seis tabelas, as quais foram criadas anteriormente e adicionados registros. Vamos dividir nossas consultas por tabela.

### 2.1 Tabela Marcas

A tabela Marcas possui apenas dois campos, onde são armazenados o código da marca e a descrição, para esta tabela vamos fazer uma consulta que mostre todas as Marcas Cadastradas, verifique esta consulta na figura 61.

```
mysql> SELECT * FROM marcas;
+-----+-----+
| marca_cod | marca_desc |
+-----+-----+
| 1 | Citroen |
| 2 | Renault |
| 3 | Fiat |
| 4 | Ford |
+-----+-----+
4 rows in set (0.00 sec)
```

**Figura 61** – Mostrando todos os registros da tabela Marcas

## 2.2 Tabela Modelos

A tabela Modelos possui os campos para armazenar as informações do código e da descrição do modelo, além do código da Marca que é um campo chave estrangeira.

Para esta tabela vamos fazer uma consulta que mostre todos os modelos cadastrados para a marca que possui código igual 1, ou seja, da marca Citroen.

```
mysql> SELECT * FROM modelos WHERE mod_marcacod='1';
+-----+-----+-----+
| mod_cod | mod_desc | mod_marcaod |
+-----+-----+-----+
| 1 | XSARA PICASSO | 1 |
| 2 | C5 BREAK | 1 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

**Figura 62** – Mostrando todos os Modelos com código da marca igual a 1

Entre as tabelas Marcas e Modelos existe um relacionamento como apresentado no início dessa aula na figura 52, desse modo, podemos utilizar a cláusula INNER JOIN em conjunto com o comando SELECT. Desse modo, vamos fazer uma consulta que mostre a descrição do modelo e a descrição da marca para todos os modelos cadastrados.

```
mysql> SELECT mod_desc,marca_desc
-> FROM modelos INNER JOIN marcas
-> ON modelos.mod_marcaod = marcas.marca_cod;
+-----+-----+
| mod_desc | marca_desc |
+-----+-----+
| XSARA PICASSO | Citroen |
| C5 BREAK | Citroen |
| LOGAN | Renault |
| CLIO | Renault |
| SCENIC | Renault |
| PALIO | Fiat |
| STRADA ADVENTURE | Fiat |
| SIENA | Fiat |
| FIESTA | Ford |
| FOCUS | Ford |
| RANGER | Ford |
+-----+-----+
11 rows in set (0.03 sec)
```

**Figura 63** – Fazendo consulta relacionando as tabelas Marcas e Modelos

## 2.3 Tabela Automóveis

A tabela Automóveis possui vários campos para armazenar as características do automóvel, além do código do Modelo que é um campo chave estrangeira, pois

esta tabela está relacionada com a tabela Modelos.

Vamos fazer uma consulta que mostra a descrição do modelo, o ano e o preço de todos os carros que têm o preço menor que R\$ 45000,00, verifique esta consulta na figura 64.

```
mysql> SELECT mod_desc,aut_ano,aut_preco
-> FROM automoveis INNER JOIN modelos
-> ON automoveis.aut_modelocod = modelos.mod_cod
-> WHERE aut_preco < 45000;
+-----+-----+-----+
| mod_desc | aut_ano | aut_preco |
+-----+-----+-----+
| PALIO   | 2003   | 18000.00 |
| FIESTA  | 2004   | 29900.00 |
| SIENA   | 2003   | 20000.00 |
| PALIO   | 1999   | 14900.00 |
+-----+-----+-----+
4 rows in set (0.03 sec)
```

**Figura 64** – Mostrando registros da tabela Automovies

Podemos fazer uma consulta UPDATE para alterar o preço de todos os carros que têm o preço menor que R\$ 45000,00 para acrescentar R\$ 2500,00 ao preço destes automoveis, verifique esta consulta na figura 65.

```
mysql> UPDATE automoveis SET aut_preco=aut_preco+2500
-> WHERE aut_preco<45000;
Query OK, 4 rows affected (0.03 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT mod_desc,aut_ano,aut_preco
-> FROM automoveis INNER JOIN modelos
-> ON automoveis.aut_modelocod = modelos.mod_cod
-> WHERE aut_preco < 45000;
+-----+-----+-----+
| mod_desc | aut_ano | aut_preco |
+-----+-----+-----+
| PALIO   | 2003   | 20500.00 |
| FIESTA  | 2004   | 32400.00 |
| SIENA   | 2003   | 22500.00 |
| PALIO   | 1999   | 17400.00 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

**Figura 65** – Utilizando consulta UPDATE

Na figura 65, observe a utilização do comando UPDATE, a seguir utilizamos uma consulta SELECT para mostrar os valores, mostrando as alterações feitas através da consulta UPDATE.

## 2.4 Tabela Funcionários e Tabela Clientes

As tabelas Funcionários e Clientes são tabelas que possuem campos para

armazenar as informações referentes aos funcionários e aos clientes, sendo que estas tabelas estão relacionadas com a tabela Operacao.

Vamos fazer uma consulta que mostre o nome, o endereço e o bairro de todos os clientes classificando pelo nome do cliente, verifique a figura 66.

```
mysql> SELECT cli_nome, cli_end, cli_bairro FROM clientes
-> GROUP BY cli_nome;
+-----+-----+-----+
| cli_nome | cli_end | cli_bairro |
+-----+-----+-----+
| ANA      | RUA DA CIDADE | CENTRO
| JOANA    | RUA DAS FLORES | BOA VISTA
| MARCOS ANTONIO | RUA DAS FLORES | CENTRO
+-----+-----+-----+
3 rows in set <0.00 sec>
```

**Figura 66 – Utilizando GROUP BY**

Na consulta da figura 66 a cláusula GROUP BY agrupou a saída dos dados pelo campo cli\_nome.

Para a tabela funcionários faça uma consulta que mostre o nome de todos os funcionários que nasceram entre 01/01/1980 e 01/01/2007, para fazer esta consulta estaremos usando a cláusula BETWEEN, verifique a consulta na figura 67.

```
mysql> SELECT func_nome FROM funcionarios
-> WHERE func_datanasc BETWEEN '1980/01/01' AND '2007/01/01';
+-----+
| func_nome |
+-----+
| PAULO
| PEDRO
+-----+
2 rows in set <0.00 sec>
```

**Figura 67 – Utilizando cláusula BETWEEN**



### Síntese

Nesta aula você desenvolveu na prática um projeto de banco de dados, utilizando o MySQL. Primeiramente, apresentamos o modelo entidade relacionamento do projeto, a seguir, todas as entidades, os relacionamentos e os campos necessários para iniciar a criação do banco.

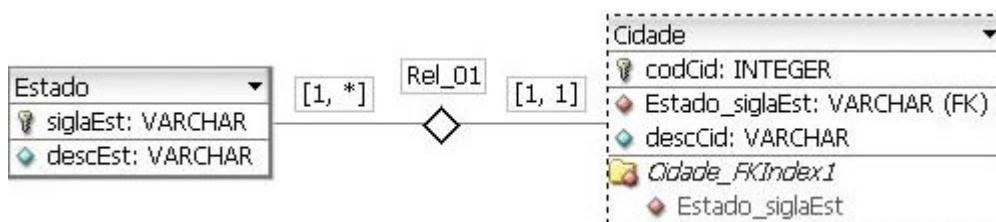
Na próxima aula conheceremos **PhpMyAdmin**, ferramenta de fácil acesso e aprendizado, que nos permite criar banco de dados MySQL.

**Exercícios Propostos**

**1) Assinale a alternativa correta para definir um campo codDisc como chave primária durante a criação da tabela Disciplina:**

- a. ( ) **CREATE TABLE** Disciplina(codDisc int(3) not null,  
descDisc varchar(30) not null,  
**PRIMARY KEY()**);
- b. ( ) **CREATE TABLE** Disciplina(codDisc int(3) not null,  
descDisc varchar(30) not null,  
**PRIMARY KEY(descDisc)**);
- c. ( ) **CREATE TABLE** Disciplina(codDisc int(3) not null,  
descDisc varchar(30) not null,  
**PRIMARY KEY(codDisc)**);
- d. ( ) **CREATE TABLE** Disciplina(codDisc int(3) not null,  
descDisc varchar(30) not null,  
**PRIMARY (codDisc)**);

**2) Utilizando o relacionamento entre as tabelas como base, assinale a alternativa correta do comando para criação da tabela Cidade:**



**Dica:** lembre-se de que o Estado\_siglaEst da tabela Cidade é o campo chave estrangeira, que esta fazendo o relacionamento com o campo siglaEst, que é chave primária na tabela Estado.

- a. ( ) **CREATE TABLE** Cidade(codCid int(3) not null,  
Estado\_siglaEst varchar(2) not null,  
descCid varchar(30) not null,  
**PRIMARY KEY(codCid)**);

- b. ( ) **CREATE TABLE** Cidade(codCid int(3) not null,  
Estado\_siglaEst varchar(2) not null **REFERENCES** Cidade(codCid),  
descCid varchar(30) not null,  
**PRIMARY KEY**(codCid));
- c. ( ) **CREATE TABLE** Cidade(codCid int(3) not null,  
Estado\_siglaEst varchar(2) not null **REFERENCES** Estado(codCid),  
descCid varchar(30) not null,  
**PRIMARY KEY**(codCid));
- d. ( ) **CREATE TABLE** Cidade(codCid int(3) not null,  
Estado\_siglaEst varchar(2) not null **REFERENCES** Estado(siglaEst),  
descCid varchar(30) not null,  
**PRIMARY KEY**(codCid));

## Aula 8

# UTILIZANDO O PhpMyAdmin PARA CRIAR UM BANCO DE DADOS

## Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Criar um novo banco de dados, usando a ferramenta PhpMyAdmin;
- Utilizar o PhpMyAdmin para criar, excluir e alterar tabelas em um banco de dados.

## Conteúdos da aula



Acompanhe os conteúdos desta aula. Se você preferir, assinale-os à medida em que for estudando.

- Conhecendo o PhpMyAdmin;
- Criando o Banco de Dados;
- Criando e excluindo tabelas;
- Exercícios propostos.



Olá! Chegamos a nossa oitava aula. Agora você já conhece todos os comandos necessários para criar um banco de dados em MySQL e também para manipular os registros utilizando os comandos SQL. Nesse momento, conheceremos o PhpMyAdmin, ferramenta utilizada para criação de um banco de dados, sem que seja necessário utilizar os comandos do MySQL Console.

Boa Aula!

8

Tupu Virtual



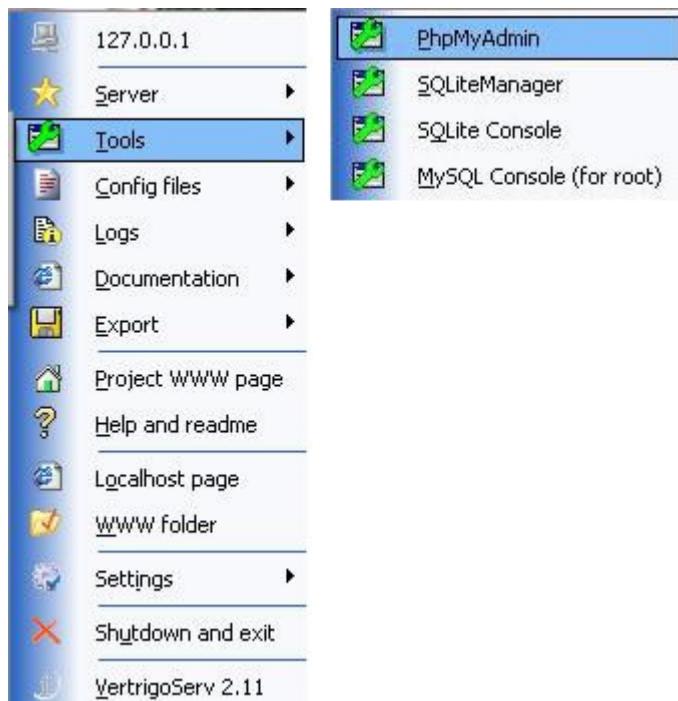
## 1 CONHECENDO O PHPMYADMIN

Para acessar o PhPMYAdmin, você deve clicar no ícone do VertrigoServ, mostrado na barra de tarefas, conforme mostra a figura 68.



**Figura 68** – Ícone do Vertrigo na barra de tarefas

Em seguida, escolha a opção Tools, e clique em PhpMyAdmin. Veja a figura 69.



**Figura 69** – Abrindo o PhpMyAdmin

Ao abrir a tela, precisamos digitar o usuário e a senha para acesso ao PhpMyAdmin. O usuário para acesso é o root, a senha padrão é **vertrigo**. Verifique a tela que será apresentada na figura 70. Clique no botão **OK** para acessar o PhpMyAdmin.



**Figura 70** – Acessando o PhpMyAdmin

Na tela inicial do PhpMyAdmin, apresentada na figura 71, é possível criar um novo banco de dados, alterar ou excluir um banco de dados já existente. Veja na tela inicial a utilização do PhpMyAdmin.

**Figura 71** – Tela inicial do PhpMyAdmin

## 1.1 Criando um banco de dados

Digite um nome para criar um novo banco de dados. A seguir, clique no botão **Criar**. Veja a figura 72, onde criaremos um novo banco chamado empresa.



**Figura 72** - Criando novo Banco de Dados

Agora definiremos as tabelas que farão parte desse novo banco. A figura 73 mostra a tela de consulta SQL com a mensagem: “Nenhuma tabela encontrada no Banco de Dados.” Então criaremos uma tabela para esse banco. Precisamos especificar o nome da nova tabela e o número de campos, em seguida, cliquemos no botão **Executar**.



**Figura 73** – Criando tabelas para o banco

Criaremos a tabela Funcionários para o banco empresa, contendo 4 campos.

Após clicar no botão executar, aparecerá a tela mostrada na figura 74, para definirmos os campos da tabela Funcionários.

| Campo    | Tipo    | Tamanho/Definir <sup>1</sup> | Collation | Atributos |
|----------|---------|------------------------------|-----------|-----------|
| codFunc  | INT     | 3                            |           |           |
| nome     | VARCHAR | 50                           |           |           |
| endereco | VARCHAR | 50                           |           |           |
| codDept  | INT     | 3                            |           |           |

Comentários da tabela:   
Storage Engine: MyISAM  
Collation:

Salvar Ou Adicionar 1 campo(s) Executar

**Figura 74** – Definindo campos para a tabela Funcionarios

Para cada um dos campos é necessário especificar o nome, o tipo de dados e o tamanho do campo. Para o campo codFunc, é necessário escolher a opção Chave Primária. Também é possível selecionar a opção nul/not null para cada um dos campos. Clique no botão Salvar para que os campos sejam acrescentados na tabela Funcionários.

A figura 75 mostra a tela com a consulta SQL utilizada e, em seguida, a lista-gem dos campos criados.

```

Tabela Funcionarios foi criado.

consulta SQL:
CREATE TABLE `Funcionarios` (
  `codFunc` INT(3) NOT NULL ,
  `nome` VARCHAR(50) NOT NULL ,
  `endereco` VARCHAR(50) NOT NULL ,
  `codDepto` INT(3) NOT NULL ,
  PRIMARY KEY (`codFunc`)
) ENGINE = MYISAM ;
  
```

|                          | Campo           | Tipo        | Collation         | Atributos | Nulo | Padrão | Extra | Ações |
|--------------------------|-----------------|-------------|-------------------|-----------|------|--------|-------|-------|
| <input type="checkbox"/> | <b>codFunc</b>  | int(3)      |                   |           | Não  |        |       |       |
| <input type="checkbox"/> | <b>nome</b>     | varchar(50) | latin1_swedish_ci |           | Não  |        |       |       |
| <input type="checkbox"/> | <b>endereco</b> | varchar(50) | latin1_swedish_ci |           | Não  |        |       |       |
| <input type="checkbox"/> | <b>codDepto</b> | int(3)      |                   |           | Não  |        |       |       |

↑ Marcar todos / Desmarcar todos Com marcados:

Figura 75 – Criando os campos da tabela Funcionarios

Vamos adicionar mais uma tabela ao banco empresa. Será a tabela Departamentos. Os campos que devem ser definidos são apresentados na figura 76.

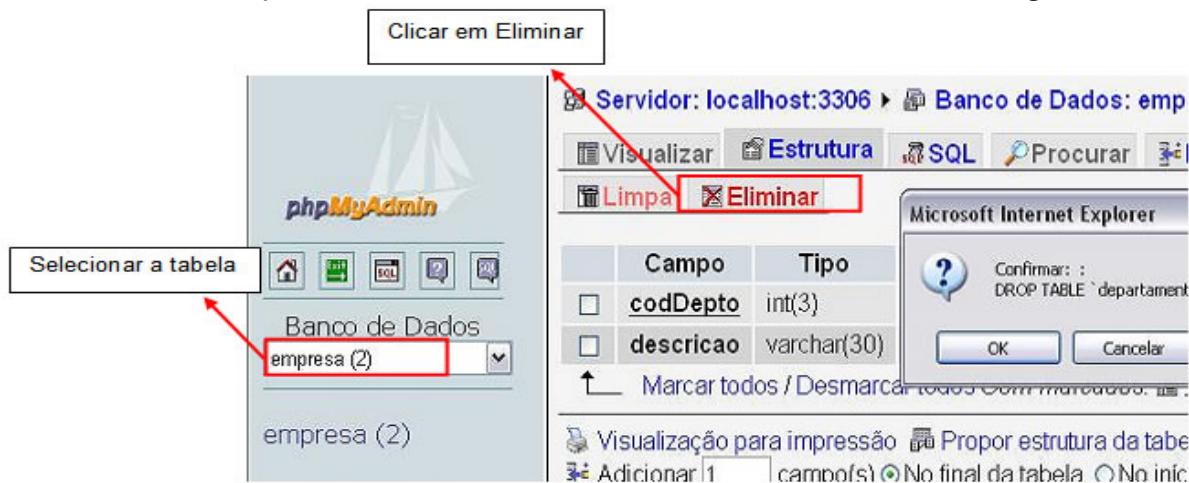
|                          | Campo            | Tipo        | Collation         | Atributos | Nulo | Padrão | Extra | Ações |
|--------------------------|------------------|-------------|-------------------|-----------|------|--------|-------|-------|
| <input type="checkbox"/> | <b>codDepto</b>  | int(3)      |                   |           | Não  |        |       |       |
| <input type="checkbox"/> | <b>descricao</b> | varchar(30) | latin1_swedish_ci |           | Não  |        |       |       |

↑ Marcar todos / Desmarcar todos Com marcados:

Figura 76 – Criando tabela Departamentos

Para **excluir** uma **tabela**, é necessário selecioná-la. Clique no nome da tabela (aparece no menu, ao lado esquerdo da janela do navegador). Em seguida, clique no botão **eliminar**: será apresentada uma mensagem de confirmação da exclusão.

Para confirmar, clique no botão **OK**. A tabela será excluída. Observe a figura 77.



**Figura 77 – Excluindo tabela**

Para **excluir** um **banco de dados**, o processo é semelhante ao de exclusão de tabela. O banco de dados deve ser selecionado, aparecerão listadas as tabelas que pertencem a este banco.

Clique na opção **Eliminar**, será apresentada a mensagem de confirmação de exclusão.

Clique no botão **OK**, o banco selecionado será excluído. Verifique essas informações na figura 78.



**Figura 78 – Excluindo Banco de Dados**

Com as opções apresentadas nesta aula, verificamos que é possível criar um banco de dados utilizando a ferramenta PhpMyAdmin. Lembre-se: para cada ação desenvolvida pela ferramenta, executamos um comando SQL. O conhecimento desses comandos é essencial para criação e administração de um banco de dados MySQL.

**Síntese**

Nesta aula você conheceu a ferramenta PhpMyAdmin para a criação de um banco de dados MySQL e verificou a possibilidade de criação do banco, das tabelas, além da manipulação e cadastro das informações.

Você deve ter verificado que, para cada uma das ações executadas na ferramenta, foi utilizado o comando SQL.

Com esta aula, terminamos o módulo Interface com o usuário – Utilizando o banco de dados MySQL.

Continue estudando e se aperfeiçoando na criação de novos bancos de dados, pois a aplicação desse recurso para criação de aplicações Web é de extrema importância.

Excelente Estudo e Um Grande Abraço!

**REFERÊNCIAS**

**BUYENS**, Jim.; Aprendendo MySQL e PHP. SÃO PAULO: Editora Makron Books, 2002.

**DATE**, C. J.; Introdução a Sistemas de Banco de Dados. 8<sup>a</sup> Ed., RIO DE JANEIRO: Editora Campus,2004.

**MUTO**, Cláudio Adonai.; PHP & MySQL – Guia Introdutório. 2<sup>a</sup> Ed., RIO DE JANEIRO: Editora Brasport,2004.

**HERNANDEZ**, Michael; Aprenda a Projetar seu Próprio Banco de Dados. SÃO PAULO: Editora Makron Books,2000.

**HEUSER**, Carlos Alberto; Projeto de Banco de Dados. 4<sup>a</sup> Ed., PORTO ALEGRE: Editora Sagra Luzzato,2001.

**SILBERSCHATZ**, Abraham, **KORTH**, Henry F, **SUDARSHAN**, S; Sistema de Banco de Dados. 5<sup>a</sup> Ed., RIO DE JANEIRO: Editora Campus,2006.

<http://www.mysqlbrasil.com.br/porquemysql> Acessado em: 24/06/2007

<http://www.mysql.com/> Acessado em: 24/06/2007

Copyright © Tupy Virtual 2007

Nenhuma parte desta publicação pode ser reproduzida por qualquer meio sem a prévia autorização desta instituição.

Autor: Francini Reitz Spanceski

Interface Com o Usuário - MySQL: Material didático / Francini Reitz Spanceski

Design institucional: Thiago Vedoi de Lima; Cristiane de Oliveira - Joinville: Tupy Virtual, 2007

Ficha catalográfica elaborada pela Biblioteca Universitária Tupy Virtual

## Créditos

### SOCIESC – Sociedade Educacional de Santa Catarina

#### Tupy Virtual – Ensino a Distância

Rua Albano Schmidt, 3333 – Joinville – SC – 89206-001

Fone: (47)3461-0166

E-mail: ead@sociesc.org.br

Site: www.sociesc.org.br/portalead

### Design Gráfico

Thiago Vedoi de Lima

### Equipe Didático-Pedagógica

Francini Reitz Spanceski

### EDIÇÃO – MATERIAL DIDÁTICO

#### Diretor Geral

Sandro Murilo Santos

#### Professor Conteudista

Francini Reitz Spanceski

#### Diretor de Administração

Vicente Otávio Martins de Resende

#### Design Institucional

Thiago Vedoi de Lima

Cristiane Oliveira

#### Diretor de Ensino, Pesquisa e Extensão

Roque Antonio Mattei

#### Ilustração Capa

Thiago Vedoi de Lima

#### Diretor do Instituto Superior Tupy

Wesley Masterson Belo de Abreu

#### Projeto Gráfico

Equipe Tupy Virtual

#### Diretor da Escola Técnica Tupy

Luiz Fernando Bublitz

#### Revisão Ortográfica

Nádia Fátima de Oliveira

#### Coordenador da Escola Técnica Tupy

Alexssandro Fossile

Alan Marcos Blenke

#### Coordenador do Curso

Juliano Prim

#### Coordenador de Projetos

José Luiz Schmitt

#### Revisora Pedagógica

Nádia Fátima de Oliveira

---

### EQUIPE TUPY VIRTUAL

Raimundo Nonato Gonçalves Robert

Wilson José Mafra

Thiago Vedoi de Lima

Cristiane Oliveira