

Mysql

“Quanto mais informações você
busca, mais você evolui.”

Vitor Ramos

SQL

Structured Query Language - Linguagem de Consulta Estruturada

- Desenvolvida pela IBM, nos anos 70;
- Inicialmente chamada SEQUEL
- Parte do SystemR (protótipo de BD relacional)

Concorrência

- Relational Software Inc - lançou o Oracle (baseado em SQL)

SQL

- surgem vários SGBD baseados em SQL
- Problemas - falta de padrões
- American National Standard Institute (ANSI) e International Standards Organizations (ISO) definem padrão SQL.

SQL1 -1986, com modific. em 1989

SQL2 -1992

SQL3 –1999

e uma última atualização em 2003

Categorias

- **DDL – Data Definition Language**
 - Composta pelos comandos Create, Alter e Drop
 - Responsável por dar forma ao banco de dados
- **DML – Data Manipulation Language**
 - Comandos Select, Insert, Delete e Update
 - Responsável por manipular os dados acrescentando, modificando, apagando e fazendo consultas
- **DCL – Data Control Language**
 - Subgrupo da DML, composta pelos comandos Grant e Revoke
 - Responsável por controlar acesso dos usuários aos dados.

MySQL

- Michael Widenius desenvolveu o UNIREG em 1979.
- Em 1994 a TcX começou a desenvolver aplicações baseadas na Web tendo como base o banco UNIREG e teve problemas.
- TcX procurou outro banco, o mSQL(de David Hughes) com características pobres e desempenho inferior ao UNIREG.
- UNIREG e mSQL se uniram. TcX construiu seu servidor baseado na estrutura do UNIREG e utilizou utilitários do mSQL e fez API's para o novo servidor lançando em maio de 1995 o MySQL

MySQL

- Atualmente a MySQL AB(companhia dos fundadores e principais desenvolvedores do MySQL) desenvolve o programa.
- MySQL foi criado pela necessidade de um banco de dados relacional que pudesse tratar grandes quantidades de dados em máquinas de custo relativamente barato.

MySQL

- Características
 - Um dos bancos de dados relacionais mais rápidos do mercado
 - Tem linguagem simples
 - Suportado por Sistemas com filosofia UNIX, embora outros SO também forneçam suporte, como Windows por exemplo
 - Código fonte aberto
 - Alta estabilidade
 - Baixo custo

MySQL

- O Servidor MySQL foi desenvolvido originalmente para lidar com bancos de dados muito grandes de maneira muito mais rápida que as soluções existentes e tem sido usado em ambientes de produção de alta demanda por diversos anos de maneira bem sucedida. Apesar de estar em constante desenvolvimento, o Servidor MySQL oferece hoje um rico e proveitoso conjunto de funções. A conectividade, velocidade e segurança fazem com que o MySQL seja altamente adaptável para acessar bancos de dados na Internet.

Conceitos

- Banco de dados
 - Conjunto de tabelas
- Tabela
 - Conjunto de registros(Ex. Conjunto de Alunos)
- Registro
 - Conjunto de campos que representa uma entidade (Ex. Dados de um Aluno)
- Campo
 - Menor unidade de informação a ser armazenada

Conceitos

- Tabela Alunos

Campos

| RA | nome | email |
|-------|-----------|------------------------|
| 12345 | Guilherme | Gui@uol.com.br |
| 12346 | Elaine | Brito@cotil.unicamp.br |
| 12347 | Roberto | Beto@vivax.com.br |
| 12348 | José | Ze@gmail.com.br |

Registros

Exercícios de fixação

1. Que empresa desenvolveu preliminarmente o SQL?
2. O desenvolvimento inicial do SQL estava associada a que sistema de banco de dados?
3. Qual a relação existente entre SQL e SEQUEL?
4. Qual foi o objetivo do ANSI sobre os padrões do SQL, e qual o motivo que levou a esta intervenção?
5. Quais são os grupos de instruções que compõem a estrutura do SQL? Descreva-os o mais detalhadamente possível.
6. Cite os comandos do tipo DDL mencionados
7. Cite os comandos do tipo DML mencionados
8. Qual o significado de banco de dados, tabela, registro e campo?

MySQL

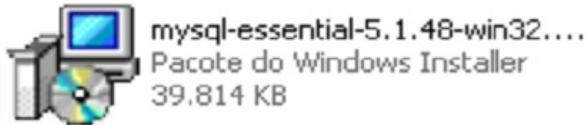
- Fazer Download
- Instalar
- Conectar via MsDOS

Download do MySQL

- Acessar mysql.org
- Escolher Download
- Vamos fazer o download da versão gratuita
 - **Download MySQL Community Server**
 - **Escolher o sistema operacional**
 - (no nosso caso Windows)

Instalação do MySql

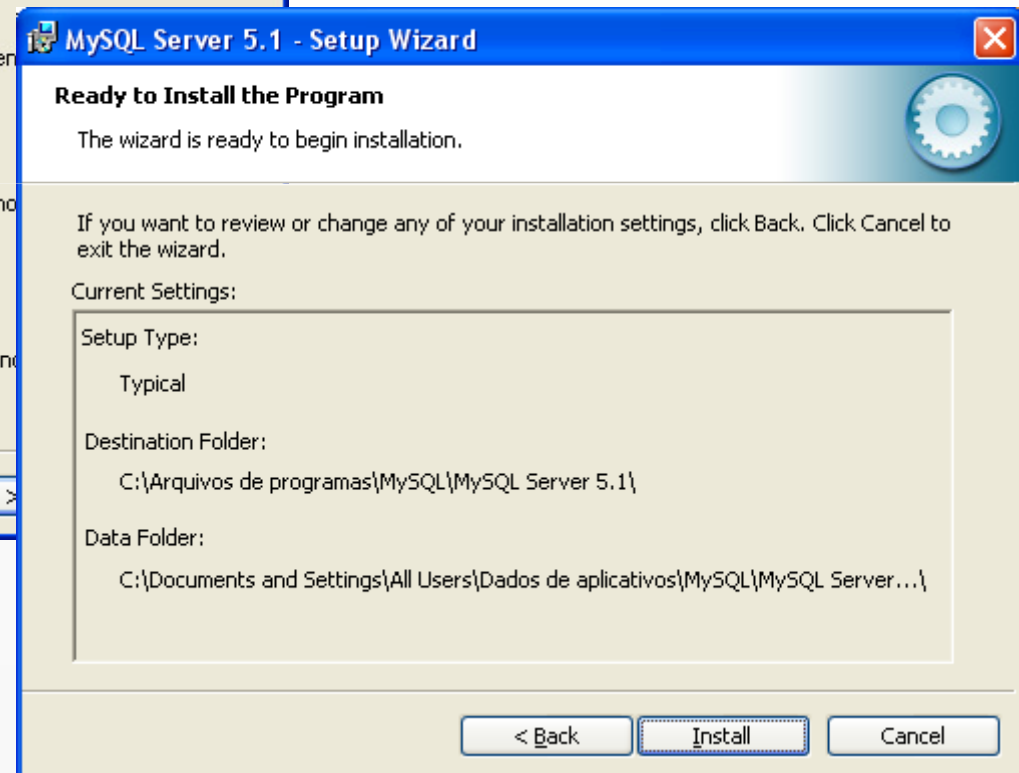
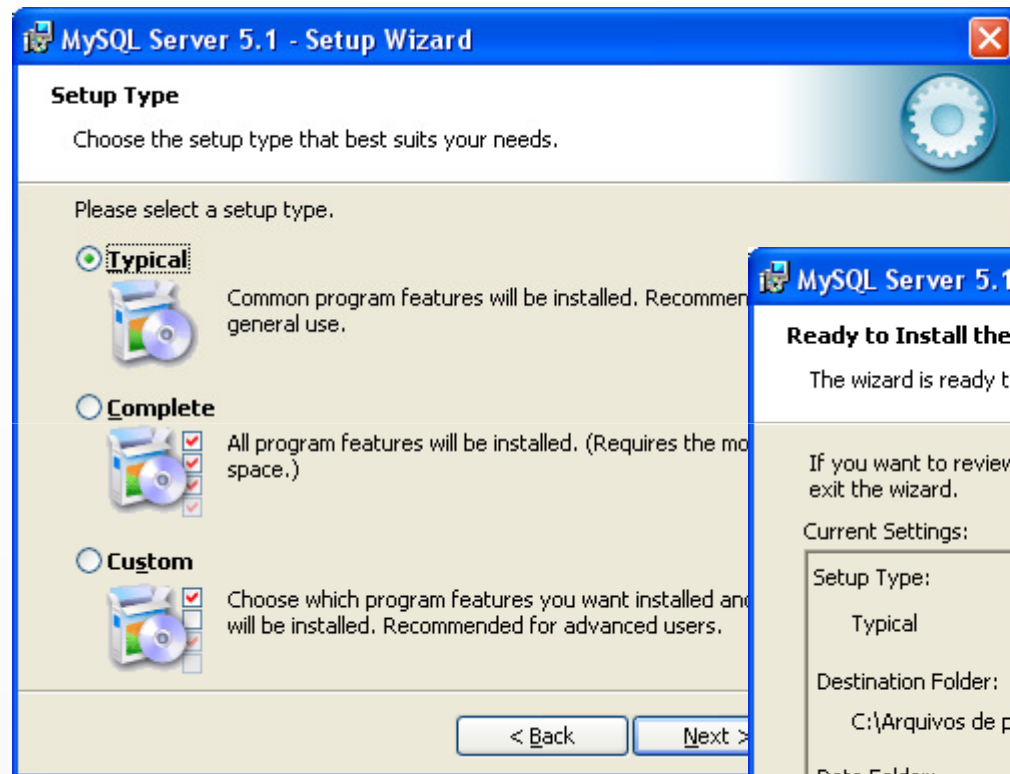
- Execute o arquivo baixado



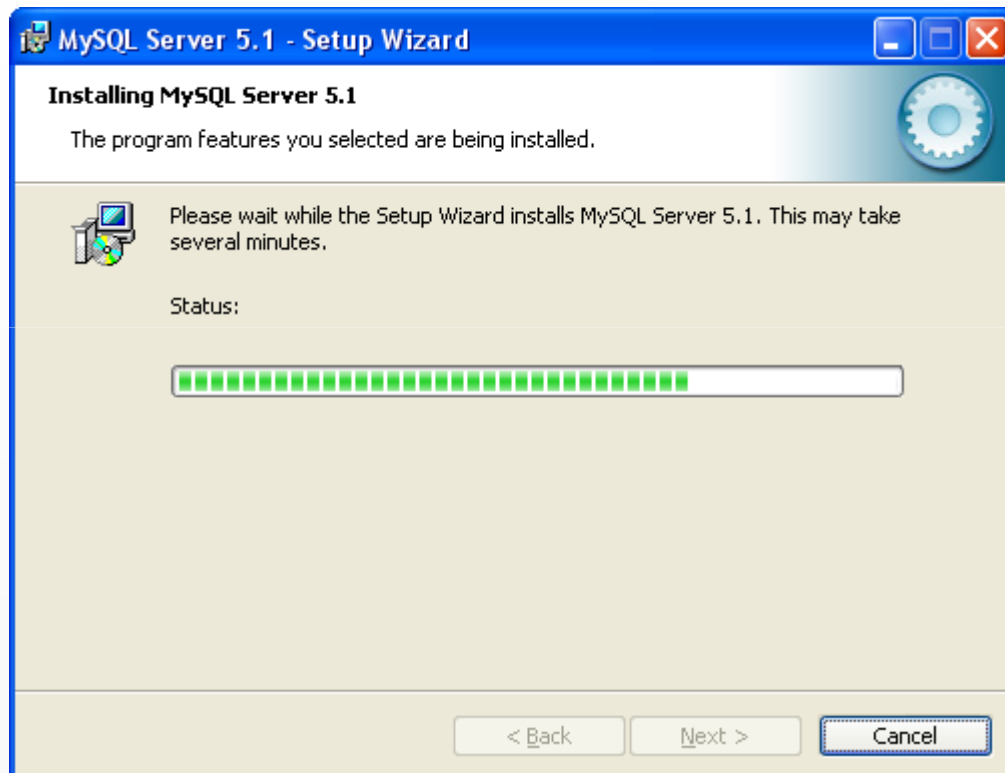
- Será iniciado o assistente de instalação



Instalação do MySql



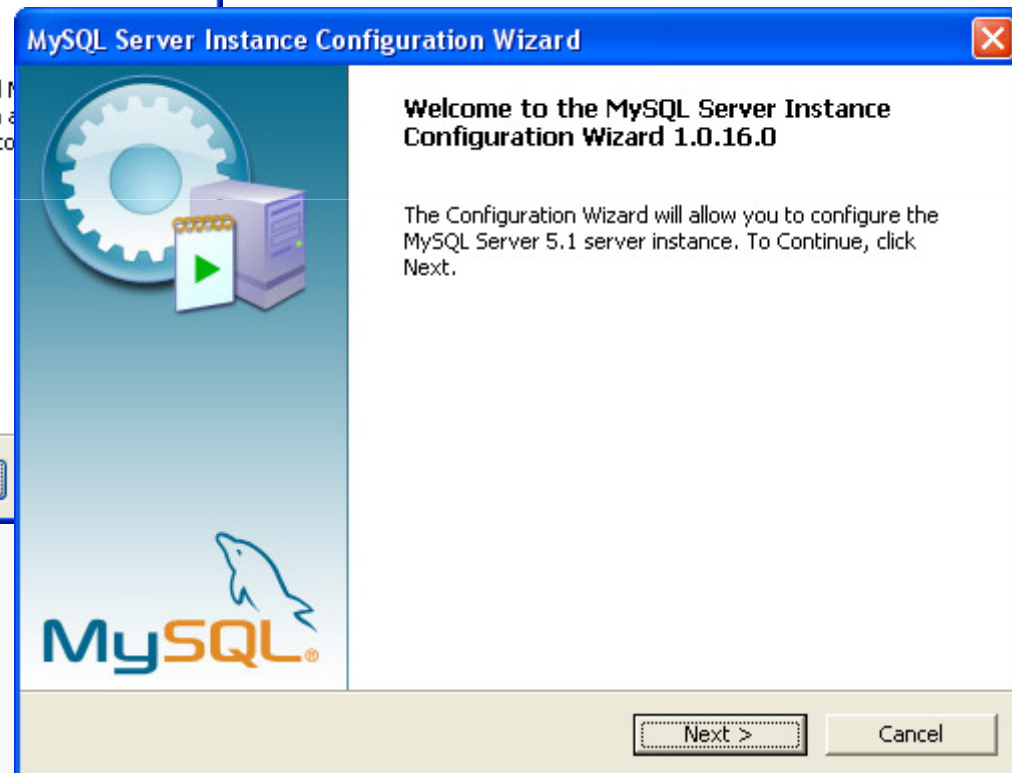
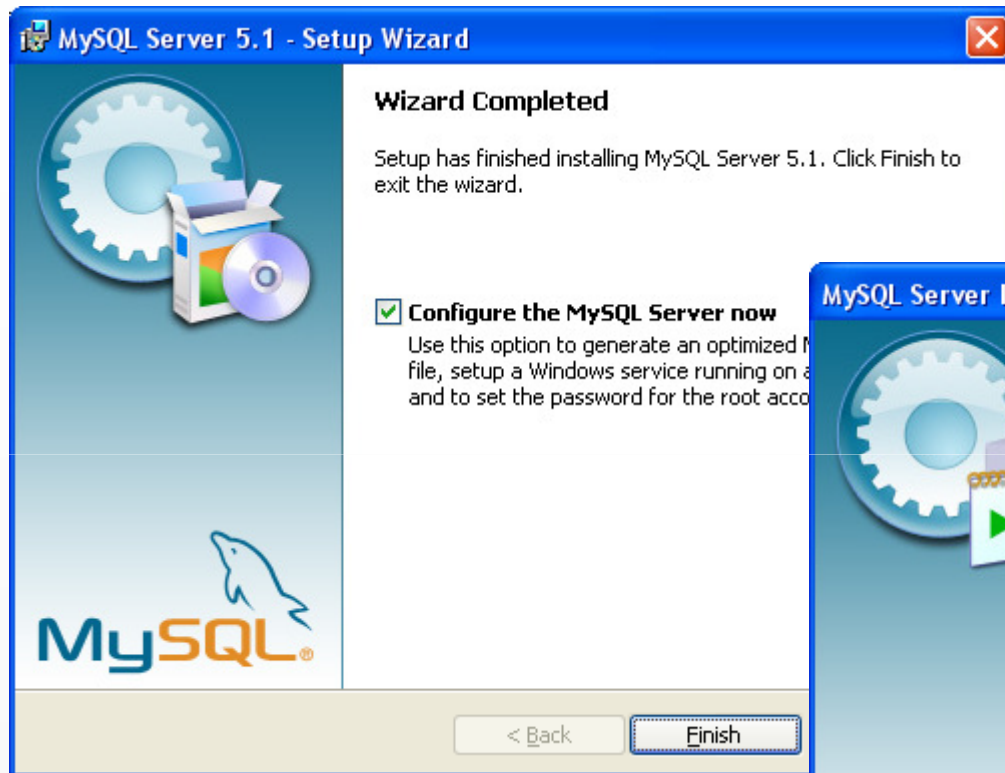
Instalação do MySql



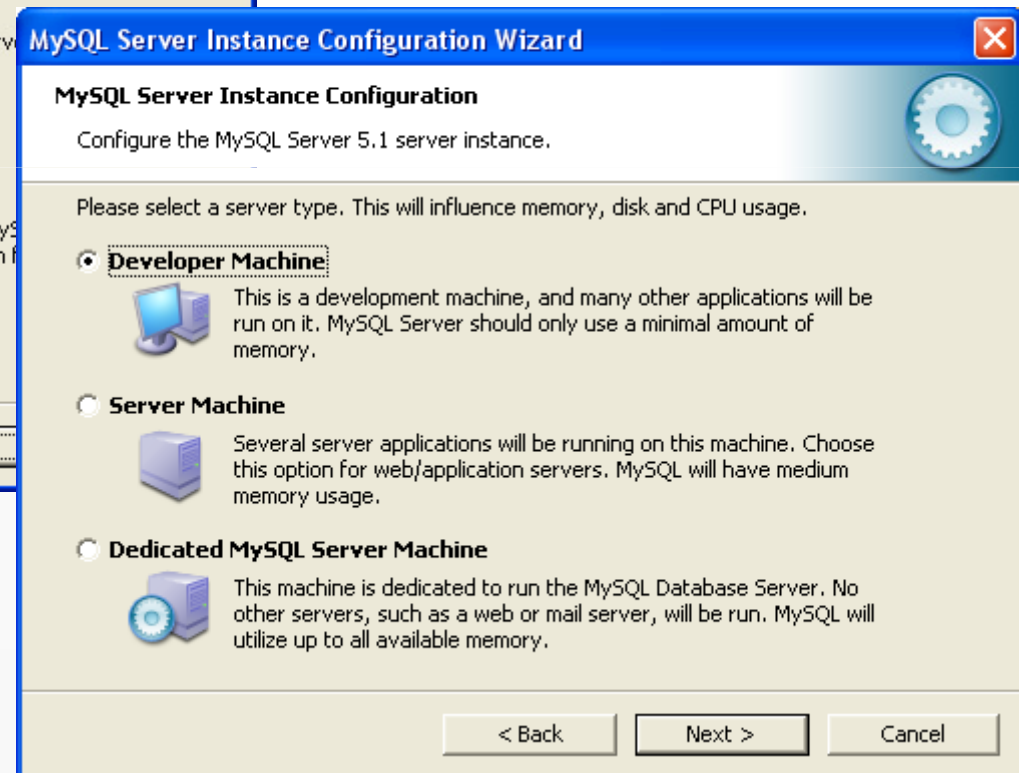
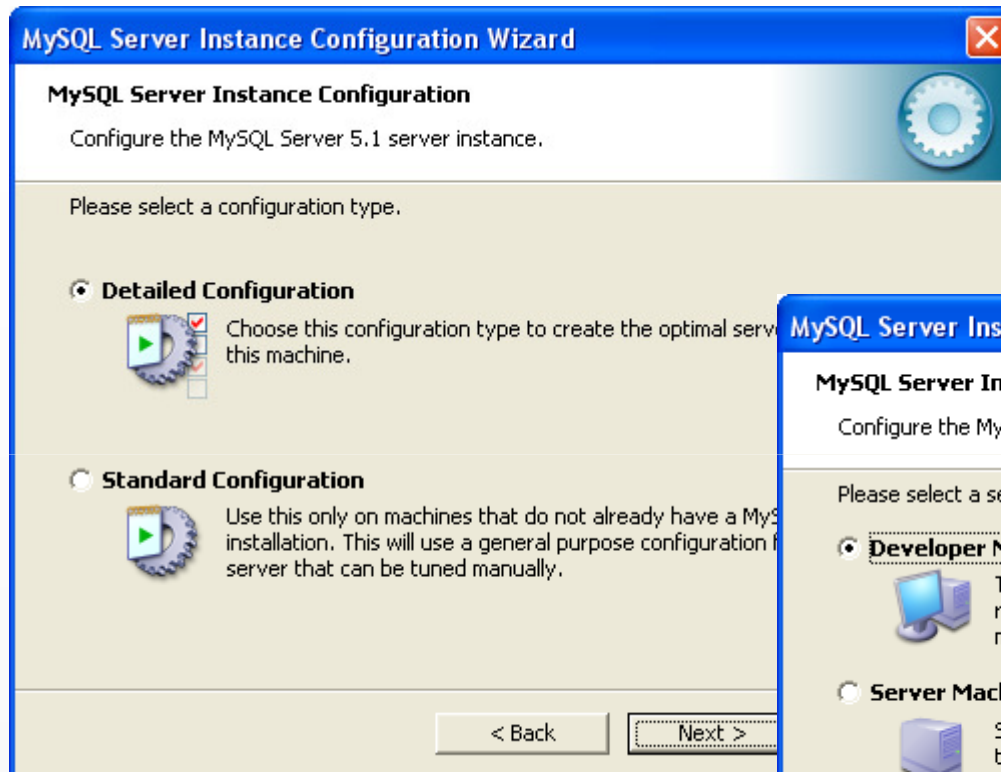
Disciplina de Banco de Dados



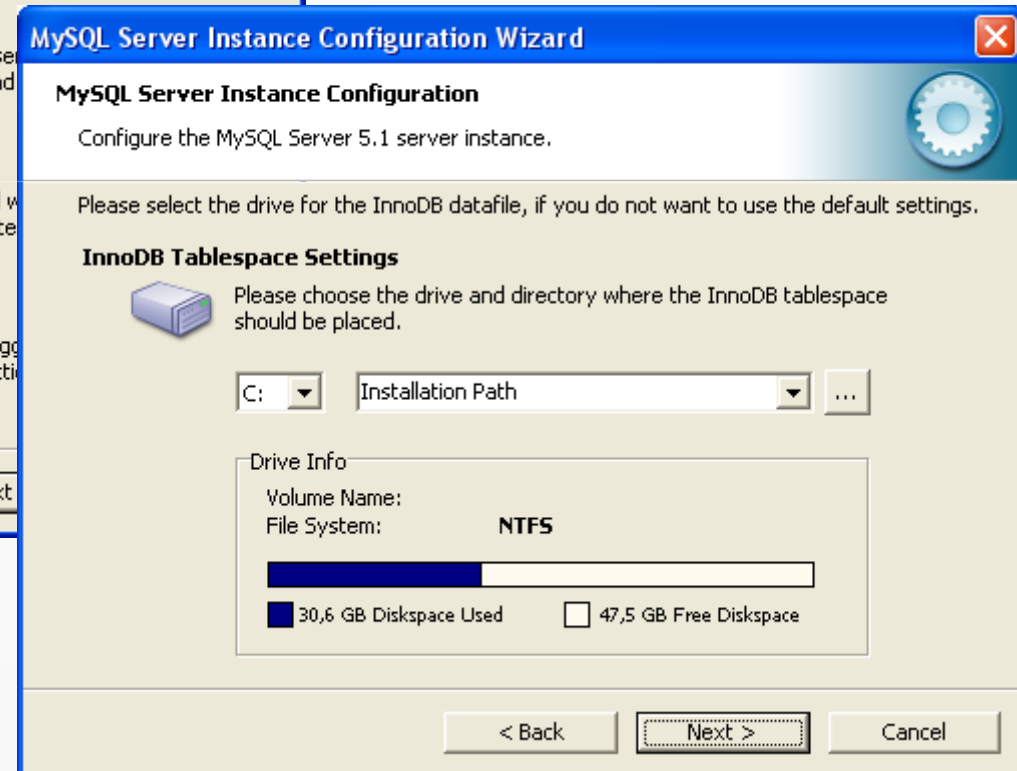
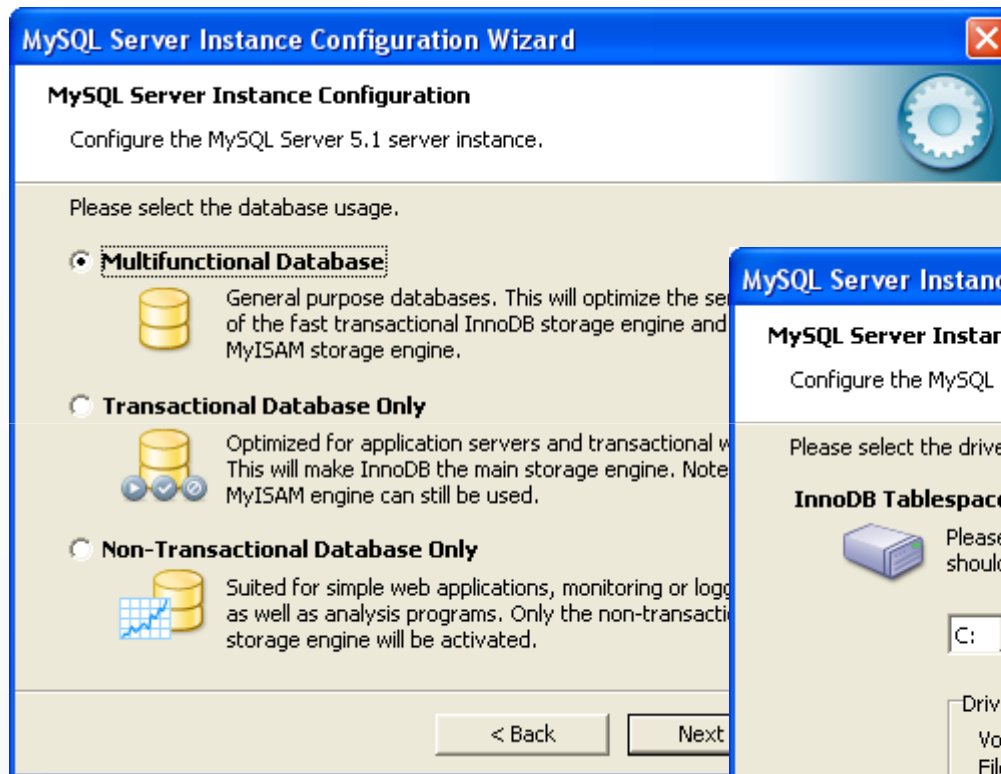
Instalação do MySql



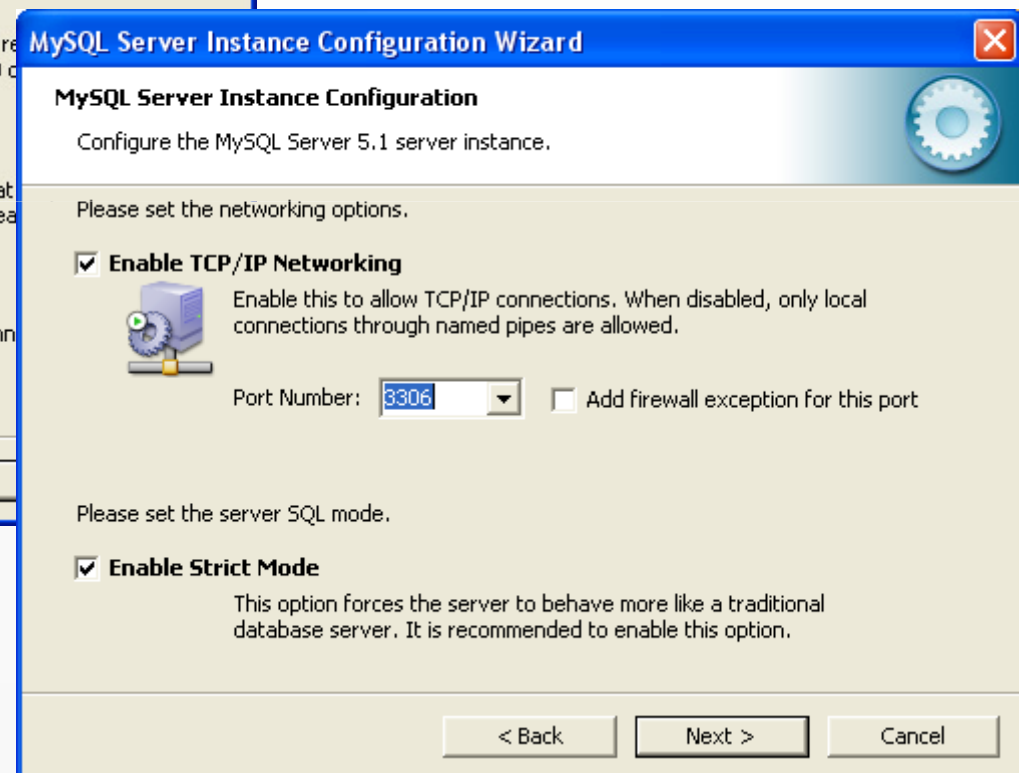
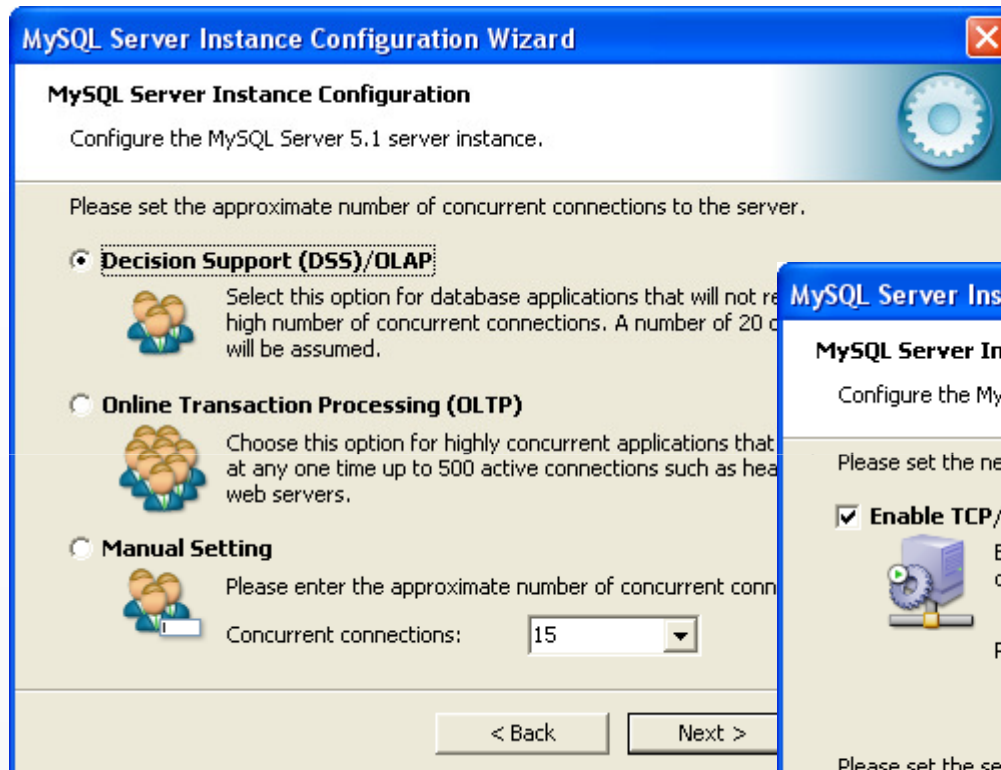
Instalação do MySql



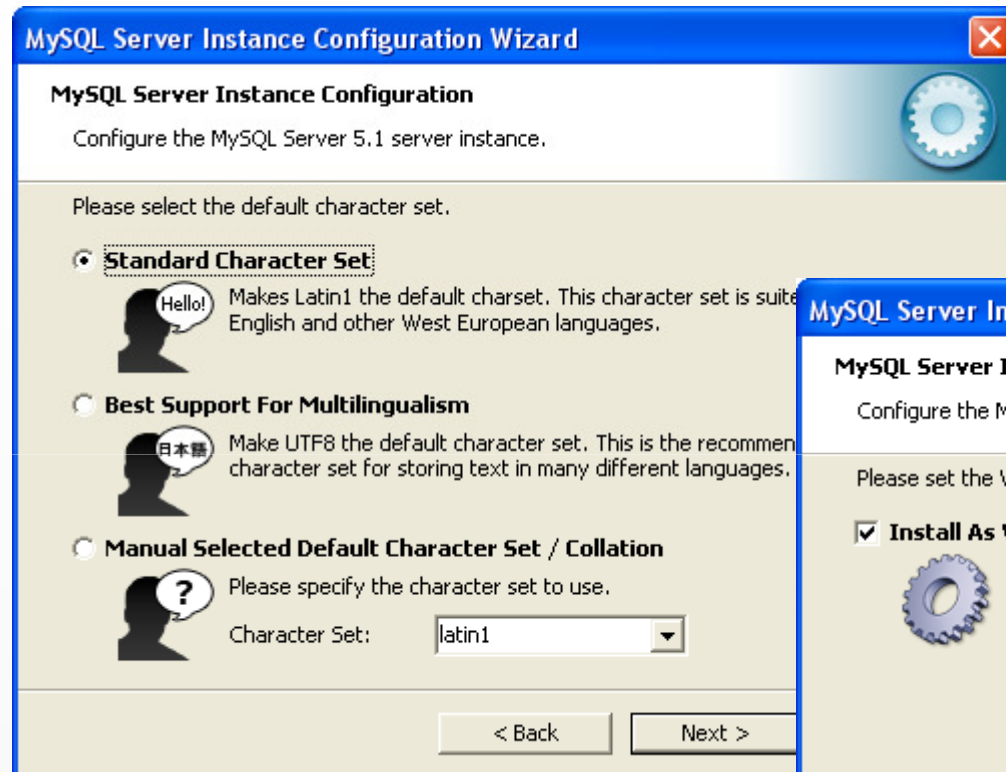
Instalação do MySql



Instalação do MySql



Instalação do MySql

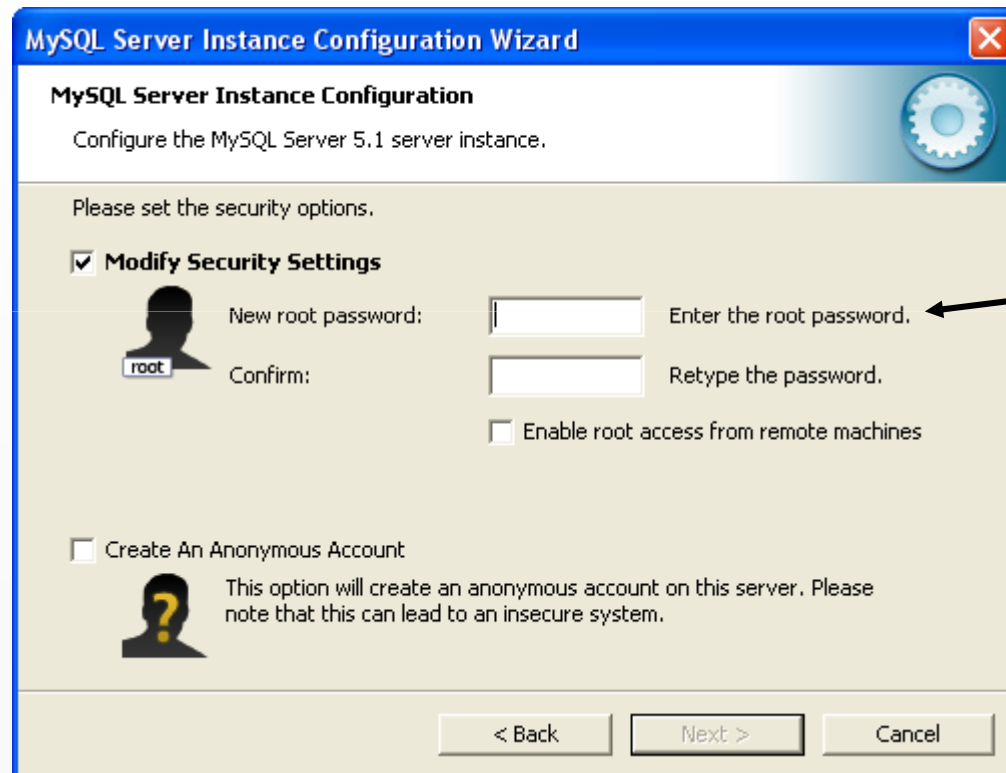


IMPORTANTE
Selecionar esta opção
para poder usar no

MsDOS

Elaine Brito

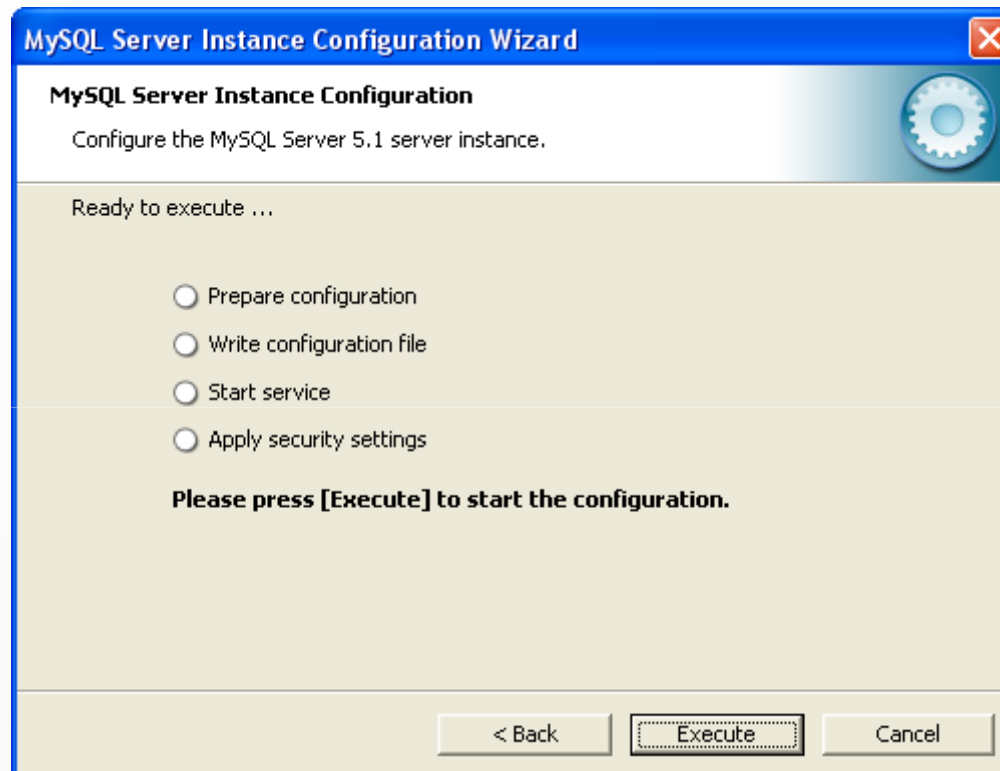
Instalação do MySql



The image shows the 'MySQL Server Instance Configuration Wizard' window. The title bar reads 'MySQL Server Instance Configuration Wizard'. The main heading is 'MySQL Server Instance Configuration' with a subtitle 'Configure the MySQL Server 5.1 server instance.' Below this, it says 'Please set the security options.' There are two main sections. The first section is 'Modify Security Settings', which is checked. It contains a 'New root password:' field with a text input box, a 'Confirm:' field with a text input box, and an unchecked checkbox for 'Enable root access from remote machines'. The second section is 'Create An Anonymous Account', which is unchecked. It features a question mark icon and a warning: 'This option will create an anonymous account on this server. Please note that this can lead to an insecure system.' At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

**Importante
Definir uma senha e
repetí-la**

Instalação do MySql



Conectar via MsDOS

- No prompt de comando vamos para a pasta MySQL

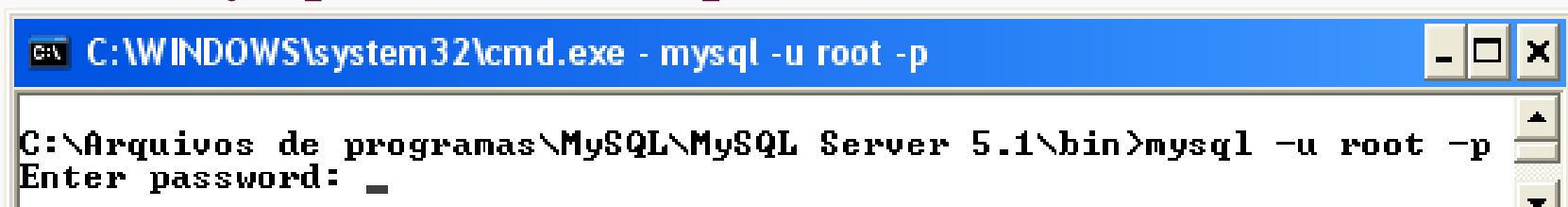
`Cd c:\arquivos de programas\mysql`

- Precisamos entrar na pasta MySQL server 5.1 e dentro dela na pasta bin

`Cd mysql server 5.1\bin`

- Devemos agora executar o mysql

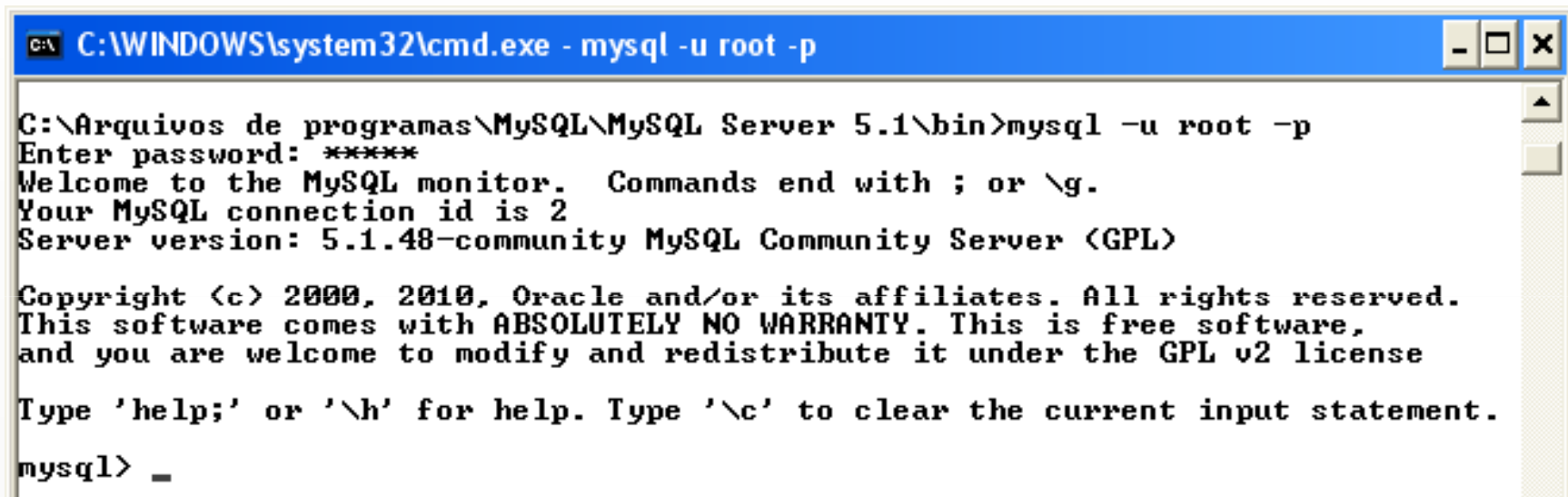
`mysql – user root -p`

A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\WINDOWS\system32\cmd.exe - mysql -u root -p". The command prompt shows the current directory as "C:\Arquivos de programas\MySQL\MySQL Server 5.1\bin" and the command "mysql -u root -p" has been entered. The prompt "Enter password:" is displayed, followed by a single underscore character "_".

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
C:\Arquivos de programas\MySQL\MySQL Server 5.1\bin>mysql -u root -p
Enter password: _
```


Conectar via MsDOS

- Digitando a senha, ele conecta...



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

C:\Arquivos de programas\MySQL\MySQL Server 5.1\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.48-community MySQL Community Server (GPL)

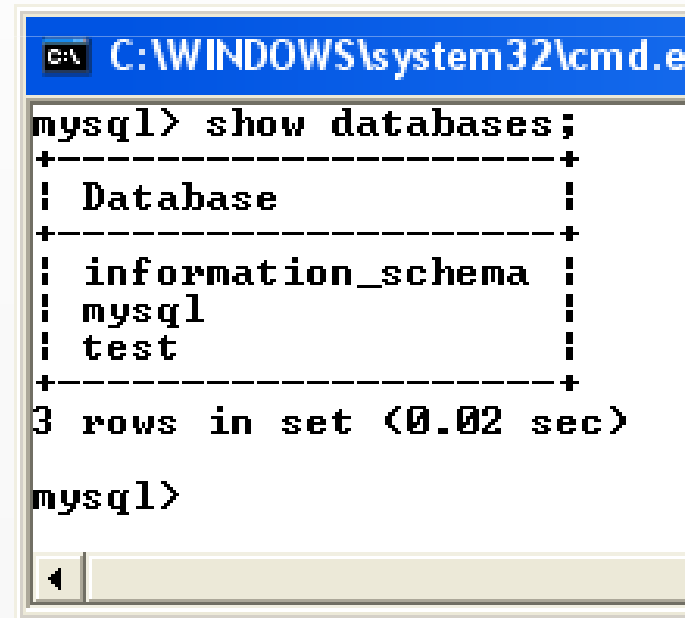
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> _
```

- Agora já está pronto para se utilizado.

Bancos de dados

- O MySQL já vem com dois bancos de dados criados.
- Para visualizá-los basta executar o comando
show databases;



```
C:\WINDOWS\system32\cmd.e
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| test      |
+-----+
3 rows in set (0.02 sec)

mysql>
```

Criação de Banco de Dados

- Quando se cria um banco de dados com o MySQL em ambiente Windows, é criada apenas uma pasta vazia, dentro da qual serão armazenados os arquivos gerados utilizando os comandos para criar tabelas.
- Como o MySQL trata o banco de dados como um esquema, temos duas opções de sintaxe para criar um banco de dados:

```
Mysql> CREATE DATABASE Exemplo;
```

ou

```
Mysql> CREATE SCHEMA Exemplo;
```

Visualizar Bancos de Dados

- `Mysql> SHOW DATABASES; [Enter]`
 - `+-----+`
 - `| Databases`
 - `+-----+`
 - `| information_schema`
 - `| mysql`
 - `| test`
 - `| Exemplo`
 - `+-----+`
- } criados na instalação do MySQL

Nomenclatura - banco de dados:

- Máximo 64 caracteres;
- Permitido: letras, números, traços, underlines;
- Proibido: barras e pontos;
- Evitar: acentos e cedilhas.

Cláusula IF NOT EXIST

- Não podemos criar bancos de dados com nomes iguais.
- Se tentarmos, será exibida a mensagem:
Can't create databases 'Exemplo'. Database exists
- Para evitar esse erro usamos a cláusula
IF NOT EXIST

```
Mysql>CREATE DATABASE IF NOT EXIST Exemplo;
```

Excluindo banco de dados

- Ao apagar um banco de dados, todas as tabelas e dados também serão excluídos.

Mysql> **DROP DATABASE** Exemplo;

Ou

Mysql> **DROP SCHEMA** Exemplo;

- Com DROP também podemos usar a clausula IF EXISTS:

Mysql> **DROP DATABASE IF EXISTS** Exemplo;

Selecionando um banco de dados

- Podemos ter vários bancos de dados, mas só podemos manipular um por vez.
- Mysql> USE Exemplo;

Dentro do Banco de dados

- Bancos de dados: pasta vazia
- Precisamos criar tabelas dentro dele
- Para criar Tabelas precisamos definir sua estrutura = seus campos
- Exemplo: Tabela: Aluno
Campos: RA, nome, email
- Também precisamos definir tipos dos Campos

Tipos de Dados

- No MySQL os tipos de dados são divididos em três grupos:
 - Tipos Numéricos
 - Tipos de Data
 - Tipo de Cadeia

Tipos Numéricos

- Exatos:
 - NUMERIC
 - DECIMAL
 - INTEGER
 - SMALLINT, entre outros
- Aproximados:
 - FLOAT
 - REAL ou DOUBLE
 - DECIMAL ou DEC.

Tipos Numéricos

| Tipo | Bytes | De | Até |
|-----------------------|-------|-----------------------------|---------------------|
| TINYINT | 1 | -128 | 127 |
| SMALLINT | 2 | -32768 | 32767 |
| MEDIUMINT | 3 | -8388608 | 8388607 |
| INT ou Integer | 4 | -2147483648 | 2147483647 |
| BIGINT | 8 | -9223372036854775808 | 9223372036854775807 |
| Bit ou Bool | 1 | Inteiro que pode ser 0 ou 1 | |

Tipos Numéricos

- NUMERIC e DECIMAL
 - implementados como o mesmo tipo.
 - preserva a exatidão (Ex. dados monetários).
- Exemplo:

Precisão: número de dígitos
que serão armazenados

Escala: número de dígitos
que serão armazenados
após o ponto decimal

– salario DECIMAL(5,2)

Tipos Numéricos

- **FLOAT** - números aproximados
 - Números pequenos
 - permite uma especificação opcional da precisão
 - quando não especifica precisão utiliza quatro bytes
- **REAL e DOUBLE**
 - não aceitam especificações de precisão.
 - implementados como valores de ponto flutuante de 8 bytes de dupla precisão

Tipos de Data e Hora

- DATE
 - para apenas do valor da data, sem a parte da hora.
 - formato 'YYYY-MM-DD'.
 - faixa de '1001-01-01' até '9999-12-31'.
- DATETIME
 - para valores que contém data e a hora.
 - formato 'YYYY-MM-DD HH:MM:SS'.
 - faixa de '1001-01-01 00:00:00' até '9999-12-31 23:59:59'.

Tipos de Data e Hora

- **TIME**
 - formato 'HH:MM:SS'
 - faixa '-838:59:59' até '838:59:59'.
 - Hora é grande pois pode ser usado para intervalos de tempo entre dois eventos. Por exemplo e não só para hora do dia que seria até 24

Tipos de Data e Hora

- **TIMESTAMP**
 - para valores que contém data e a hora.
 - Margem vai desde 1 de janeiro de 1970 ao ano 2037
- **YEAR**
 - formato com 2 ou 4 algarismos.
 - faixa de 1901 até 2155.

Tipos de Cadeia

- CHAR(N)
 - caracteres alfanuméricos
 - tamanho é fixo e instaurado ao ser criado.
 - pode ter de 0 a 255 caracteres.
 - Exemplo: endereço CHAR(30);
 - Observe que não há acentos nem cedilhas no nome do campo, pois muitos servidores não acentuam, e sua tabela teria difícil acesso.

Tipos de Cadeia

- **VARCHAR(N)**
 - aloca apenas o espaço necessário para gravação
 - CHAR tamanho definido fixo (mesmo que não usado, aquele espaço em disco é alocado)
 - trocamos espaço por velocidade, pois este campo é 50% mais lento que o anterior.
 - Exemplo: endereço VARCHAR(30);
 - Define um campo chamado endereço que pode conter até 30 letras. Se você preencher apenas duas, o campo não ocupará todos os 30 bytes, mas apenas 2.

Tipos de Cadeia

- TEXT/BLOB
 - para guardar grandes quantidades de caracteres.
 - pode conter de 0 a 65535 bytes,
 - TEXT não é sensível a letras maiúsculas e minúscula quando uma comparação é realizada, e o BLOB sim.

Tipos de Cadeia

- MediumTEXT/MediumBLOB
 - Máximo de 16.777.215 caracteres.
- LongTEXT/LongBLOB
 - Máximo de 4.294.967.295 caracteres.

Tipos de Cadeia

- SET

- permite que o usuário faça uma escolha dado determinado número de opções.
- cada campo pode conter até, 64 opções.
- Exemplo:
 - curso SET("informatica", "geomatica") NOT NULL;
 - Neste exemplo este campo pode conter apenas os seguintes itens: " "
 - "informatica"
 - "geomatica"
 - "informatica,geomatica"

Tipos de Cadeia

- ENUM
 - semelhante ao SET, com a diferença que apenas um valor pode ser escolhido.
 - Exemplo:
 - sexo ENUM("masculino", "feminino") NOT NULL;
 - Neste exemplo este campo pode conter os seguintes valores: “ “
 - “masculino“
 - “feminino“

Tabelas

- SQL oferece três instruções para definição do esquema da base de dados:
- **Create Table**
 - define a estrutura da tabela
 - cria a tabela vazia
- **Drop Table**
 - Elimina a tabela da base de dados
- **Alter Table**
 - Permite modificar a definição da tabela

Criação de Tabela

```
mysql> CREATE TABLE teste(  
    > codigo INT,  
    > nome CHAR(15),  
    > email CHAR(25));
```

Nome da tabela

Tipos dos campos

Nome dos campos

Restrições

- Uma maneira de limitar os dados que podem ser inseridos em uma tabela é a definição de tipo.
- Podemos desejar definir outras restrições como:
 - Dados de um coluna em relação a outras colunas ou linhas.
 - Valores aceitáveis.

Restrições

- **Não Nulo:**
 - Define que uma coluna não pode conter valor nulo.
 - A cláusula NOT NULL especifica que uma coluna não admite valor vazio.
 - Exemplo: `nome CHAR(15) NOT NULL;`
 - o campo nome não pode ser vazio, é de preenchimento obrigatório.

Restrições

- **Unicidade:**

- Define que os dados contidos na coluna (ou grupo de colunas) sejam únicos (não se repitam) em relação a todas as outras linhas da tabela.

- A cláusula **UNIQUE** especifica que os dados não se repetem.

Restrições

```
CREATE TABLE alunos (  
    ra      char(5) UNIQUE,  
    nome char(50),  
    nasc  date );
```

Ou

```
CREATE TABLE alunos (  
    ra      char(5),  
    nome char(50),  
    nasc  date,  
    UNIQUE (ra) );
```

Restrições

Se uma restrição de unicidade faz referencia a um grupo de colunas, elas são separadas por vírgula:

```
CREATE TABLE exemplo (  
  a    char(5),  
  b    char(50),  
  c    date,  
  UNIQUE (a,c)    );
```

Restrições

- **Chave primária:**
- A chave primária indica que a coluna, ou grupo de colunas, pode ser utilizado como identificador único para as linhas da tabela
- É a junção de restrição de unicidade com a restrição de não nulo. Apenas a unicidade não garante identificador único pois não exclui os valores nulos.
- Uma tabela pode ter no máximo uma chave primária, mas pode ter várias restrições de unicidade e de não nulo.

Restrições

```
CREATE TABLE alunos (  
    ra      char(5) UNIQUE NOT NULL,  
    nome char(50),  
    nasc  date    );
```

ou

```
CREATE TABLE alunos (  
    ra char(5) PRIMARY KEY,  
    nome char(50),  
    nasc date);
```

A chave primária pode ser composta de vários atributos:

```
CREATE TABLE exemplo (  
    a      char(5),  
    b      char(50),  
    c      date,  
    PRIMARY KEY (a,c));
```


Restrições

- **Auto incremento:**
 - Este recurso, faz com que conforme novos registros são criados, automaticamente estes obtém valores que correspondem ao valor deste mesmo campo no registro anterior, somado a 1.
 - Exemplo: código `INT AUTO_INCREMENT`;
 - Soma um a cada registro automaticamente neste campo. Começando de 1, com inserção subsequente.

Praticando...

- Antes de criar uma tabela, precisamos selecionar o banco de dados dentro do qual ela será criada, para isso utilize o comando `USE nome do banco;`
- Criar tabela:

```
CREATE TABLE teste( codigo INTEGER  
  AUTO_INCREMENT NOT NULL,  
  nome CHAR(15) NOT NULL,  
  email CHAR(30),  
  telefone CHAR(8),  
  PRIMARY KEY(codigo));
```

Comandos relativos as tabelas:

- **Mostrar tabelas** - Lista todas as tabelas existentes no banco de dados atual.

```
mysql>show tables;
```

- **Mostrar colunas** - Mostra as colunas da tabela.

```
mysql>show columns from teste;
```

- **Mostrar estrutura** - Mostra a estrutura da tabela.

```
mysql>describe teste;
```

Alteração de Tabelas

- Quando notamos que as necessidades da aplicação mudaram ou que foi cometido um erro, podemos modificar a estrutura das tabelas já criadas.
- Podemos incluir ou excluir colunas, restrições, modificar nome de coluna ou da própria tabela.
- Tudo isso pode ser feito através do comando **ALTER TABLE**

Modificação da estrutura de uma tabela

- ADD <campo> <tipo>
 - Insere novo campo
- DROP <campo>
 - Remove determinado campo
- MODIFY<campo><tipo>
 - Modifica o tipo de determinado campo

Alter Table - ADD

- Inserir na tabela teste o campo nascimento que conterá a data de nascimento dos cadastrados.

```
mysql>alter table teste add nascimento date;
```

A nova coluna não pode possuir a restrição de não-nulo, porque a coluna inicialmente deve conter valores nulos. Porém, a restrição de não-nulo pode ser adicionada posteriormente.

Observe as alterações com o describe teste;

Alter Table - ADD

- Inserir na tabela teste o campo endereço após o campo nome.

```
mysql>alter table teste add endereco char(50)  
after nome;
```

Observe as alterações com o describe teste;

Obs. Para inserir antes de todos os outros campos use
first

Alter Table - MODIFY

- O campo email foi criado com limite de 30 caracteres. Observe isso com o comando describe teste;
- Trocar para 40 caracteres .
- **mysql>alter table teste modify email CHAR(40);**
- Execute o describe teste; novamente para observar a alteração.

Alter Table - CHANGE

- Trocar no nome da coluna email por e_mail .
- **mysql>alter table teste change email e_mail char(30);**
- Execute o describe teste; para observar a alteração.

Alter Table - Drop

- Abaixo vemos como excluir o campo codigo da tabela Teste:

```
mysql>alter table teste drop codigo;
```

Observe as alterações com o describe teste;

Alter Table – ADD Primary Key

- Abaixo vemos como definir o campo nome como chave para a tabela Teste:

```
mysql>alter table teste add primary key  
(nome,nascimento);
```

Observe as alterações com o describe teste;

Alter Table - Drop Primary Key

- Exclui a chave primária, mas não a coluna

```
mysql>alter table teste drop primary key;
```

Observe as alterações com o describe teste;

Incluir Restrição

- Exemplos:

```
ALTER TABLE exemplo ADD  
CONSTRAINT algum nome UNIQUE  
(codigo);
```

Excluir Restrição

- Para excluir uma restrição é necessário conhecer seu nome (que pode ter sido dado pelo usuário ou atribuído pelo sistema).
 - ALTER TABLE exemplo DROP CONSTRAINT nome_restricção;

Renomeando a tabela

- Para alterar o nome da tabela A para B

```
mysql> ALTER TABLE A RENAME TO B;
```

Drop Table

- Exclui a tabela. Todos os dados e definições da tabela são removidos, assim tenha cuidado com este comando!

Observação: não vamos executar este comando pois vamos continuar usando a tabela teste em aula apenas observe o comando.

```
mysql>drop table teste;
```


Praticando...

- Faça as alterações necessárias na tabela para que ela tenha as características iniciais;
- Tabela teste
 - código inteiro, auto-incremento, chave primária
 - nome CHAR(15)
 - email CHAR(30)
 - telefone CHAR(8)

Manipulando a base de dados

- Uma base de dados pode ser manipulada com quatro operações básicas:
 - Incluir,
 - Apagar,
 - Alterar e
 - Pesquisar.
- Vale lembrar que como toda linguagem para computadores, o MySQL tem suas regras. Um erro de parênteses que seja pode resultar no inverso do que você espera. Portanto, fique atento a sintaxe de seus comandos.

Manipulando a base de dados

- **Inserindo registros**

- Para se adicionar dados a uma tabela, usamos o comando INSERT, que diz por si só sua função, como o exemplo que segue:
- `mysql>INSERT INTO teste VALUES (NULL, 'Elaine', 'elaine.brito@cotil.unicamp.br', '34444444');`

Observações:

- Todos os campos que contêm texto, ou seja, CHAR, VARCHAR, BLOB, TEXT, etc. têm de ficar entre apóstrofos
- Para campos do tipo número, não se usam apóstrofos.
- A entrada NULL em um campo do tipo auto-incremento, permite que o MySQL providencie o conteúdo deste campo de forma automática. No caso do primeiro campo, o valor será 1, no segundo 2, no terceiro 3 e assim consecutivamente.
- Se possuíssemos um campo DATE, a entrada NULL faria com que o valor gravado no registro se torne a data atual.

Observações:

- É importante lembrar-se sempre de passar para o comando INSERT um número de parâmetros igual ao número de campos na tabela que está recebendo os dados. Caso contrario, você obterá uma mensagem de erro.

Pesquisando registros

- As pesquisas no MySQL são feitas através do comando SELECT.

- Exemplo:

```
mysql>SELECT * FROM teste;
```

- Resultado:
 - Lista todos os campos(*) de todos os registros da tabela teste.

Pesquisando registros

- Se queremos ver apenas alguns campos da tabela, especificamos os nomes das colunas desejadas, separadas por virgulas.

- Exemplo:

```
mysql>SELECT codigo,nome FROM teste;
```

- Resultado:
 - Lista os campos codigo e nome de todos registros da tabela teste.

Pesquisando registros

- Ação:

```
mysql>SELECT * FROM teste WHERE  
      (nome = 'Elaine');
```

- Resultado:
 - Lista todos os registros da tabela teste que possui 'Elaine' no campo nome.

Alterando registros

- Ação:

```
mysql>UPDATE teste SET nome = 'Elaine Brito' WHERE nome = 'Elaine';
```

- Resultado: Procura na tabela um registro que contenha no campo nome o conteúdo 'Elaine', definido pelo comando WHERE. Encontrado o registro, ele é substituído pelo nome definido no comando SET, que é 'Elaine Brito'.

Apagando registros

- Ação:
 - mysql>DELETE FROM
teste WHERE (telefone =
'34444444');
- Resultado: Apaga da tabela teste todos os registros que têm o conteúdo '34444444' no campo telefone.

Operadores Aritméticos:

- São responsáveis pela execução de operações matemáticas simples:

| | |
|---|---------------|
| + | Adição |
| - | Subtração |
| * | Multiplicação |
| / | Divisão |

Operadores Relacionais:

- São utilizados quando precisamos fazer comparações entre dois valores:

| | |
|----|------------------|
| > | Maior que |
| < | Menor que |
| = | Igual a |
| <> | Diferente de |
| >= | Maior ou igual a |
| <= | Menor ou igual a |

Operadores lógicos:

- AND (&&)
 - O operador lógico AND, ou E, deve ser usado em uma pesquisa que se deseja entrar dois valores.
 - O AND, verifica ambas as cláusulas da comparação, e só retorna algum valor se as duas tiverem uma resposta verdadeira.
 - Observe o exemplo:

```
mysql>SELECT * FROM teste WHERE (nome =  
        'ELAINE') AND (telefone = '34444444');
```

Esta pesquisa mostrara todos os registros que contém no campo nome o conteúdo 'Elaine', E (AND) no campo telefone, o conteúdo '34444444'.

Operadores lógicos:

- OR (||)
 - O operador lógico OR, ou OU, deve ser usado em uma pesquisa que se deseja entrar dois valores.
 - O OR, verifica ambas as cláusulas da comparação, e retorna valores se qualquer um dos membros obtiver resultado.

```
mysql>SELECT * FROM teste WHERE (nome  
= 'Elaine') OR (telefone = '34444444');
```

Esta pesquisa fará com que todos os resultados que contenham o conteúdo 'Elaine' no campo nome, OU telefone '34444444' sejam exibidos na tela.

Operadores lógicos:

- NOT (!)
 - O operador lógico NOT, ou NÃO, realiza uma pesquisa, excluindo valores determinados do resultado.
- `mysql>SELECT * FROM teste WHERE`
- `(nome != 'Elaine');`
- Esta pesquisa listará todos os registros da base de dados teste, NÃO (NOT) mostrando aqueles que possuem 'Elaine' como conteúdo do campo nome.

Operadores lógicos:

- ORDER BY
 - O operador lógico ORDER BY, ou ORDENAR POR, simplesmente lista os registros, colocando-os em ordem de acordo com o campo solicitado.
- `mysql>SELECT * FROM teste WHERE`
- `(nome = 'Elaine') ORDER BY telefone;`
- O resultado desta busca resultara em todos os registros contendo 'Elaine' no campo nome, e a listagem será organizada de acordo com a ordem do telefone.

ORDER BY

- ASC e DESC especificam o tipo de classificação e são, respectivamente, abreviações das palavras em ingles ascending e descending, ou seja, classificação crescente ou decrescente.
- Quando não especificamos nenhum , o padrão é ascendente

Exercícios

- Crie uma tabela Empregados como a seguir:

| Campo | Tipo | Descrição |
|----------------|---------------|-----------------------------------|
| codigo | Integer | Código do funcionário(não nulo) |
| nome | Char(40) | Nome do funcionário (não nulo) |
| setor | Char(2) | Setor onde o funcionário trabalha |
| cargo | Char(20) | cargo do funcionário |
| salario | Decimal(10,2) | salário do funcionário |
| Chave Primária | | Será o campo codigo |

Inserção de registros

| codigo | nome | setor | cargo | salario |
|-----------|-----------------|-------|--------------|---------|
| 1 | Cleide Campos | 1 | Secretária | 1000 |
| 3 | Andreia Batista | 6 | Programadora | 1500 |
| 4 | Cristiano Souza | 6 | Programador | 1500 |
| 6 | Mario Souza | 4 | Analista | 2200 |
| 7 | Ana Silva | 4 | Secretária | 1000 |
| 9 | Silvia Soares | 5 | Supervisora | 1650 |
| 10 | José da Silva | 1 | Programador | 1500 |
| 15 | Manoel Batista | 1 | Projetista | 2500 |
| 25 | João Silva | 4 | Supervisor | 1650 |

Exemplo:

Insert into empregados values (1,'Cleide Campos','1','secretaria',1000);

Listagem de registros

- Apresentar a listagem completa dos registros da tabela Empregados;
- Apresentar uma listagem dos nomes e dos cargos de todos os registros da tabela Empregados;
- Apresentar uma listagem dos nomes dos empregados do setor 1
- Listagem dos nomes e dos salários por ordem de nome (a-z)
- Listagem dos nomes e dos salários por ordem de nome em formato descendente (z-a)
- Listagem dos setores e nomes colocados por ordem do campo setor em formato ascendente e do campo nome em formato descendente.
- Listagem de nomes ordenados pelo campo nome em formato ascendente, dos empregados do setor 4.

Alteração de Registros

- O empregado de código 7 teve um aumento de salário para 2500.50.
- Andreia Batista foi transferida do departamento 5 para o departamento 3.
- Todos os empregados da empresa tiveram um aumento de salário de 20%.
- Todos os empregados do setor 1 foram demitidos , exclua-os.
- Mario Souza pediu demissão, exclua-o.

Modificação da estrutura de uma tabela

- Inserir na tabela Empregados o campo admissao que conterà a data de admissão dos empregados.
- Em seguida será necessário atualizar a tabela com as datas de admissão dos empregados ativos.

| codigo | nome | seto | cargo | salario | admissao |
|--------|-----------------|------|--------------|---------|------------|
| 3 | Andreia Batista | 3 | Programadora | 1800 | 2000-10-20 |
| 4 | Cristiano Souza | 6 | Programador | 1800 | 1999-09-10 |
| 7 | Ana Silva | 4 | secret ria | 3000.6 | 2005-05-15 |
| 9 | Sylvia Soares | 5 | Supervisora | 1980 | 2008-04-25 |
| 25 | João Silva | 4 | Supervisor | 1980 | 2000-11-15 |

Praticando

- Apresente a listagem dos empregados que foram admitidos em 20/10/2000
- Apresente a listagem dos funcionários que foram admitidos após 01/01/2000
- O departamento 2 foi reaberto e admitiu-se os seguintes empregados:

| codigo | nome | setor | cargo | salário | admissão |
|--------|---------------|-------|-------------|---------|------------|
| 20 | Aline Brito | 2 | Supervisora | 1700.00 | 2009-09-05 |
| 22 | Silvia Mendes | 2 | Gerente | 2000.00 | 2009-09-01 |
| 24 | Moacir Campos | 2 | Programador | 2000.00 | 2009-09-10 |
| 26 | Marco Silva | 2 | Programador | 2000.00 | 2009-09-15 |

Exercícios com Operadores

- Apresentar nome e salário dos empregados que ganham acima de 1700.00(valor do salário) mais uma comissão de 50.00 (totalizando 1750.00).
- Listar os empregados do setor 5.
- Listar os empregados cujo cargo é programador.
- Listar empregados com salário até 2000.00

Exercícios - Operadores Lógicos

- Listar programadores do setor 2.
- Listar empregados que sejam supervisor ou supervisora.
- Listar empregados que não sejam gerentes.

Operadores auxiliares

- **Between**

- Definição de intervalos de valores para a cláusula where.

<expressão> [Not] BETWEEN <mínimo> and <máximo>

- **IN**

- Algumas vezes não é possível definir um intervalo sequencial de valores.

<expressão> [Not] IN <valor1,valor2,...,valorN>

Exercícios

- Listar empregados com salário entre 1700.00 e 2000.00
- Listar programadores e programadoras

Verificação de caracteres

- Para verificar sequência de caracteres dentro de um campo do tipo string (char ou varchar), pode-se utilizar junto com a cláusula where uma condição baseada no uso do operador LIKE.

<expressão> [NOT] LIKE <valor>

- Exemplos:
- ‘A%’ – começa com letra A
- ‘_A%’ – segunda letra do nome A
- ‘%AN%’ - possui AN em qualquer posição

Exercícios

- Listar empregados cujo nome comece com a letra A
- Listar empregados cujo nome tem a segunda letra A
- Listar empregados que tem a sequência AN em qualquer posição do nome.

Se vazio

- Uma ocorrência bastante útil é verificar a existência de campos que possuam valores em branco ou não. Para isso usa-se junto ao where o operador IS NULL.

<expressão> IS [NOT] NULL

Exemplo:

Select * from Empregados where nome is null

Funções Agregadas

- AVG() – média aritmética
- MAX() – Maior valor
- MIN() – Menor valor
- SUM() – Soma dos valores
- COUNT() – Número de valores
 - ALL- contagem dos valores não vazios
 - Distinct – contagem dos valores não vazios e únicos

Função([all]<expressão>/[distinct]<expressão>)

Exercícios

- Média aritmética dos salários de todos os empregados
- Média aritmética dos salários de todos os empregados do setor 3
- Soma dos salários de todos os empregados
- Soma dos salários de todos os empregados do setor 5
- Maior salário existente entre todos os empregados
- Menor salário existente entre todos os empregados
- Numero de empregados do setor 3
- Número de empregados que ganham mais que 2000.00
- Número de setores existentes no cadastro de empregados.

Múltiplas Tabelas

- Uma das grandes características de um sistema de banco de dados relacional é a capacidade de interagir com múltiplas tabelas, como se elas fossem apenas uma.
- Para exemplificar esse tipo de operação vamos criar duas tabelas com um campo comum

Tabela cliente

```
Create table cliente (  
  codigo char(3) primary key,  
  nome char(40) not null,  
  endereco char(50) not null,  
  cidade char(20) not null,  
  estado char(2) not null,  
  cep char(9) not null);
```

Tabela conta

```
Create table conta (  
  numero char(6) primary key,  
  valor decimal(10,2) not null,  
  vencimento date not null,  
  codcli char(3) not null);
```

Relacionamento de tabelas

- Para determinar o relacionamento entre tabelas, temos que ter no mínimo duas tabelas que possuam algum campo em comum. No exemplo das tabelas cliente e conta, existe o campo de código de cliente (codigo na tabela cliente e codcli na tabela conta) como campo comum, pois apesar de terem nomes diferentes, a estrutura e o tipo são iguais.

Relacionamento de tabelas

- Imagine a necessidade de obter uma relação das contas existentes e seus respectivos clientes.

SELECT numero, codcli

FROM conta;

- Muito simples pois tanto o campo numero como cod cli estão na mesma tabela, mas imagine agora que eu quero exibir o nome do cliente e não o código

Relacionamento de tabelas

- Dentro do conceito exposto, imagine a necessidade de obter uma relação das contas existentes (tabela conta) e o nome dos clientes (tabela cliente) que possuem essas contas.

```
SELECT conta.numero, cliente.nome  
FROM cliente, conta  
WHERE cliente.codigo=conta.codcli;
```

Esse comando pode ser entendido como selecione os campos nome de cliente e numero da união das tabelas cliente e conta onde o codigo de cliente seja igual ao codcli de conta.

Exercícios

- Apresentação de uma listagem ordenada por nomes de clientes, mostrando a relação de contas que cada um possui e seus respectivos valores.
- Listagem que apresente as contas existentes do cliente “Organização Tupiniquim”. Na listagem devem constar o nome do cliente, o número da conta e seu valor correspondente.
- Apresentar os nomes dos clientes e a data de vencimento de todas as contas do mês de setembro de 2009. A listagem deve ser apresentada na ordem cronológica de vencimento.
- Apresentação do nome dos clientes e de todas as contas que possuem vencimento no mês de outubro de qualquer ano.

Informações Agrupadas

- Obter a quantidade de contas existente de cada cliente.
- Para solucionar esta necessidade, deve-se utilizar junto a WHERE a cláusula GROUP BY

```
SELECT codcli, COUNT(*) FROM conta  
GROUP BY codcli;
```


Informações Agrupadas

- Para exibir o nome do cliente e não o código...

```
SELECT cliente.Nome, COUNT(*)  
FROM cliente, conta  
WHERE cliente.codigo = cobranca.codcli  
GROUP BY cliente.Nome;
```

Podemos usar a cláusula group by para calcular a média de contas para cada elemento de um cliente.
Exemplo:

```
SELECT codcli, avg(valor) FROM  
conta GROUP BY codcli;
```

Observe que quando usamos group by a função avg() retorna a média calculada para cada componente do grupo especificado, no caso “codcli”.

Podemos também testar o valor retornado por avg():
Por exemplo: Exibir os clientes que possuem a média dos valores das contas maior que 2000

```
SELECT codcli, avg(valor) FROM conta  
GROUP BY codcli having  
avg(valor)>2000;
```

Note que a cláusula having usada aqui tem o mesmo significado que where nas consultas normais.

OBS: a cláusula “where” não pode ser usada para restringir **grupos** que deverão ser exibidos. Acompanhando o group by utilizamos a cláusula “having”.

Rollup

Quando se utiliza a cláusula group by, como por exemplo, totalizar as contas por cliente, o gerenciador retorna a consulta com uma linha para cada cliente. O modificador ROLLUP faz com que o mysql retorne também as linhas totalizadas, ou seja, o total por cliente e o total geral.

Exemplo:

```
Select codcli, sum (valor) From conta  
Group by codcli with ROLLUP;
```

Rollup

| codcli | sum(valor) |
|--------|------------|
| 300 | 2600 |
| 500 | 900 |

Select codcli,sum(valor) from
conta group by codcli;

| codcli | sum(valor) |
|--------|------------|
| 300 | 2600 |
| 500 | 900 |
| | 3500 |

Select codcli,sum(valor) from
conta group by codcli with
rollup;

Rollup

Exemplo mais complexo:

```
Select cliente.nome, conta.numero , sum (conta.valor)as  
total  
From conta ,cliente where conta.codcli = cliente.codigo  
Group by cliente.nome, conta.numero with ROLLUP;
```

Qualificadores

- AS
 - define um nome(“alias”) para uma coluna diferente do rótulo de coluna original
- ALL
 - Especifica que a consulta deverá extrair todos os elementos indicados – é o padrão
- DISTINCT
 - Faz com que o SQL ignore valores repetidos na tabela.

Definição de Apelidos

- SQL permite dar um nome diferente de uma tabela, campo ou fórmula do nome real existente.
- Isso é conseguido com o comando AS utilizado junto com o SELECT
- Quando pedimos para exibir o nome e a quantidade de contas existentes para cada cliente, foram apresentadas as colunas nome e count. Para que a coluna count seja chamada de contas usamos a seguinte sintaxe:

```
SELECT cliente.Nome, COUNT(*) AS contas  
FROM cliente, conta  
WHERE cliente.codigo = conta.codcli  
GROUP BY cliente.Nome;
```


DISTINCT

- Queremos saber quais as cidades dos nossos clientes, mas como podemos ter vários clientes da mesma cidade, usamos o **DISTINCT** para não mostrar várias vezes o mesmo nome de cidade.
- **SELECT DISTINCT cidade FROM Clientes;**

Exercícios

- Apresentar uma listagem identificada pelos apelidos Cliente (para representar o campo nome) e Vencidos (para representar o número de contas vencidas existente na tabela conta que será calculada pela função count) de todos os clientes que possuem contas com vencimento anterior a 31/12/2009.
- Apresentar uma listagem de contas em atraso, anteriores à data de 31/12/2009, em que devem ser apresentados, além do nome do cliente, o valor da conta, o valor dos juros (10%) e o valor total a ser cobrado, ordenados por cliente.

Subquery

Uma subquery é um comando `SELECT` inserido em uma cláusula de um outro comando SQL. Pode-se desenvolver comandos sofisticados a partir de comandos simples, utilizando-se subqueries. Elas podem ser muito úteis quando for necessário selecionar linha a partir de uma tabela com uma condição que dependa de dados da própria tabela.

Subquery

A subquery geralmente é identificada como um comando aninhado SELECT. Em geral, ela é executada primeiro e seu resultado é usado para completar a condição de pesquisa para a pesquisa primária ou externa.

Regras Gerais

- A subquery deve ser colocada entre parênteses, deve ser colocada depois de um operador de comparação, cláusula ORDER BY não deve ser incluída em uma subquery.

Exemplo:

```
mysql> SELECT nome FROM cliente WHERE cidade = (SELECT cidade FROM cliente where nome='Elaine Brito');
```

1º descobrirá a cidade de Elaine Brito, depois exibirá o nome dos clientes da mesma cidade que Elaine Brito.

Exercícios

Para o exercício de cliente e conta:

Exiba o código do cliente que possui a conta de menor valor

```
Select codcli from conta where  
valor=(select min(valor) from conta);
```

Outro exemplo

```
Select RA from ALUNOS where  
NOTA=(select max(NOTA) from ALUNOS);
```

Utilizando JOINS

Utilizar a cláusula WHERE para fazer seus JOINS (relacionamentos), limita os relacionamentos a apenas um tipo deles, o INNER JOIN.

Temos três tipos de Joins:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

INNER JOIN

Retorna apenas as linhas das tabelas que sejam comuns entre si, ou seja, as linhas em ambas as tabelas que possuam o campo de relacionamento com o mesmo valor.

No exemplo anterior, somente as pessoas que possuem contas são exibidas.

LEFT JOIN

Ir  listar todas as linhas da primeira tabela relacionada no join, logo ap s a cl usula from.

Quando a linha listada n o possuir equival ncia na tabela destino , as colunas da tabela destino aparecer o com valores nulos

RIGHT JOIN

Ir  listar todas as linhas referentes   segunda tabela relacionada no join

Neste caso tamb m , quando a linha listada n o possuir equival ncia na tabela destino , as colunas da tabela destino aparecer o com valores nulos

Exemplos

Crie as seguintes tabelas:

Cli

Com os campos :

Codigo - inteiro - auto numeração – chave

Nome – char (30)

Lance os seguintes registros:

| Codigo | Nome |
|--------|-----------|
| 1 | José |
| 2 | Elisio |
| 3 | Roberto |
| 4 | Guilherme |

Pedido

com os campos:

nr - inteiro – chave

cliente – inteiro

valor – float(5,2)

Lance os seguintes registros:

| nr | Cliente | valor |
|----|---------|--------|
| 1 | 2 | 100.50 |
| 2 | 2 | 120.00 |
| 3 | 1 | 20.00 |
| 4 | 3 | 60.00 |
| 5 | 3 | 110.00 |

INNER JOIN

```
Select cli.nome, pedido.nr, pedido.valor  
From pedido inner join cli  
On (pedido.cliente = cli.codigo);
```

Observe que o cliente Guilherme não fez nenhum pedido. E se você quiser um relatório que mostre também os clientes que não fizeram nenhum pedido? Você terá que usar uma junção chamada left join:

```
mysql> select pedido.nr,cli.nome,pedido.valor  
-> from cli left join pedido  
-> on cli.codigo=pedido.cliente;
```

| nr | nome | valor |
|------|-----------|--------|
| 3 | José | 20.00 |
| 1 | Elisio | 100.50 |
| 2 | Elisio | 120.00 |
| 4 | Roberto | 60.00 |
| 5 | Roberto | 110.00 |
| NULL | Guilherme | NULL |

6 rows in set (0.00 sec)

Veja que agora o cliente que não fez nenhum pedido, no caso o Guilherme, também foi exibido. Observe que com left join você deve usar on no lugar de where.

Se quiser exibir apenas os clientes que não fizeram nenhum pedido, use algo como:

```
mysql> select pedido.nr,cli.nome,pedido.valor  
-> from cli left join pedido  
-> on cli.codigo=pedido.cliente  
-> where pedido.nr is null;  
+-----+-----+-----+  
| nr      | nome          | valor  |  
+-----+-----+-----+  
| NULL    | Guilherme    | NULL   |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

Lance mais alguns Clientes na tabela Cli mas não lance Pedidos para eles.

```
mysql> select pedido.nr,cli.nome,pedido.valor  
-> from cli left join pedido  
-> on cli.codigo=pedido.cliente;
```

| nr | nome | valor |
|------|-----------|--------|
| 3 | José | 20.00 |
| 1 | Elisio | 100.50 |
| 2 | Elisio | 120.00 |
| 4 | Roberto | 60.00 |
| 5 | Roberto | 110.00 |
| NULL | Guilherme | NULL |
| NULL | Elaine | NULL |
| NULL | Maria | NULL |
| NULL | Thais | NULL |

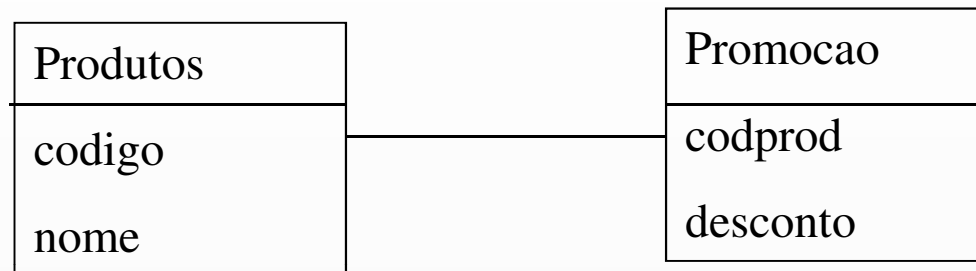
```
9 rows in set (0.00 sec)
```


Abaixo vemos exemplos do uso de count():

```
mysql> select count(valor) from cli left join pedido
-> on cli.codigo=pedido.cliente;
+-----+
| count(valor) |
+-----+
|           5 |
+-----+
1 row in set (0.03 sec)
```

```
mysql> select count(*) from cli left join pedido
-> on cli.codigo=pedido.cliente;
+-----+
| count(*) |
+-----+
|           9 |
+-----+
1 row in set (0.00 sec)
```

Exercício



| codigo | nome |
|--------|-----------|
| 1 | caderno |
| 2 | caneta |
| 3 | estojo |
| 4 | lapiseira |

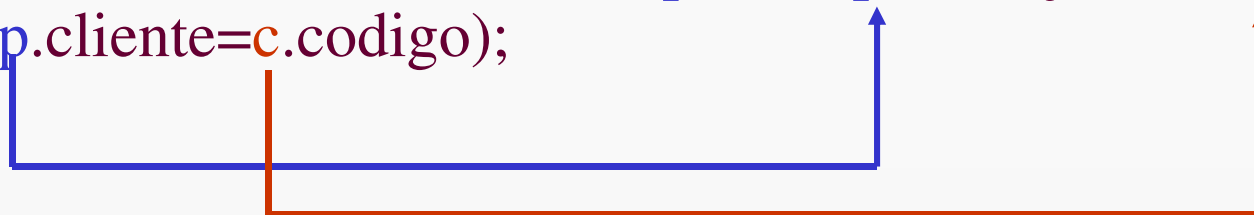
| codprod | desconto |
|---------|----------|
| 1 | 5 |
| 3 | 7 |

Exercício

1. Quero saber o nome dos produtos que estão em promoção.
2. Quero uma lista de nome dos produtos e, quando houver, mostrar também o desconto.

Aliases para Tabelas

- Quando usamos Join, o nome da tabela é citado para diferenciar a qual campo se está fazendo referência. Quando a consulta é complexa e envolve várias tabelas, referenciar o nome da tabela pode aumentar muito o tamanho da consulta e em algumas ferramentas como Delphi, há um limite de 255 caracteres para a consulta.
- Para criar uma alias para uma tabela, basta acrescentar um identificador à frente do nome da tabela. A partir de então, basta utilizar este alias para se referenciar a tabela.
- Select nome,nr,valor from **pedido p** inner join **cli c** on (p.cliente=**c**.codigo);



Exercícios

Criar a tabela Proprietários e a tabela Carros:

- create table proprietarios (rg char (16) primary key, nome char(40));
- create table carros (renavam char (12) primary key, modelo char(20), marca char(20), cor char(10), rg char(16));

Insira os seguintes valores para proprietários:

| rg | nome |
|-----------|-------------------|
| 123456789 | João da Silva |
| 654123987 | Maria de Oliveira |
| 987654321 | José de Souza |

Insira os seguintes valores para carros:

| renavam | modelo | marca | cor | rg |
|--------------|--------|------------|----------|-----------|
| 123456789123 | Fiesta | Ford | Prata | 123456789 |
| 123456789124 | Palio | Fiat | Vermelho | 123456789 |
| 123456789125 | Corsa | Chevrolet | Amarelo | 987654321 |
| 123456789126 | Gol | Volkswagen | Branco | 987654321 |

Listar o Renavam, modelo, marca, cor e nome do proprietário de todos os carros.

- Quando queremos recuperar todas as linhas de uma tabela, inclusive aquelas que não possuem linhas equivalentes na outra, trabalhamos com Right / Left Join

```
mysql> select renavam,modelo,marca,cor,nome as Proprietario
-> from carros c right join proprietarios p
-> on (c.rg=p.rg);
```

| renavam | modelo | marca | cor | Proprietario |
|--------------|--------|------------|----------|-------------------|
| 123456789123 | Fiesta | Ford | Prata | João da Silva |
| 123456789124 | Palio | Fiat | Vermelho | João da Silva |
| | | | | Maria de Oliveira |
| 123456789125 | Corsa | Chevrolet | Amarelo | José de Souza |
| 123456789126 | Gol | Volkswagen | Branco | José de Souza |

- Observe que foram listados todos os proprietários (pois a tabela à direita (right) do join na linha de comando é proprietários), independente de terem um carro relacionado a ele ou não.

- Observe que nesse exemplo específico, se usarmos o left join, como a tabela de carros é a da esquerda(left) e todos os carros tem um proprietário, a resposta é igual à do inner join.

```
mysql> select renavam,modelo,marca,cor,nome as Proprietario
-> from carros c left join proprietarios p
-> on (c.rg=p.rg);
```

| renavam | modelo | marca | cor | Proprietario |
|--------------|--------|------------|----------|---------------|
| 123456789123 | Fiesta | Ford | Prata | João da Silva |
| 123456789124 | Palio | Fiat | Vermelho | João da Silva |
| 123456789125 | Corsa | Chevrolet | Amarelo | José de Souza |
| 123456789126 | Gol | Volkswagen | Branco | José de Souza |

- Se houvesse carros cadastrados sem proprietário, eles apareceriam listados com o campo proprietário vazio.

Exercícios

- Exiba quantos carros tem cada proprietário.
- Observe essa realidade de dados para montar a consulta.

| renavam | modelo | marca | cor | Proprietario |
|--------------|--------|------------|----------|-------------------|
| 123456789123 | Fiesta | Ford | Prata | João da Silva |
| 123456789124 | Palio | Fiat | Vermelho | João da Silva |
| 123456789125 | Corsa | Chevrolet | Amarelo | Maria de Oliveira |
| 123456789126 | Gol | Volkswagen | Branco | José de Souza |

Exercícios

- Agora quero exibir quantos carros tem cada proprietário que possui carros, ou seja, quem não possui nenhum carro não deve ser exibido.

- Voltando agora a trabalhar com as tabelas já feitas anteriormente:

Cli

| codigo | nome |
|--------|-----------|
| 1 | José |
| 2 | Elisio |
| 3 | Roberto |
| 4 | Guilherme |
| 5 | Elaine |
| 6 | Maria |
| 7 | Thais |

Pedido

| nr | cliente | valor |
|----|---------|--------|
| 1 | 2 | 100.50 |
| 2 | 2 | 120.00 |
| 3 | 1 | 20.00 |
| 4 | 3 | 60.00 |
| 5 | 3 | 110.00 |

Para se certificar de ter essas tabelas execute o comando show tables; e depois os comandos select * para cada uma delas.

Execute o comando:

```
mysql> select cli.nome,pedido.nr,pedido.valor  
-> from pedido inner join cli  
-> on pedido.cliente=cli.codigo;
```

| nome | nr | valor |
|---------|----|--------|
| Elisio | 1 | 100.50 |
| Elisio | 2 | 120.00 |
| José | 3 | 20.00 |
| Roberto | 4 | 60.00 |
| Roberto | 5 | 110.00 |

Exercícios:

Exibir o total dos pedidos por cliente:

Rollup

Como vimos, quando se utiliza a cláusula group by, como por exemplo, totalizar os pedidos por cliente, o gerenciador retorna a consulta com uma linha para cada cliente. O modificador ROLLUP faz com que o mysql retorne todos os pedidos e também as linhas totalizadas, ou seja, o total por cliente e o total geral.

Exemplo:

```
Select cli.nome, pedido.nr , sum(pedido.valor)as total  
From pedido left join cli on (pedido.cliente = cli.codigo)  
Group by cli.nome, pedido.nr with ROLLUP;
```

```
mysql> select cli.nome,pedido.nr,sum(pedido.valor) as total
-> from pedido inner join cli
-> on pedido.cliente=cli.codigo
-> group by cli.nome, pedido.nr with rollup;
```

| nome | nr | total |
|---------|------|--------|
| Elisio | 1 | 100.50 |
| Elisio | 2 | 120.00 |
| Elisio | NULL | 220.50 |
| José | 3 | 20.00 |
| José | NULL | 20.00 |
| Roberto | 4 | 60.00 |
| Roberto | 5 | 110.00 |
| Roberto | NULL | 170.00 |
| NULL | NULL | 410.50 |

Verifique a tabela Empregados criada anteriormente, como mostra a figura abaixo:

```
mysql> select * from empregados;
```

| codigo | nome | setor | cargo | salario | admissao |
|--------|-----------------|-------|--------------|---------|------------|
| 3 | Andreia Batista | 3 | Programadora | 1800.00 | 2000-10-20 |
| 4 | Cristiano Souza | 6 | Programador | 1800.00 | 1999-09-10 |
| 7 | Ana Silva | 4 | Secretária | 3000.60 | 2005-05-15 |
| 9 | Silvia Soares | 5 | Supervisora | 1980.00 | 2008-04-25 |
| 20 | Aline Brito | 2 | Supervisora | 1700.00 | 2009-09-05 |
| 22 | Silvia Mendes | 2 | Gerente | 2000.00 | 2009-09-01 |
| 24 | Moacir Campos | 2 | Programador | 2000.00 | 2009-09-10 |
| 25 | João Silva | 4 | Supervisor | 1980.00 | 2000-11-15 |
| 26 | Marco Silva | 2 | Programador | 2000.00 | 2009-09-15 |

9 rows in set (0.00 sec)

Faça uma consulta para verificar a média de salários de cada setor e a média geral dos salários

Faça uma consulta para verificar a média de salários de cada setor, porém apresente resposta apenas para setores com mais de 2 empregados.

Faça uma consulta para verificar a média de salários de cada setor, porém apresente resposta apenas para setores com média maior que R\$1600,00.

OBS: a cláusula “where” não pode ser usada para restringir **grupos** que deverão ser exibidos. Acompanhando o group by utilizamos a cláusula “having”.

Exercícios com Subquery

- Exiba o nome de todos os empregados que trabalham no mesmo setor da Aline Brito
- Exiba o nome de todos os empregados que tem salário superior ao do João Silva

Para o exercício de cliente e conta :

- Exiba o cliente que possui a menor conta
- Exiba o nome do cliente que possui a menor conta
- Exiba a data de vencimento da maior conta
- Exiba a cidade dos cliente que possuem conta acima do valor médio das contas

Exercícios

- Primeiro vamos criar três tabelas no banco de dados: funcionarios, pagamentos e descontos.

- create table funcionarios(
 codigo_funcionario int,
 nome varchar(50))
- create table pagamentos(
 codigo_pagto int,
 codigo_funcionario int,
 valor decimal(10,2))
- create table descontos(
 codigo_desconto int,
 codigo_funcionario int,
 valor decimal(10,2))

Inserindo dados...

- Funcionarios

- 1 Luis
- 2 Marina
- 3 Letícia
- 4 Gustavo
- 5 Mateus

Inserindo dados...

- Pagamentos

1 100

1 200

3 300

5 400

5 500

Inserindo dados...

- Descontos

1 50

2 20

5 30

Exemplo de Inner Join

```
select f.nome, p.valor as pagamento  
from funcionarios f INNER JOIN pagamentos p  
ON f.codigo_funcionario = p.codigo_funcionario;
```

| | nome | pagamento |
|---|---------|-----------|
| 1 | Luis | 100.00 |
| 2 | Luis | 200.00 |
| 3 | Letícia | 300.00 |
| 4 | Mateus | 400.00 |
| 5 | Mateus | 500.00 |

- Apesar de termos cinco funcionários na tabela, ele mostrou apenas três, o motivo é que apenas estes três tem pagamentos. Veja que o inner join fez uma junção entre funcionarios e pagamentos e desconsiderou os funcionários sem pagamentos.

Inner join com três tabelas

- select f.nome, p.valor as pagamento,
d.valor as desconto
from funcionarios f INNER JOIN
pagamentos p
ON f.codigo_funcionario = p.codigo_
funcionario
INNER JOIN descontos d
ON f.codigo_funcionario = d.codigo_
funcionario

| | nome | pagamento | desconto |
|---|--------|-----------|----------|
| 1 | Luis | 100.00 | 50.00 |
| 2 | Luis | 200.00 | 50.00 |
| 3 | Mateus | 400.00 | 30.00 |
| 4 | Mateus | 500.00 | 30.00 |

- Neste caso apenas dois funcionários foram mostrados já que incluímos na consulta os descontos, ou seja, a leitura que esta consulta fez é: mostrar funcionários que tem pagamentos e descontos.

Exemplo de Left join

- ```
select f.nome, p.valor as pagamento
from funcionarios f LEFT JOIN
pagamentos p
ON f.codigo_funcionario= p.codigo_funcionario;
```

|   | nome    | pagamento |
|---|---------|-----------|
| 1 | Luis    | 100.00    |
| 2 | Luis    | 200.00    |
| 3 | Marina  | NULL      |
| 4 | Letícia | 300.00    |
| 5 | Gustavo | NULL      |
| 6 | Mateus  | 400.00    |
| 7 | Mateus  | 500.00    |

- Os funcionários 3 e 5 não tem pagamentos, mas ainda assim eles apareceram na consulta, já que a função Left Join considera apenas a coluna da esquerda e retorna Null (nulo) quando a coluna da direita não tiver um valor correspondente.

# Incluindo o desconto...

- select f.nome, p.valor as pagamento,  
d.valor as desconto  
from funcionarios f LEFT JOIN  
pagamentos p  
ON f.codigo\_funcionario= p.codigo\_  
funcionario  
LEFT JOIN descontos d  
ON f.codigo\_funcionario= d.codigo\_  
funcionario

|   | nome    | pagamento | desconto |
|---|---------|-----------|----------|
| 1 | Luis    | 100.00    | 50.00    |
| 2 | Luis    | 200.00    | 50.00    |
| 3 | Marina  | NULL      | 20.00    |
| 4 | Letícia | 300.00    | NULL     |
| 5 | Gustavo | NULL      | NULL     |
| 6 | Mateus  | 400.00    | 30.00    |
| 7 | Mateus  | 500.00    | 30.00    |

- O que fizemos foi uma espécie de left join em cascata e é útil quando queremos partir de uma base (funcionarios) e listar todas as correspondências ou não das tabelas (pagamentos e descontos) a ela relacionadas

# Criando Tabelas

```
CREATE TABLE fornecimento
(produto INT(4)
UNSIGNED (sem sinal)
ZEROFILL (preenchido por zero int(5) valor 4
 retorna 00004)
DEFAULT '00000' (se nada for digitado ele
 preenche com default)
NOT NULL, (não permite valor nulo)
fornecedor CHAR(20)
DEFAULT ''
NOT NULL,
preco DOUBLE(16,2)
DEFAULT '0.00'
NOT NULL,
PRIMARY KEY(produto, fornecedor)); (chave
```



# Carregando dados em tabela vazia

- Se você está começando com uma tabela vazia, um caminho fácil para preencher é criar um arquivo de texto contendo uma linha para cada um de seus produtos, então carregue os conteúdo do arquivo dentro da tabela com uma declaração única.
- Você pode criar um arquivo de texto "fornecimento.txt" contendo um registro por linha, com valores separados por tabulações, conforme as colunas foram listadas na declaração de CREATE TABLE. Para valores desconhecidos, usar valores NULL. Para representá-los em seu arquivo de texto, use \N

# Carregando dados em tabela vazia

- Exemplo:
- Digite no bloco de notas:

```
1 ' ' 2
2 'aaa' 1.5
```

Com tab entre cada campo.

Salve esse arquivo texto com o nome  
fornecimento.txt

# Carregando dados em tabela vazia

- Para carregar o arquivo de texto "fornecimento.txt" dentro da tabela de produtos use este comando:

```
LOAD DATA LOCAL INFILE "fornecimento.txt" INTO
TABLE fornecimento;
```

Aqui você deve indicar o caminho completo até chegar no arquivo. Ex  
C:/meus documentos/fornecimento.txt

# Exercícios

- Exiba produto, fornecedor e preço de quem tem o produto mais caro da loja.
- Qual é o preço mais alto por produto?

# Fazendo perguntas ao MySQL

- `Select version();`
- `Select current_date;`
- `Select user();`
- Podemos perguntar 2 coisas ao mesmo tempo:
- `Select version(), current_date;`
- `Select version(); Select now();`

# Extras

- Podemos usar mysql como calculadora!  
Digite na linha de comando:
- `Select 2+2;`
- `Select pi();`
- `Select cos(pi());`
- `Select cos(pi()/2);`
- `Select cos(pi())/2;`

- `Select round(pi(),2);` Definindo o número de casa decimais
- `Select round(pi(),10);`
- `Select sqrt(9);`
- `Select abs(-2);`
- `Select mod(5,2);`
- `Select power(3,2);`

# Funções String

- `SELECT CONCAT('My', 'S', 'QL');`
- `SELECT CONCAT_WS(",","First name","Second name","Last Name");`  
-> 'First name,Second name,Last Name'
- `SELECT REPEAT('MySQL', 3);`  
-> 'MySQLMySQLMySQL'
- `SELECT CHAR_LENGTH('Elaine');`



# Funções String

- `SELECT SUBSTRING('Unicamp',5);`  
-> 'amp'
- `SELECT SUBSTRING('Unicamp' FROM 4);`  
-> 'camp'
- `SELECT SUBSTRING('Unicamp',5,2);`  
-> 'am'

# Funções String

- `SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);`  
-> 'www.mysql'
- `SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);`  
-> 'mysql.com'

Retorna a substring da string 'www.mysql.com' por exemplo antes de 2 ocorrências do delimitador .

Se cont é positivo, tudo a esquerda do delimitador final (contando a partir da esquerda) é retornado. Se cont é negativo, tudo a direita do delimitador final (contando a partir da direita) é retornado.

# Funções String

- `SELECT REVERSE('abc');`  
    -> 'cba'
- `SELECT UCASE('Elaine');`  
    -> 'ELAINE'
- `SELECT Upper('Elaine');`  
    -> 'ELAINE'
- `SELECT LCASE('MYSQL');`  
    -> 'mysql'
- `SELECT Lower('MYSQL');`  
    -> 'mysql'

# Exercícios

- Exiba o nome de todos os empregados que tem o mesmo cargo que a Silvia Soares ( obs. Não importa o sexo do funcionário)
- Exiba o nome dos empregados contratados no mesmo ano que Marco Silva

# Funções Data

- `SELECT DAYOFWEEK ('2007-09-15');`  
-> 7
- `SELECT WEEKDAY('2007-09-15');`  
-> 5    OBS:(0 = Segunda, 1 = Terça, ... 6 = Domingo)
- `SELECT DAYOFMONTH ('2007-09-15');`  
-> 15
- `SELECT MONTH ('2007-09-15');`  
-> 9

# Funções Data

- `SELECT DAYNAME('2007-09-15');`  
-> 'Saturday'
- `SELECT MONTHNAME ('2007-09-15');`  
-> 'September'

# Exercícios

- Para acompanhar as explicações a seguir, crie uma tabela para armazenar dados de seus animais de estimação.

```
CREATE TABLE animal (nome VARCHAR(20),
 dono VARCHAR(20),
 especie VARCHAR(20),
 sexo CHAR(1),
 nascimento DATE,
 morte DATE);
```

# Cálculos de Data

- MySQL fornece várias funções que pode fazer cálculos em datas, por exemplo, para calcular idades ou extrair partes das datas.
- Para determinar quantos anos tem seus animais de estimação, ele pega a diferença entre a data de nascimento e a data corrente. Converte as duas datas para dias, tomam a diferença, e divide por 365 (o número de dias em um ano):

```
mysql> SELECT nome,
 (TO_DAYS(NOW())-TO_DAYS(nascimento))/365 FROM
 animal ;
```

```
+-----+-----
--+
```

```
| nome |
 (TO_DAYS(NOW())-TO_DAYS(nascimeto))/365 |
```

```
+-----+-----
```

```
| Kika | 6.15
```



# Cálculos de Data

- Há algumas coisas que podem ser melhoradas. Primeiro, o resultado pode ser examinado mais facilmente se os resultados forem apresentadas em alguma ordem.
- Segundo, o título da coluna de idade não está significativo.
- O primeiro problema pode ser tratado inserindo a cláusula ORDER BY ao nome, para classificar a saída por nome. Para lidar com o título da coluna, fornece um apelido para coluna

```
• SELECT nome,
 (TO_DAYS(NOW())-TO_DAYS(nascimento))/365 AS
```

# Cálculos de Data

- Para classificar a saída por idade em vez de nome, somente use a cláusula ORDER BY:
- ```
mysql> SELECT nome,  
  (TO_DAYS(NOW())-TO_DAYS(nascimento))/36  
  5 AS idade FROM animal ORDER BY idade;
```

Cálculos de Data

- Uma pergunta parecida pode ser utilizada para determinar a idade dos animais quando morreram.

```
SELECT nome, nascimento, morte,  
(TO_DAYS(morte)-TO_DAYS(nascimento))/36  
5 AS idade FROM animal WHERE morte IS  
NOT NULL ORDER BY idade;
```

- A pergunta usa morte IS NOT NULL em vez de morte != NULL porque NULL é um valor especial.

Cálculos de Data

- Se você deseja saber que animais têm aniversários no próximo mês? Para este tipo de cálculo, ano e dia são irrelevantes, você simplesmente deseja extrair o mês da coluna de nascimento. MySQL fornece várias funções de extração de partes da data, tal como YEAR(), MONTH() and DAYOFMONTH(). MONTH() é a função apropriada aqui. Para ver como isto trabalha, faz uma pergunta simples que exibe o valor de ambos data de nascimento e mês(nascimento):

```
SELECT nome, nascimento,  
MONTH(nascimento) FROM animal ;
```

Cálculos de Data

- Descobririndo animais com aniversários no próximo mês é fácil, também.
- Suponha o mês corrente é abril. Então o valor de mês é 4 e você espera animais nascidos no mês de maio (mês 5) :

```
SELECT nome, nascimento FROM animal WHERE MONTH(nascimento)
```

Cálculos de Data

- Há uma pequena complicação se o mês corrente é dezembro. Você não faz somente a soma do número do mês (12) e espera animais nascidos em mês 13, porque não há tal mês.
- Você espera animais nascidos em janeiro (mês 1).
- Você pode escrever a pergunta de modo que você não tenha que usar um número de mês particular na pergunta. `DATE_ADD()` permite você somar um intervalo de tempo para uma data.

- Você deve somar um mês ao valor de NOW(), o resultado produzido é o mês em que esperávamos os aniversários

```
SELECT nome, nascimento FROM animal WHERE  
MONTH(nascimento) = MONTH(DATE_ADD(NOW(),  
INTERVAL 1 MONTH));
```

- Um caminho diferente para efetuar a mesma tarefa é somar 1 para obter o próximo mês depois do mês corrente:

```
SELECT nome, nascimento FROM animal WHERE  
MONTH(nascimento) = MOD(MONTH(NOW()), 12) +  
1;
```

- Note que MÊS volta um número entre 1 e 12 e MOD(alguma coisa,12) volta um número entre 0 e 11.
- EXEMPLO: MOD (1,12) → 1

MOD (2,12) → 2

MOD (12,12) → 0

- Quando fazemos a adição, vamos sempre para o

Indexação de Tabelas

- Índices permitem encontrar registros com um valor específico de um campo mais rápido.
- Sem um índice: inicia a leitura do primeiro registro percorre toda a tabela até que encontre os registros procurados.
- Com um índice para as colunas de interesse: pode obter uma posição adequada para procurar no meio do arquivo de dados sem ter que varrer todos os registros.

Indexação de Tabelas

- Índice é um recurso que facilita a localização de informações dentro de uma tabela, além de possibilitar a exibição ordenada de informações.
- A indexação precisa ser usada com cuidado pois causa lentidão, principalmente nas operações de atualização de dados (update).
- Chave primária não devem ser indexados pois o fato de ser chave primária já cria indiretamente uma indexação.

Indexação de Tabelas

- A indexação é útil quando:
 - Um determinado campo não deve conter valores duplicados, além do campo definido como chave primária. Para o campo que for chave única deve ser criado um índice e utiliza-se a cláusula opcional UNIQUE.
 - Exemplo: Imagine além do campo código do cliente (que está definido como chave primária) o campo CNPJ não pode conter valores duplicados. Neste caso, para o campo CNPJ, cria-se um índice.
 - Quando consultas são realizadas com grande frequência sobre um campo que não é definido como chave primária.
 - Exemplo: Pode-se consultar cliente pelo código ou pelo CNPJ.

Indexação de Tabelas

- A seguir será criado um índice de nomes para a tabela Empregados , usando como chave de indexação o campo nome. A indexação criada está recebendo o nome **nomeind**.

```
mysql> create index nomeind on empregados(nome);  
Query OK, 9 rows affected (0.17 sec)  
Records: 9  Duplicates: 0  Warnings: 0
```

Indexação de Tabelas

- Um índice criado pode ser removido com o comando:
 - DROP INDEX <indice> ON <tabela> .

```
mysql> drop index nomeind on empregados;  
Query OK, 9 rows affected (0.16 sec)  
Records: 9  Duplicates: 0  Warnings: 0
```

Chave Estrangeira

- A restrição de chave estrangeira especifica que o valor da coluna (ou grupo de colunas) deve corresponder a algum valor que existe em uma linha de outra tabela.
- Este comportamento mantém a integridade referencial entre duas tabelas relacionadas.
- Para trabalharmos com integridade referencial, é necessário criar as tabelas como InnoDB.

Exemplo

- `CREATE TABLE estudante (
ra INT NOT NULL AUTO_INCREMENT
PRIMARY KEY,
nome CHAR(30) NOT NULL
) ENGINE=InnoDB;`

```
CREATE TABLE treinamento (  
id INT NOT NULL AUTO_INCREMENT  
PRIMARY KEY,  
nome CHAR(30) NOT NULL  
) ENGINE=InnoDB;
```

- **CREATE TABLE** notas (
ra INT NOT NULL,
treinamento_id INT NOT NULL,
date DATE NOT NULL,
nota DOUBLE NOT NULL,
PRIMARY KEY(ra, treinamento _id, date),
INDEX indtreina (treinamento _id),
FOREIGN KEY (ra) REFERENCES estudante(id)
ON DELETE CASCADE,
FOREIGN KEY (treinamento _id)
REFERENCES treinamento(id) ON DELETE
RESTRICT
) ENGINE=InnoDB;

Integridade

- A tabela Notas define duas restrições de chave estrangeira, fazendo referência à tabela estudante e à tabela treinamento.
- Isto torna impossível criar registros em Notas com ocorrência de ra ou treinamento _id que não existam nas tabelas estudante e treinamento.
- Porém o que ocorre se um estudante ou um treinamento for deletado depois que já foi feito o registro em Notas que os referenciam?

Integridade

- O InnoDB implementa as seguintes restrições de integridade:
- CASCADE (ao se remover um registro da tabela referenciada pela chave estrangeira os registros relacionados àquele removido serão eliminados em todas as tabelas relacionadas)
- RESTRICT ou NO ACTION (não permite a remoção de registros que possuam relacionamentos em outras tabelas)
- SET NULL e SET DEFAULT (atribuem os valores DEFAULT ou NULL para as chaves estrangeiras cujos registros relacionados foram excluídos)

Integridade

- No exemplo é possível que um treinamento possua várias avaliações em datas distintas. Neste caso, foram criadas as tabelas como tipo InnoDB (TYPE=InnoDB), para que as regras de integridade sejam respeitadas.
- As regras definidas foram: um CASCADE para estudante , isto é, se for removido um registro da tabela de estudante , todas as suas notas serão removidas automaticamente.
- No caso da tabela de treinamento , não será possível remover um treinamento que possua notas cadastradas para ele.

Integridade

- Além da restrição ON DELETE, o InnoDB permite também o ON UPDATE, que aplica as restrições no caso de atualizações dos campos relacionados entre as tabelas.
- As ações possíveis são as mesmas.

- É importante ressaltar que o FOREIGN KEY não cria automaticamente um índice na(s) coluna(s) referenciada(s). É necessário criar explicitamente um índice nas colunas que serão chaves estrangeiras.
- No exemplo, a coluna ra já é um índice, visto que esta é o primeiro campo da chave primária da tabela.
- Como treinamento_id não é o primeiro campo de nenhuma chave, foi adicionado o índice indtreina para esta chave estrangeira.
- Caso não seja criado o índice nas chaves estrangeiras, o MySQL exibirá o erro "ERROR 1005: Can't create table './test/notas.frm' (errno: 150)", onde o erro significa que há uma definição incorreta das chaves estrangeiras.