

# Banco de Dados Relacional



# Ementa

- ▶ Tipos de Bancos de Dados  
Relacionais x Não Relacionais
- ▶ Modelagem de Banco de Dados
- ▶ Normalização e Desnormalização de Dados
- ▶ Tipos de Dados
- ▶ Comandos DDL
- ▶ Comandos DML
- ▶ Comandos DQL
- ▶ Comandos DCL (Márcio)
- ▶ Comandos TCL (Márcio)
- ▶ Otimização de Queries (Márcio)

# Introdução ao MySQL

- ▶ MySQL é um Sistema Gerenciador de Banco de Dados (SGBD) relacional.
- ▶ Utiliza a linguagem SQL (Structured Query Language, ou traduzindo, Linguagem de Consulta Estruturada).
- ▶ MySQL é multiusuário e multitarefas.

# Introdução ao MySQL

- ▶ Inicialmente desenvolvido para trabalhar com projetos de pequeno e médio porte, com a capacidade de suportar por volta de cem milhões de registros em cada tabela, podendo chegar ao tamanho médio de aproximadamente cem megabytes por tabela.
- ▶ Nas versões mais recentes o MySQL ultrapassa (e muito) esses limites e capacidades das primeiras versões.

# Introdução ao MySQL

- ▶ MySQL é conhecido por ser de fácil utilização, e usado por empresas que trabalham com grandes volumes de dados, tais como, Google, NASA, Bradesco, HP, Sony entre outras grandes empresas de renome.
- ▶ Possui uma interface extremamente simples e é compatível com grande parte dos sistemas operacionais.
- ▶ Podemos dizer que essas são duas das grandes características que fazem o MySQL ser tão utilizado atualmente e estar em constante crescimento.

# Compatibilidade

- ▶ MySQL é compatível com a maioria dos sistemas operacionais existentes atualmente no mercado.
- ▶ Pelo fato de ser desenvolvido em C e C++, isso faz com que seja extremamente fácil sua acessibilidade entre os diversos sistemas operacionais.

# Compatibilidade

- ▶ Vejamos alguns desses sistemas.
  - ▶ Windows: Compatível com todas as versões
  - ▶ Linux: Sendo compatível com as principais versões, como Fedora, Core, Debian, SuSE e RedHat.
  - ▶ Unix: Sendo compatível com as versões Solaris, HP-UX, AIX, SCO.
  - ▶ FreeBSD.
  - ▶ Mac OS X Server.

# Licença

- ▶ O MySQL é de Código Aberto (Open-Source), desenvolvido e distribuído sob as licenças GNU/GLP (General Public Licence, ou traduzindo, Licença Pública Geral), a qual determina o que se pode ou não fazer à ferramenta e demais recursos.
- ▶ Além do programa, o seu código-fonte também é disponibilizado para que qualquer usuário possa editá-lo de forma que atenda suas necessidades.



# Licença

- ▶ Os princípios básicos da licença GNU/GLP são:
  - ▶ Utilização: Permite que o usuário faça uso do software para qualquer finalidade.
  - ▶ Distribuição: Livre distribuição do software entre quaisquer pessoas.
  - ▶ Didática: Permite que seu funcionamento seja estudado através de seu código-fonte
  - ▶ Colaboração: Possibilita que seu código-fonte seja modificado para evoluir a ferramenta. Como regra seu novo código-fonte tem que permanecer sendo livre segundo essa licença.

# Características

## ► Portabilidade

Devido o MySQL ter sido desenvolvido em C e C++, tornou-se extremamente fácil a portabilidade entre os diferentes sistemas, plataformas e compiladores. Possui também módulos de interface para múltiplas linguagens, tais como Delphi, Java, Python, PHP, ASP, Ruby e entre outras linguagens mais.

# Características

## ► Formas de Armazenamento

O MySQL possibilita diversos tipos de tabela para o armazenamento dos dados, tendo em conta que cada tipo tem suas próprias características. Dessa maneira temos a possibilidade de escolhermos o tipo de acordo com cada situação diferente. Enquanto um tipo tem como prioridade a velocidade, outro da prioridade ao volume de dados, entre outras características.

# Características

## ► Velocidade

Alta velocidade no acesso dos dados em razão de diversos motivos em seu desenvolvimento com tabelas ISAM, que foi substituído pelo novo sistema MyISAM na versão 5 do MySQL, além de utilização de caches em consultas, utilização de indexação, algoritmos de busca, entre outros recursos.

# Características

## Capacidade

O MySQL possui um alto poder de execução e de armazenamento. De acordo com a plataforma em que seja usado, suas tabelas poderão armazenar grandes volumes de dados, o limite ficará por conta somente do tamanho máximo de arquivos que a plataforma que estiver sendo utilizada puder manipular.

# Características

Já no caso de tabelas do tipo InnoDB, onde o armazenamento pode ser realizado em um ou vários arquivos separados, fica possível armazenar volumes de dados equivalentes a TB (Terabytes) de tamanho. E referente a expressões SQL, o MySQL suporta execuções de script SQL com até 61 milhões de tabelas "joins".

O MySQL, por ser um banco de dados poderoso, tem a capacidade de realizar bilhões de consultas em um único dia em um site e também fazer o processamento de milhões de transações por minuto.

# Características

- ▶ Como já sabemos, o MySQL trabalha com a linguagem SQL, sendo extremamente rápido. E isso foi possível devido a SQL ter sido implementada no MySQL através de códigos e funções altamente customizadas pelos seus desenvolvedores. Isso gerou a grande vantagem de velocidade no processamento dos códigos SQL.
- ▶ Porém, ao mesmo tempo trouxe um ponto negativo, sendo ele o fato de que com essa customização, nem todos os padrões das versões mais atuais do SQL tenham sido trazidos para o MySQL, porque poderiam prejudicar consideravelmente a velocidade do banco de dados. Entretanto, essa desvantagem influencia muito pouco na aplicação desenvolvida.

# Conceito de Banco de Dados

- ▶ Uma base de dados, nada mais é do que estruturas complexas de dados. Estes dados são gravados em forma de registros em tabelas.
- ▶ Imagine um arquivo de fichas, numa empresa onde há várias caixas, cada uma contendo os dados dos funcionários de um certo setor.
- ▶ Cada caixa possui várias fichas, que são os cadastros dos funcionários – cada ficha contém os dados de apenas um funcionário.



# Conceito de Banco de Dados

- ▶ Indo mais longe, podemos concluir que cada ficha contém diversas informação sobre o funcionário em questão.
- ▶ Portanto, cada caixa é uma tabela, contendo diversas fichas, que são os registros, e cada ficha possui várias informações sobre o funcionário, que são os campos.
- ▶ Como foi dito, há várias caixas, uma para cada departamento, a soma de todas as caixas forma a base de dados.

# Conceito de Banco de Dados

- ▶ Observando tudo isto de fora, podemos formar o seguinte esquema:
- ▶ Cenário exemplo de fichas em caixas:
  - ▶ Base de dados > Tabela > Registro > Coluna
- ▶ Em banco de dados relacional:
  - ▶ Banco de dados > Tabela > Linha > Campo

# Conceito de Banco de Dados

- ▶ Banco de dados > Tabela > Linha > Campo
- ▶ A linha acima mostra os termos normalmente usados para o que acabamos de aprender.
- ▶ Os campos podem ser de diferentes tipos e tamanhos, permitindo ao programador criar tabelas que satisfaçam ao escopo do projeto.

# Conceito de Banco de Dados

- ▶ A decisão de quais campos usar e quais não usar é muito importante, pois influi drasticamente na performance da base de dados que estamos desenvolvendo, portanto, é de bom grado um conhecimento sólido destes conceitos.
- ▶ A etapa de montagem das tabelas, é senão a mais importante, uma das mais importantes etapas da montagem de uma base de dados, pois um bom projeto pode facilitar muito o trabalho de programação.

# Campos de Banco de Dados

- ▶ Como já sabemos, os campos são a parte fundamental de uma base de dados. É nos campos que as informações ficam armazenadas.
- ▶ Para uma otimização da base de dados, antes de utilizar, devemos definir os campos que desejamos usar, e especificar o que cada um pode conter.

# Campos de Banco de Dados

- ▶ O MySQL oferece campos bem comuns, que até mesmo um programador novato já deve ter visto.
- ▶ Alguns deles estão aqui listados:
  - ▶ CHAR
  - ▶ VARCHAR
  - ▶ INT
  - ▶ FLOAT
  - ▶ DATE
  - ▶ TEXT/BLOB

# Atividade 1

- ▶ O que é MySQL?
- ▶ Quais as vantagens de usar MySQL?
- ▶ O que é Banco de Dados?
- ▶ Qual o conceito de banco de dados?
- ▶ O que é uma tabela em banco de dados?
- ▶ O que é um registro em banco de dados?
- ▶ O que é um campo em banco de dados?
- ▶ O que são 'tipos de dados' em banco de dados?

# TIPOS DE CAMPOS SQL



# Tipos de campos de Banco de Dados

- ▶ CHAR (M)
- ▶ Os campos CHAR são usados para caracteres alfanuméricos, como endereços e nomes. Seu tamanho é fixo e instaurado ao ser criado. Um campo do tipo CHAR pode ter de 1 a 255 caracteres.

# Tipos de campos de Banco de Dados

- ▶ Exemplo:
  - ▶ bairro CHAR(30);
- ▶ Define um campo chamado 'bairro', que pode conter até trinta letras. Observe que não há acentos no nome do campo, pois muitos servidores não acentuam, e sua tabela teria difícil acesso.

# Tipos de campos de Banco de Dados

- ▶ VARCHAR(M)
- ▶ Sua estrutura e funcionamento é idêntico ao campo anterior, salvo que no tipo CHAR, o tamanho definido é fixo, e mesmo que não usado, aquele espaço em disco é alocado. Já o campo VARCHAR, aloca apenas o espaço necessário para gravação. Contudo, vale lembrar que trocamos espaço por velocidade, pois este campo é 50% mais lento que o anterior.

# Tipos de campos de Banco de Dados

- ▶ Exemplo:
  - ▶ bairro VARCHAR(30);
- ▶ Define um campo chamado bairro que pode conter até trinta letras. Se você preencher apenas duas, o campo não ocupará todos os trinta bytes, mas apenas dois.

# Tipos de campos de Banco de Dados

			MIN	MAX
CHAR	String de tamanho fixo. Sempre é completada com espaços a direita até o tamanho definido		1	255 caracteres
		OBS	Espaços excessivos são removidos quando o valor é trazido. Os valores são ordenados e comparados ignorando caixas altas e baixas de acordo com a codificação padrão, a menos que seja fornecido uma chave binária.	
VARCHAR	String de tamanho variável		1	255 caracteres
		OBS	Os valores são ordenados e comparados ignorando caixas altas e baixas de acordo com a codificação padrão, a menos que seja fornecido uma chave binária. Nota: Espaços excessivos são removidos quando o valor é inserido.	
TINYTEXT			0	255 ( $2^8 - 1$ ) caracteres
TEXT			0	65535 ( $2^{16} - 1$ ) caracteres
MEDIUMTEXT			0	16777215 ( $2^{24} - 1$ ) caracteres
LONGTEXT			0	4294967295 ( $2^{32} - 1$ ) caracteres

# Tipos de campos de Banco de Dados

- ▶ INT(M) [Unsigned]
- ▶ O campo INT, que como o próprio número diz, suporta o conjunto dos números inteiros, originalmente numa variação de -2.147.483.648 a 2.147.483.647.
- ▶ O parâmetro Unsigned pode ser passado, excluindo os números negativos, proporcionando um intervalo de 0 até 4.294.967.295.

# Tipos de campos de Banco de Dados

- ▶ Exemplo 01:

- ▶ temperatura INT;

- ▶ Campo com nome temperatura que pode ter valores positivos e negativos.

- ▶ Exemplo 02:

- ▶ temperatura INT unsigned;

- ▶ Campo com nome temperatura que pode ter apenas valores positivos.

# Tipos de campos de Banco de Dados

Type	Storage	Minimum Value	Maximum Value
	(Bytes)	(Signed/Unsigned)	Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615



# Tipos de campos de Banco de Dados

- ▶ `FLOAT(M,D)`
- ▶ Os pontos flutuantes (FLOAT) representam pequenos números decimais, e são usados para representar números com maior precisão.
- ▶ Exemplo:
  - ▶ `voltagem_cadeira_eletrica FLOAT(4,2);`
- ▶ Float válido: 1324.50

# Tipos de campos de Banco de Dados

- ▶ DATE
- ▶ Campo usado para armazenar informações referentes a data. A forma padrão, é 'YYYY-MM-DD', onde YYYY corresponde ao ano, MM ao mês, e DD ao dia. Ele pode variar de 1000-01-01 a 9999-12-31.

# Tipos de campos de Banco de Dados

- ▶ Exemplo:
  - ▶ `data_de_nascimento DATE;`
- ▶ Data válida: 1999-12-25
- ▶ Data inválida: 1999-25-12

# Tipos de campos de Banco de Dados

- ▶ TIME
- ▶ Campo usado para armazenar informações referentes a hora. A forma padrão, é HH:MM:SS, onde HH é a hora, MM ao mês e SS os segundos.
- ▶ Exemplo:
  - ▶ `inicio_aula TIME;`

# Tipos de campos de Banco de Dados

## ► DATETIME

- Campo usado para armazenar informações referentes a data e hora. A forma padrão, é YYYY-MM-DD HH:MM:SS, onde YYYY corresponde ao ano, MM ao mês, DD ao dia, HH a hora, MM ao minuto e SS os segundos. Ele pode variar de 1000-01-01 00:00:00 até 9999-12-31 23:59:59.

# Tipos de campos de Banco de Dados

- ▶ Exemplo:
  - ▶ ocorrencia DATETIME;
- ▶ Data válida: 2016-02-26 12:15:00

# Tipos de campos de Banco de Dados

- ▶ TEXT/BLOB
- ▶ Os campos texto e blob são usados para guardar grandes quantidades de caracteres. Podendo conter de 0 a 65535 bytes, os blobs e texts são úteis para armazenar documentos completos, como este que você está lendo.
- ▶ A única diferença entre os campos BLOB e TEXT está no fato de um campo TEXT não ser sensível a letras maiúsculas e minúscula quando uma comparação é realizada, e os BLOBs sim.

# Tipos de campos de Banco de Dados

- ▶ Exemplo:
  - ▶ relatorio BLOB;
  - ▶ BLOB válido: 'Minha terra tem palmeiras onde canta o...'
- ▶ relatorio TEXT;
- ▶ TEXT válido: 'A que saudades que eu sinto...'



# Tipos de campos de Banco de Dados

- ▶ BIT OU BOOLEAN
- ▶ Um número inteiro que pode ser 0 ou 1
- ▶ Usado para representar false/ true
- ▶ Exemplo:
  - ▶ ativo bool;
- ▶ Campo para dizer se o usuário esta ativo

# Tabelas de Banco de Dados

- ▶ Um conjunto de registros, forma uma tabela. As tabelas portanto, armazenam grande quantidade de dados.
- ▶ Como no exemplo anterior, poderíamos ter centenas de nomes diferentes cadastrados em nossa tabela de pessoas. Cada conjunto de dados corresponde a um registro.

# Atividade 2

- ▶ Imagine uma ficha de cadastro de uma pessoa. Para cada dado abaixo dê um nome para o campo, inserindo o tamanho do mesmo quando necessário.
- ▶ Exemplo:
- ▶ Nome completo - nome\_completo VARCHAR(100)

# Atividade 2

- ▶ Casos para fazer:
- ▶ Código de cadastro
- ▶ Nome completo
- ▶ Idade
- ▶ Data de nascimento
- ▶ CPF
- ▶ CNPJ

# Atividade 2

- ▶ Rua
- ▶ Número
- ▶ Complemento
- ▶ CEP
- ▶ Cidade
- ▶ Estado
- ▶ Código de cadastro
- ▶ Data de cadastro
- ▶ Última atualização do cadastro
- ▶ Sexo
- ▶ Religião
- ▶ Cor
- ▶ QI
- ▶ Tipo sanguíneo
- ▶ Se é doador de órgãos
- ▶ Se tem passagem pela polícia
- ▶ Qual seu grau de ensino
- ▶ Se o usuário está ou não ativo

# OPERADORES

# Chave primária

- ▶ Usado para que não seja permitido o cadastro de dois registros com chaves primárias iguais.
- ▶ Isto é claramente útil, quando não é desejado que seja digitado um segundo registro igual ao primeiro por engano.

# Chave primária

- ▶ Para se definir uma chave primária, basta adicionar 'PRIMARY KEY' a definição do campo que se deseja a não duplicidade.
- ▶ Exemplo de uso para chave primária:
  - ▶ Um cadastro de pessoas com entrada VIP em uma festa, cadastradas com base no seu CPF. Não podem haver dois nomes com o mesmo CPF.



# AUTO INCREMENT

- ▶ 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.

# AUTO INCREMENT

- ▶ Exemplo:
  - ▶ código `INT AUTO_INCREMENT;`
- ▶ Automaticamente soma um a cada registro neste campo. Começando de 1, com inserção subsequente.
- ▶ Exemplo de uso para auto incremento:
  - ▶ O número de um pedido, ou até mesmo o número de uma nota fiscal, que sempre vai ser o número anterior + 1;

# UNSIGNED

- ▶ Parâmetro Unsigned pode ser passado, excluindo os números negativos, proporcionando um intervalo de 0 até o tamanho máximo multiplicado por dois.
- ▶ Exemplo:
  - ▶ idade TINYINT
- ▶ Faz a faixa de números ir de -128 até 127
- ▶ idade TINYINT unsigned
- ▶ Faz a faixa de números ir de 0 até 255

# UNIQUE

- ▶ Parâmetro Unique pode ser passado para que um campo seja único dentro de uma tabela.
- ▶ Exemplo:
  - ▶ idade CPF
- ▶ Permite o cadastro de dois ou mais CPFs iguais
  - ▶ idade CPF UNIQUE
- ▶ Não permite o cadastro duplicado.

# NULL e NOT NULL

- ▶ O parâmetro NULL indica se o campo pode ou não estar vazio.
- ▶ Quando dizemos que um campo pode ser NULL, significa que aquela informação, de certa forma, é opcional.
- ▶ Ao indicar um campo como NOT NULL nós estamos dizendo que o preenchimento daquele campo é obrigatório.

# NULL e NOT NULL

- ▶ Exemplo:
  - ▶ cpf CHAR(11) NOT NULL
- ▶ Faz com que o campo CPF seja obrigatório.
  - ▶ celular CHAR(11) NULL
- ▶ Faz com que o campo CELULAR seja de preenchimento opcional

# Atividade 3

- ▶ Complemente a atividade anterior da seguinte forma:
- ▶ Verifique quais campos criados necessitam ser do tipo chave primária. Ou seja, quais campos não podem repetir registro dentro do banco de dados. Nestes campos, adicione o PRIMARY KEY (PK)
- ▶ Analise e coloque NULL para os campos opcionais e NOT NULL para os campos obrigatórios

# Atividade 3

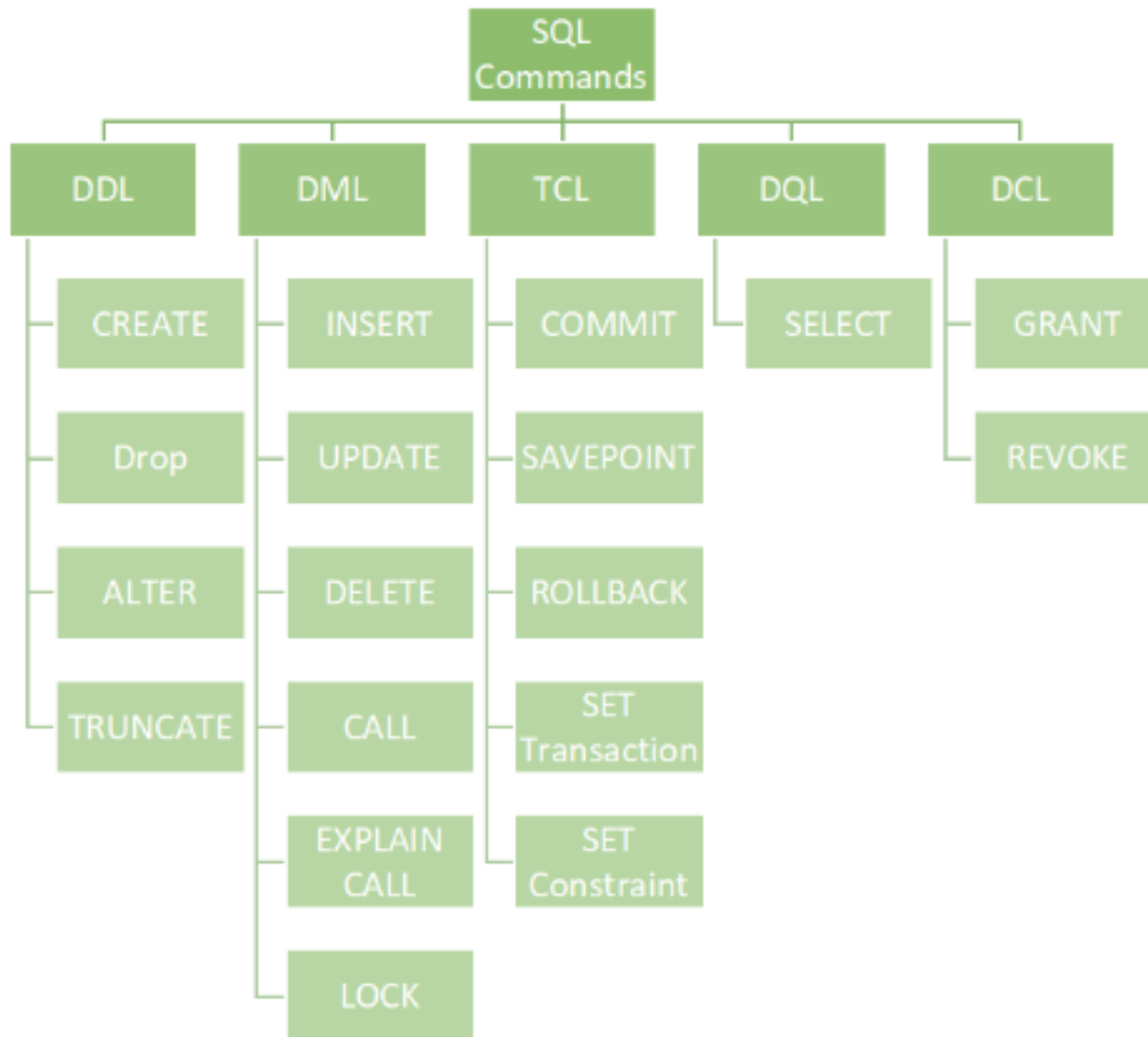
- Verifique também quais campos criados necessitam ser do tipo auto incremento. Ou seja, quais campos precisam ter o valor automaticamente inserido e incrementado. Nestes campos, adicione o AUTO INCREMENT (AI) e também veja quais devem ser Unsigned. Ou seja, campos que não precisam ter números negativos. Nestes campos, adicione o UNSIGNED (U).



# SQL

- ▶ Para todas as operações que precisarmos fazer nós vamos executar um comando SQL.
- ▶ Existem comandos dos mais variados tipos: Para criar bancos, tabelas, campos, exibir informações, retornar consulta, atualizar, excluir, etc.
- ▶ Temos que ter em mente que vamos utilizar um gerenciador de banco de dados chamado MySQL Workbench, que vai agilizar alguns destes comandos para nós.

# Comandos SQL



# DDL - Data Definition Language

# CREATE DATABASE

- ▶ O comando **CREATE** nos permite criar um novo banco de dados.
- ▶ Para criar um novo banco de dados a sintaxe é:
  - ▶ **CREATE DATABASE** nome\_do\_banco\_de\_dados
- ▶ Exemplo:
  - ▶ **CREATE DATABASE** revenda

# USE

- ▶ O comando **USE** seleciona o banco de dados que vamos usar. Depois de criar um banco de dados novo é preciso selecionar o banco para podermos criar as tabelas e fazer uso do mesmo.
- ▶ Para selecionar um banco de dados a sintaxe é:
  - ▶ **USE** nome\_do\_banco\_de\_dados
- ▶ Exemplo:
  - ▶ **USE** revenda

# CREATE TABLE

- ▶ O comando **CREATE** nos permite criar tabelas em um banco de dados selecionado. Nós podemos criar uma tabela vazia, ou criar a mesma já informando quais campos ela deve possuir.
- ▶ Para criar uma tabela vazia a sintaxe é:
  - ▶ **CREATE TABLE** nome\_da\_tabela
- ▶ Exemplo:
  - ▶ **CREATE TABLE** fornecedor

# CREATE TABLE

- ▶ O comando CREATE nos permite criar tabelas em um banco de dados selecionado. Nesta continuação vamos ver como criar uma tabela com os seus devidos campos.
- ▶ Para criar uma tabela e seus campos a sintaxe é:

```
CREATE TABLE nome_da_tabela (  
    'campo1' tipo(tamanho) opcionais,  
    'campo2' tipo(tamanho) opcionais  
)
```

# CREATE TABLE

```
CREATE TABLE 'testes' (  
    'campo1' tinyint(2) UNSIGNED NOT NULL AUTO_INCREMENT,  
    'campo2' smallint(5) UNSIGNED DEFAULT NULL,  
    'campo3' mediumint(9) UNSIGNED DEFAULT NULL,  
    'campo4' int(15) UNSIGNED DEFAULT NULL,  
    'campo5' bigint(25) UNSIGNED DEFAULT NULL,  
    'campo6' varchar(255) DEFAULT NULL,  
    'campo7' char(10) DEFAULT NULL,  
    'campo8' text,  
    'campo9' bit(1) NOT NULL DEFAULT '1', -- o campo9 será NOT NULL e terá o valor 1 como padrão  
    'campo10' float(4,2) NOT NULL,  
    PRIMARY KEY ('campo1'), -- nesta linha estou dizendo para que o campo1 seja do tipo PRIMARY KEY  
    UNIQUE KEY 'campo7' ('campo7') -- nesta linha estou dizendo para que o campo7 seja do tipo UNIQUE  
);
```



# ALTER TABLE

- ▶ O comando **ALTER** nos permite editar um banco de dados ou tabela existente.
- ▶ Exemplos:
- ▶ Inserindo coluna na tabela:
  - ▶ **ALTER TABLE** endereco **ADD** país varchar(25);
- ▶ Remove chave primaria da tabela:
  - ▶ **ALTER TABLE** endereco **DROP** primary key;

# ALTER TABLE

- ▶ Inserindo chave primaria na tabela
  - ▶ `ALTER TABLE endereco ADD PRIMARY KEY(id_endereco);`
- ▶ Modificar definições de uma coluna
  - ▶ `ALTER TABLE endereco MODIFY bairro varchar(50);`
- ▶ Excluir coluna da tabela:
  - ▶ `ALTER TABLE endereco DROP cidade;`
- ▶ Renomear tabela:
  - ▶ `ALTER TABLE endereco RENAME localizacao;`

# SHOW TABLES

- ▶ O comando SHOW mostra todas as tabelas de um banco de dados selecionado.
- ▶ A sintaxe para exibir todas as tabelas de um banco é:
- ▶ **SHOW TABLES**

# DESC

- ▶ O comando DESC exibe os campos de uma tabela.
- ▶ Para ver os campos a sintaxe é:
- ▶ `DESC nome_da_tabela`
- ▶ Exemplo:
  - ▶ `DESC fornecedor`

# DROP DATABASE

- ▶ O comando DROP nos permite excluir um banco de dados.
- ▶ Para excluir um novo banco de dados a sintaxe é:
- ▶ **DROP DATABASE** nome\_do\_banco\_de\_dados
- ▶ Exemplo:
  - ▶ **DROP DATABASE** revenda

# DROP TABLE

- ▶ O comando DROP nos permite excluir tabelas em um banco de dados selecionado.
- ▶ Para excluir uma tabela a sintaxe é:
- ▶ **DROP TABLE** nome\_da\_tabela
- ▶ Exemplo:
  - ▶ **DROP TABLE** fornecedor

# TRUNCATE TABLE

- ▶ O comando TRUNCATE limpa uma tabela. Ou seja, exclui todos os dados dela.
- ▶ Para limpar uma tabela usamos a sintaxe:
- ▶ `TRUNCATE TABLE` tabela
- ▶ Exemplo:
  - ▶ `TRUNCATE TABLE cliente`

# DML - Data Manipulation Language



# Comando SELECT

- ▶ O comando SELECT serve pra exibir dados de uma tabela.
- ▶ Por exemplo, preciso consultar todas as pessoas de SC que estão cadastradas na tabela cliente. Farei o comando SQL abaixo:
- ▶ `SELECT * FROM cliente WHERE uf = 'SC';`

# Comando SELECT

- ▶ O \* significa que deve-se retornar todos os campos da tabela.
- ▶ cliente é o nome da tabela
- ▶ uf = 'SC' é o nosso argumento.
- ▶ Argumento = condição.

# Comando SELECT

- ▶ Eu poderia também retornar apenas alguns campos, ao invés de selecionar todos.
- ▶ Ficaria:
  - ▶ `SELECT nome, marca, veiculo FROM cliente WHERE uf = 'SC';`
- ▶ No lugar do \* eu disse que os campos de nome, marca e veiculo devem retornar da tabela.

# Operadores Condicionais

- ▶ = IGUAL
- ▶ > MAIOR QUE
- ▶ < MENOR QUE
- ▶ <= MENOR IGUAL
- ▶ >= MAIOR IGUAL A
- ▶ != ou < > DIFERENTE
- ▶ AND operador E
- ▶ OR operador OU

# Operadores Condicionais

- ▶ Exemplo:
  - ▶ `SELECT nome, marca, veiculo FROM cliente WHERE idade >= 18;`
- ▶ Pedindo retorno dos dados onde a pessoa tem 18 anos ou mais.

# Atividade 4

- ▶ Abra o MySQL Workbench. Use o banco de dados 'sakila'
- ▶ Responda os itens abaixo:
- ▶ Consulte todos os address que tem como district 'Buenos Aires'
- ▶ Consulte todos os address que o district não seja 'California'

# Pesquisando com LIKE

- ▶ A função LIKE, do MySQL, faz uma busca sofisticada por uma substring dentro de uma string informada. Traduzindo, ele faz uma pesquisa.
- ▶ % - Busca zero ou mais caracteres
- ▶ `SELECT * FROM tabela WHERE campo LIKE critério;`

# Pesquisando com LIKE

- ▶ Exemplo 1: Quero pesquisar os nomes que começam com A:
  - ▶ `SELECT * FROM cliente WHERE nome LIKE 'A%';`
- ▶ Exemplo 2: Quero pesquisar os modelos de carros que terminam com i
  - ▶ `SELECT * FROM cliente WHERE modelo LIKE '%i';`
- ▶ Exemplo 3: Quero pesquisar todos os telefones que tenham 30 em algum lugar do número
  - ▶ `SELECT * FROM cliente WHERE telefone LIKE '%30%';`



# Atividade 5

- ▶ Execute comandos de consulta usando o LIKE:
- ▶ Todos os Atores que começam com a letra 'A';
- ▶ Todos os Atores que possui o ultimo nome igual a 'CAGE';

# Pesquisando com LIKE

- ▶ A função LIKE, do MySQL, faz uma busca sofisticada por uma substring dentro de uma string informada na posição informada. Traduzindo, ele faz uma pesquisa na posição da string que queremos.
- ▶ `_` - Busca somente um caractere.
- ▶ `SELECT tabela WHERE campo LIKE critério;`

# Pesquisando com LIKE

- ▶ Exemplo 1: Quero pesquisar as marcas de carro onde a segunda letra é O:
  - ▶ `SELECT * FROM cliente WHERE marca LIKE '_o%';`
- ▶ Exemplo 2: Quero pesquisar os modelos de carros que a terceira letra é 0:
  - ▶ `SELECT * FROM cliente WHERE modelo LIKE '__0%';`

# Atividade 6

► Execute comandos de consulta usando o LIKE:

1. Endereços onde o telefone tem os quatro primeiros dígitos iniciando com 7058.
2. Endereços onde o telefone tem os segundo dígito iniciando em 6.

# Funções e Operadores

# Função DISTINCT

- ▶ A função **DISTINCT** é utilizada para não mostrar valores repetidos.
- ▶ Se pesquisarmos:
  - ▶ **SELECT** marca **FROM** cliente;
- ▶ O resultado seria: Volkswagen, Volkswagen , Fiat, Volkswagen, Fiat... e muito mais.
- ▶ Porém, eu não quero que apareça a Volkswagen três vezes. Quero apenas saber quais marcas estão na tabela

# Função DISTINCT

- ▶ Podemos resolver isto com a função DISTINCT:
  - ▶ `SELECT DISTINCT(marca) FROM cliente;`
- ▶ O resultado será: Volkswagen, Fiat, Chevrolet, Seat, Renault, Citroen, Toyota, Honda, Ferrari
- ▶ Perceba que neste exemplo eu não estou usando qualquer critério na cláusula WHERE, porém eu poderia usar em qualquer momento. Exemplo:
  - ▶ `SELECT DISTINCT(marca) FROM cliente WHERE ativo = 0`

# Função COUNT

- ▶ A função **COUNT** é utilizada para saber quantos registros existem na consulta.
- ▶ Exemplo:
- ▶ Quero saber quantos registros a tabela cliente tem:
  - ▶ **SELECT COUNT(\*) FROM cliente;**
- ▶ O resultado será: 31



# Função COUNT

- ▶ Note que neste exemplo eu também não estou usando qualquer critério na cláusula WHERE, porém eu poderia usar em qualquer momento.
- ▶ Exemplo:
  - ▶ `SELECT COUNT(*) FROM cliente WHERE celular LIKE '47%';`
- ▶ O resultado será: 10

# Função BETWEEN

- ▶ A função **BETWEEN** é utilizada para consultar valores entre um intervalo de dados previamente definido. Sempre será pedido entre um valor inicial e um valor final.
- ▶ Por exemplo, todos os registros de pessoas que tenham entre 18 e 40 anos.
- ▶ Ou então, todos os registros que tenham carro fabricado entre 2000 e 2008.
- ▶ A sintaxe ficaria da seguinte forma:
  - ▶ **SELECT \* FROM cliente WHERE ano BETWEEN 2000 AND 2008;**

# Operador ORDER BY

- ▶ O operador lógico ORDER BY, ou ORDENAR POR, simplesmente lista os registros, colocando-os em ordem de acordo com o campo solicitado.
- ▶ Exemplo:
  - ▶ `SELECT * FROM cliente WHERE estado = "sc" ORDER BY nome;`
- ▶ Este select retornará todos os dados da tabela cliente ordenando os resultados pelo nome.

# Operador ORDER BY

- ▶ A ordenação pode ser em ordem ascendente (ASC) ou descendente (DESC)
- ▶ Exemplo:
  - ▶ `SELECT * FROM cliente WHERE estado = "sc" ORDER BY nome DESC;`
- ▶ Este select retornará todos os dados da tabela cliente ordenando os resultados pelo nome em ordem descendente.

# Operador LIMIT

- ▶ O LIMIT fará o papel de exibir uma parte dos resultados.
- ▶ Ele indica qual trecho da consulta será retornado.
- ▶ Suponha que você quer retornar os 10 primeiros resultados de uma consulta. Você colocará LIMIT 0,10. Ou seja, exibir a partir do registro zero, e exibir 10 resultados.

# Operador LIMIT

- ▶ Ou então, somente do registro 5 ao 10. Ficaria: LIMIT 5,10
- ▶ A sintaxe é Sempre é LIMIT inicio quantidade
- ▶ Exemplo:
  - ▶ `SELECT * FROM cliente WHERE marca = 'ford' LIMIT 0,5;`
  - ▶ Este select retornará os dados da tabela cliente a partir do primeiro resultado, exibindo 5 registros.
  - ▶ `SELECT * FROM cliente WHERE marca = 'ford' ORDER BY ano ASC LIMIT 0,5;`

# MAX, MIN, AVG

- ▶ A função MAX(campo) retorna o maior valor encontrado para aquele campo na tabela.
- ▶ Exemplo:
  - ▶ `SELECT max(ano) FROM cliente;`

# MAX, MIN, AVG

- ▶ A função MIN(campo) retorna o menor valor encontrado para aquele campo na tabela.
- ▶ Exemplo:
  - ▶ `SELECT min(ano) FROM cliente;`



# MAX, MIN, AVG

- ▶ A função `AVG(campo)` retorna a media dos valores encontrados para aquele campo na tabela.
- ▶ Exemplo:
  - ▶ `SELECT avg(ano) FROM cliente;`

# Comando UPDATE

- ▶ UPDATE é útil quando você quer alterar registros de uma tabela. Você pode alterar um ou mais campos ao mesmo tempo.
- ▶ UPDATE não gera um conjunto de resultados. Se você quiser saber quais resultados serão alterados, examine primeiro os resultados da consulta que use os mesmos critérios e então execute a atualização.

# Comando UPDATE

- ▶ Exemplo:

- ▶ `UPDATE cliente SET nome = 'Anderson Novo' WHERE id = '1';`

- ▶ Estou pedindo para que todos os registros da tabela cliente que possuam o campo id preenchido com 1 tenham o campo nome atualizado para 'Anderson Novo'.

# SQL - Comando UPDATE

- ▶ Sintaxe:

UPDATE *tabela*

SET *campo1* = valornovo, ...

WHERE *critério*;

- ▶ Onde:

- ▶ Tabela - O nome da tabela cujos os dados você quer modificar.
- ▶ Valornovo - Uma expressão que determina o valor a ser inserido em um campo específico nos registros atualizados.
- ▶ Critério - Uma expressão que determina quais registros devem ser atualizados. Só os registros que satisfazem a expressão são atualizados.

# COMANDO DELETE

- ▶ O comando DELETE nos permite excluir um ou mais registros de uma tabela. A exclusão se aplica a todos os registros onde a nossa cláusula WHERE for verdade.
- ▶ Para excluir um dado usamos a sintaxe:
  - ▶ **DELETE FROM** nome\_da\_tabela **WHERE** condição
- ▶ Exemplo:
  - ▶ **DELETE FROM** cliente **WHERE** ativo = 0
- ▶ Obs: Estou pedindo para que a consulta exclua todos os cadastros de clientes onde o estado dele é inativo.

# COMANDO INSERT

- ▶ O comando INSERT nos permite cadastrar dados em uma tabela existente.
- ▶ Para cadastrar um dado usamos a sintaxe:
  - ▶ `INSERT INTO nome_da_tabela (coluna1, coluna2, ...)`  
`VALUES (valor1, valor2, ...)`
- ▶ Exemplo:
  - ▶ `INSERT INTO cliente (nome) VALUES ('Anderson')`
- ▶ Obs: Note que ao invés de eu informar um valor para o ID eu coloquei NULL. O banco de dados vai adicionar o valor deste registro automaticamente, pois cadastramos este campo como AUTO INCREMENTO

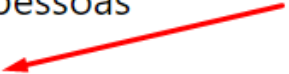
# Foreign Key

- ▶ É uma restrição usada para evitar ações que destruam ligações entre tabelas.
- ▶ A FK é um campo (ou coleção de campos) em uma tabela, que se refere a uma Primary Key de uma outra tabela.
- ▶ A tabela com a chave estrangeira é chamada de tabela filha e a tabela com a chave primária é chamada de tabela referenciada ou pai.

# Foreign Key

Tabela de pessoas


PK - Primary Key



PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Tabela de pedidos

FK Foreign Key



OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1



# Foreign Key

## ► Sintaxe:

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);
```

```
ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

# Foreign Key

## ► Atividade FK

Via script CREATE crie as seguintes tabelas, respeitando as Primary Key e Foreign Key de cada uma.

Tabela de pessoas

PK - Primary Key



PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Tabela de pedidos

FK Foreign Key



OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

# Group By

- ▶ É uma instrução que agrupa as linhas que têm os mesmos valores em linhas.
- ▶ É frequentemente usada com funções agregadas como: COUNT(), MAX(), MIN(), SUM() e AVG() para agrupar o conjunto de resultados por uma ou mais colunas.

# Group By

► Sintaxe:

*SELECT column\_name(s)*

*FROM table\_name*

*WHERE condition*

*GROUP BY column\_name(s)*

*ORDER BY column\_name(s);*

# Normalização e Desnormalização de Dados

- ▶ **Normalização:** É uma técnica de minimizar as anomalias de inserção, exclusão e atualização através da eliminação dos dados redundantes.
- ▶ **Desnormalização:** É o processo inverso de normalização onde a redundância é adicionada aos dados para melhorar o desempenho da aplicação específica e a integridade dos dados.
- ▶ A normalização evita o desperdício de espaço em disco minimizando ou eliminando a redundância.

# Normalização e Desnormalização de Dados

## Gráfico de comparação

Base para comparação	Normalização	Desnormalização
Basic	A normalização é o processo de criação de um esquema definido para armazenar dados não redundantes e consistentes.	Desnormalização é o processo de combinar os dados para que possam ser consultados rapidamente.
Propósito	Reduzir a redundância e inconsistência de dados.	Para alcançar a execução mais rápida das consultas através da introdução de redundância.
Usado em	Sistema OLTP, onde a ênfase está em fazer a inserção, apagar e atualizar anomalias mais rapidamente e armazenar os dados de qualidade.	Sistema OLAP, onde a ênfase está em tornar a pesquisa e a análise mais rápidas.
Integridade de dados	Mantido	Não pode reter
Redundância	Eliminado	Adicionado
Número de tabelas	Aumenta	Diminui
Espaço em disco	Uso otimizado	Desperdício

# Modelagem de Banco de Dados

- ▶ A modelagem de dados é uma etapa importante e essencial em qualquer projeto de desenvolvimento ou manutenção de software.
- ▶ É o ato de explorar estruturas orientadas a dados. Como outros artefatos de modelagem, modelos de dados podem ser usados para uma variedade de propósitos, desde modelos conceituais de alto nível até modelos físicos de dados.
- ▶ É um processo similar a modelagem de uma classe.

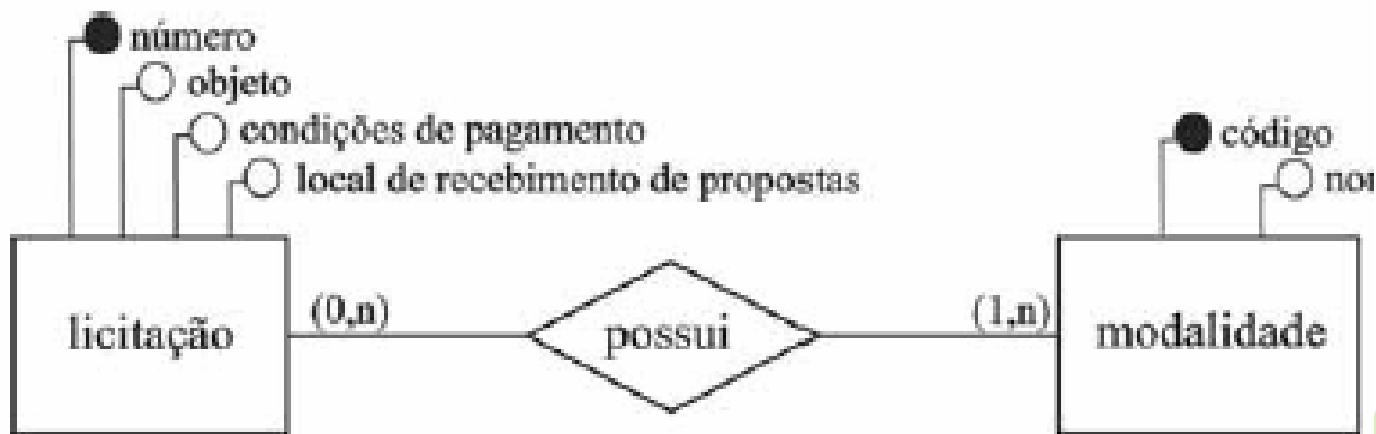
# Modelagem de Banco de Dados

- ▶ Tipos de Modelos
- ▶ **Conceitual:** algumas vezes chamadas modelos de domínio, são tipicamente usados para explorar conceitos do domínio com os envolvidos no projeto.
- ▶ **Lógico:** São usados para explorar os conceitos do domínio e seus relacionados. Descrevem os tipos de entidades lógicas, os atributos de dados e os relacionamentos.



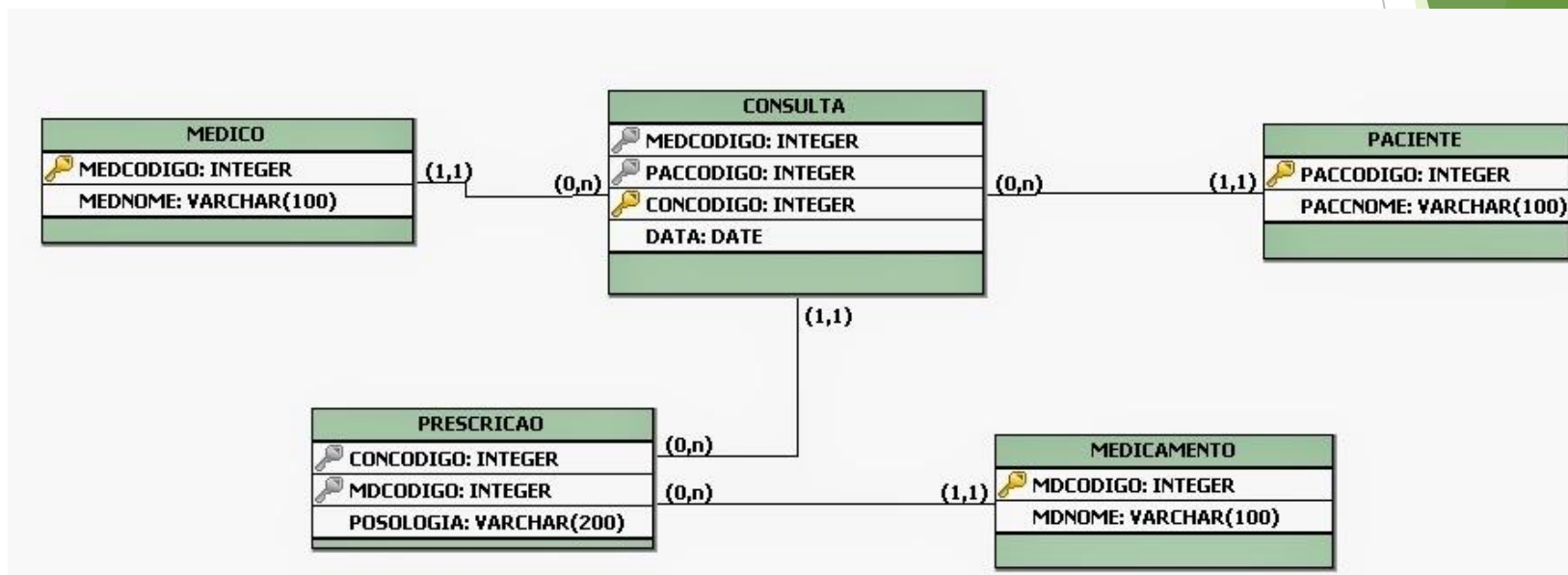
# Modelagem de Banco de Dados

- Ex.: Conceitual ou MER (Modelo Entidade Relacionamento)



# Modelagem de Banco de Dados

## ► Ex.: Modelo Lógico



# Como Modelar?

- ▶ Identificar os tipos de entidades
- ▶ Identificar os atributos
- ▶ Aplicar convenção de nomes
- ▶ Identificar relacionamentos
- ▶ Associar chaves
- ▶ Normalizar para reduzir a redundância dos dados
- ▶ Diversificar para melhorar o desempenho

# Cardinalidade

- ▶ É o número de ocorrências de uma entidade A que está associado com ocorrências de outra entidade B.
- ▶ Há três graus de relacionamento:

Um para Um

Um para Muitos ou Muitos para Um

Muitos para Muitos

# Cardinalidade

- ▶ Nas cardinalidades temos:
- ▶ Cardinalidade Máxima: Número máximo de vezes que uma entidade A pode ocorrer B. Pode assumir o valor de 1 ou N (inúmeras vezes).
- ▶ Cardinalidade Mínima: Número mínimo de vezes que uma entidade A pode ocorrer em B. Pode assumir o valor de 0 ou 1.

# Cardinalidade

- Relacionamentos:

- Relacionamento de Um para Um (1 x 1)

Cada elemento de um entidade A relaciona-se com um e somente um elemento de outra entidade B.

Nesse tipo de relacionamento a cardinalidade mínima influência na modelagem

# Cardinalidade

- Relacionamento de Um para Muitos (1 x N) ou Muitos para Um (N x 1)

O elemento de uma entidade A pode se relacionar com mais de um elemento de outra entidade B.

Durante este tipo de relacionamento a ordem influência no resultado de onde o atributo de referencia também chamado de chave estrangeira será adicionado. Normalmente onde há cardinalidade máxima N.

# Cardinalidade

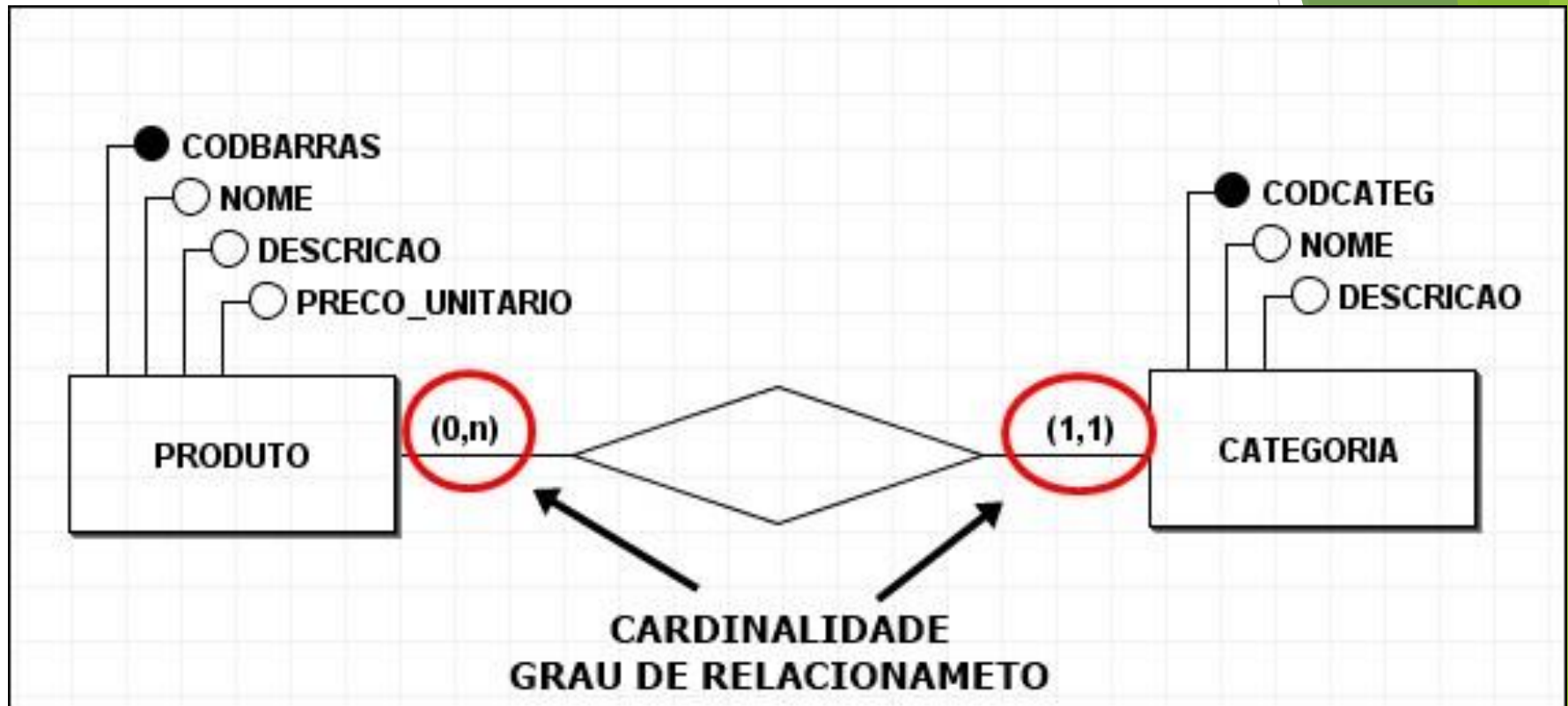
- Relacionamento de Muitos para Muitos (N x N)

Vários elementos de uma entidade A podem se relacionar com vários elementos de uma entidade B e vice-versa.

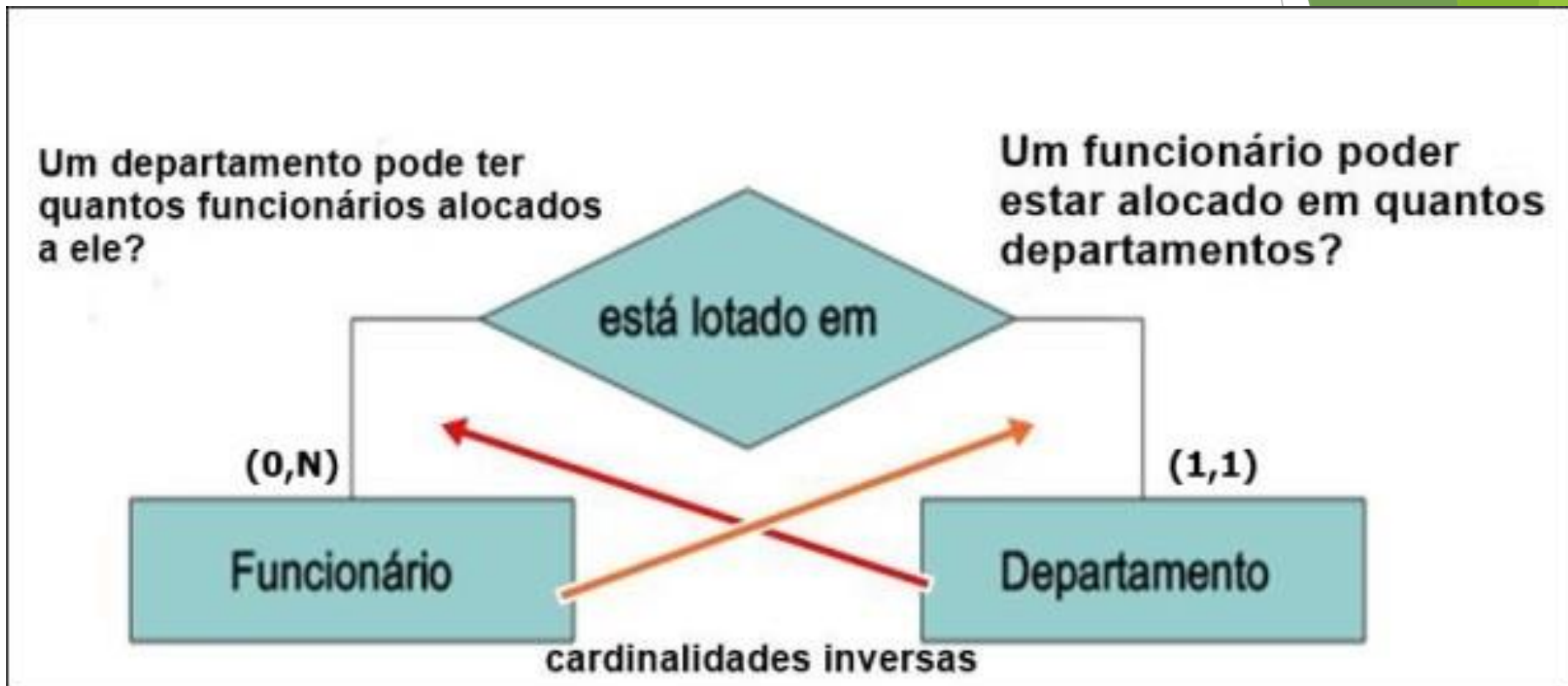
Na modelagem quando ocorre um relacionamento N x N, crie-se uma nova entidade para armazenar dados das entidades que estavam se relacionando inicialmente.



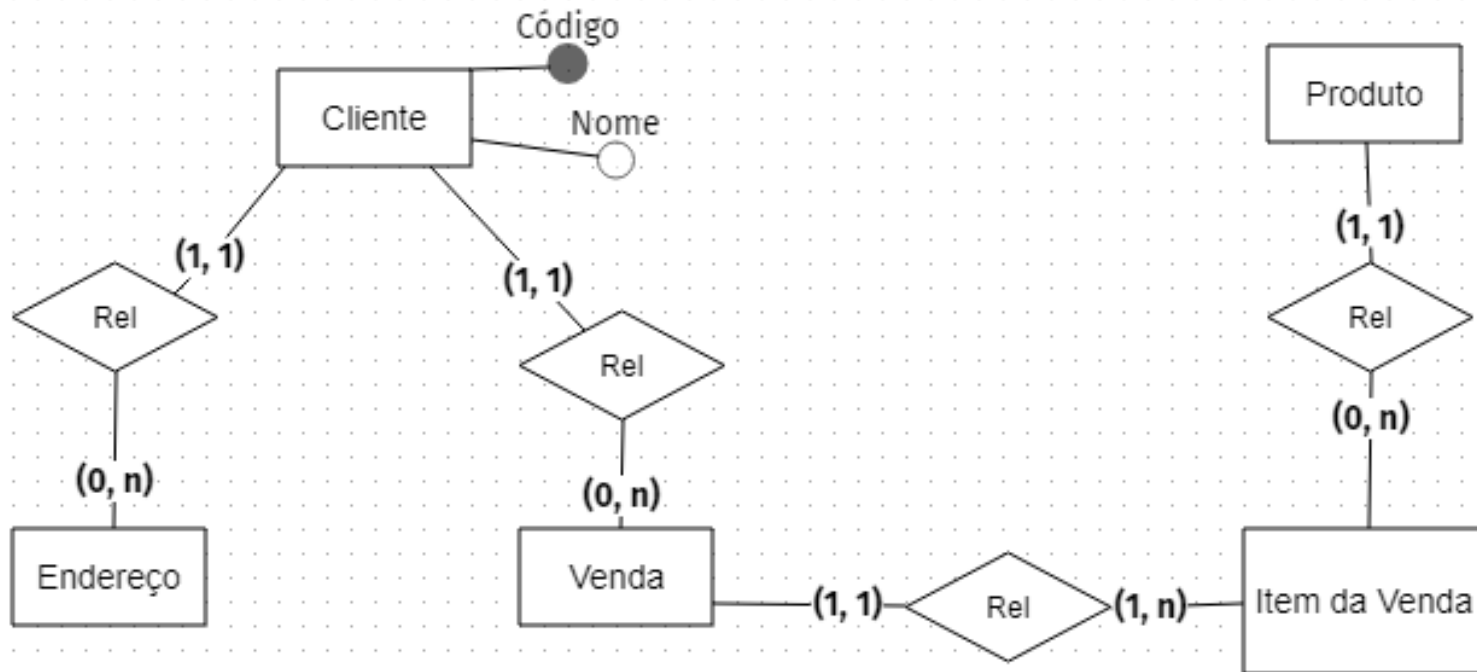
# Cardinalidade



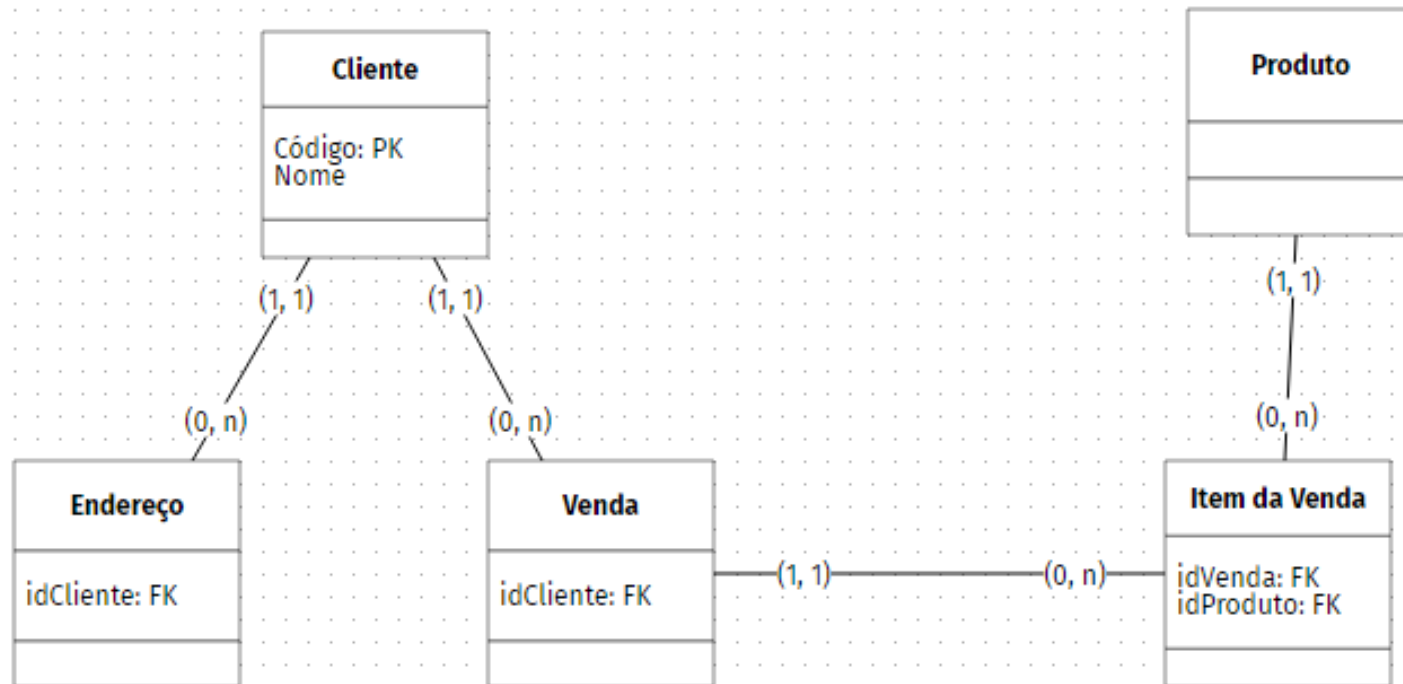
# Cardinalidade



# Como Modelar?



# Como Modelar?



# Vamos modelar?

<https://app.brmodeloweb.com/#!/main>

# Atividade Modelagem 1

- . Modele um sistema de Contas a Pagar
- . Modelo conceitual
- . Modelo lógico

# Atividade Modelagem 2

- . Modele um sistema de Pedidos
- . Modelo conceitual
- . Modelo lógico

# Delphi com Banco de Dados

- . TFDConnection - Componente de acesso ao Banco de Dados  
Normalmente temos apenas um por projeto e fica no DataModule.
- . TFDTable - Componente de acesso a Tabela
- . TDataSource - Componente “Ponte” entre a Tabela (TFDTable) com os componentes visuais (Ex.: TDBEdit ou TDBGrid)



# Delphi com Banco de Dados

The screenshot shows a Delphi application window titled "Pedidos". The form contains several database components and their connections:

- Top Section:** Contains three DBEdit components (DBEdit1, DBEdit2, DBEdit3) for fields "Id", "Número", and "Data Pedido".
- Comprador:** A DBLookupComboBox1 component.
- Fornecedor:** A DBLookupComboBox2 component.
- Items do Pedido:** A table with columns "id Produto", "Descrição Produto", and "Quantidade".

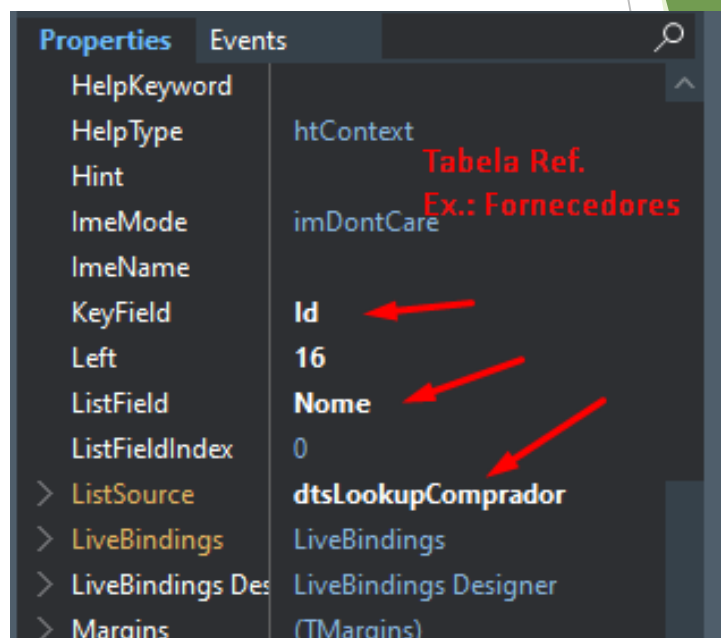
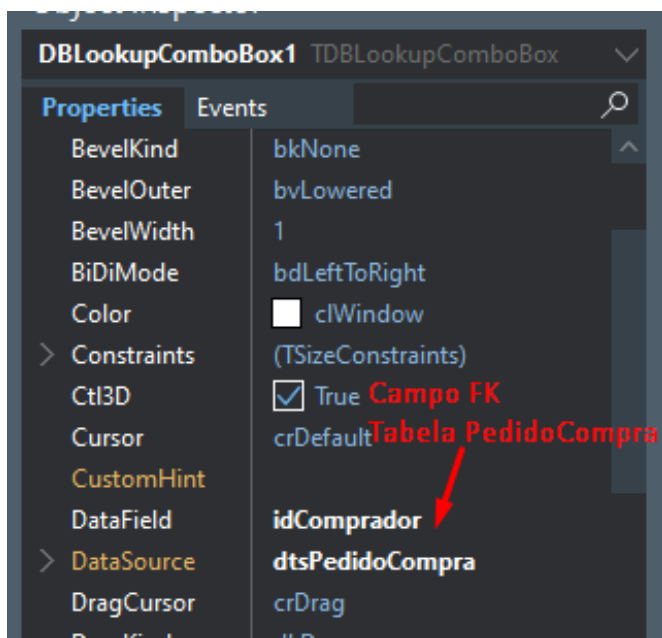
Database connections are shown on the right side of the form:

- 1:** A black box highlights the connection for "dtsPedidoCompra" and "PedidoCompra".
- 2:** A red box highlights the connection for "dtsItemCompra" and "ItemCompra".
- 3:** A blue box highlights the connection for "dtsLookupComprador" and "LookupComprador".
- 4:** A blue box highlights the connection for "dtsLookupFornecedor" and "LookupFornecedor".
- 5:** A purple box highlights the connection for "LookupProdutos".

Navigation buttons (back, forward, etc.) are visible on the left side of the form.

# Delphi com Banco de Dados

Lookup - Busca de Registros em tabelas referenciadas (1 x 1)



# Delphi com Banco de Dados

Lookup - Busca de Registros em tabelas referenciadas (1 x 1)  
(Na Grid)

New Field

Field properties

Name: LookupProduto Component: mCompraLookupProduto2

Type: String Size: 250

Field type

☐ Data ☒ Lookup ☐ Aggregate

☐ Calculated ☐ InternalCalc

Lookup definition

Key Fields: idProduto Dataset: LookupProdutos

Lookup Keys: Id Result Field: Descricao

OK Cancel Help

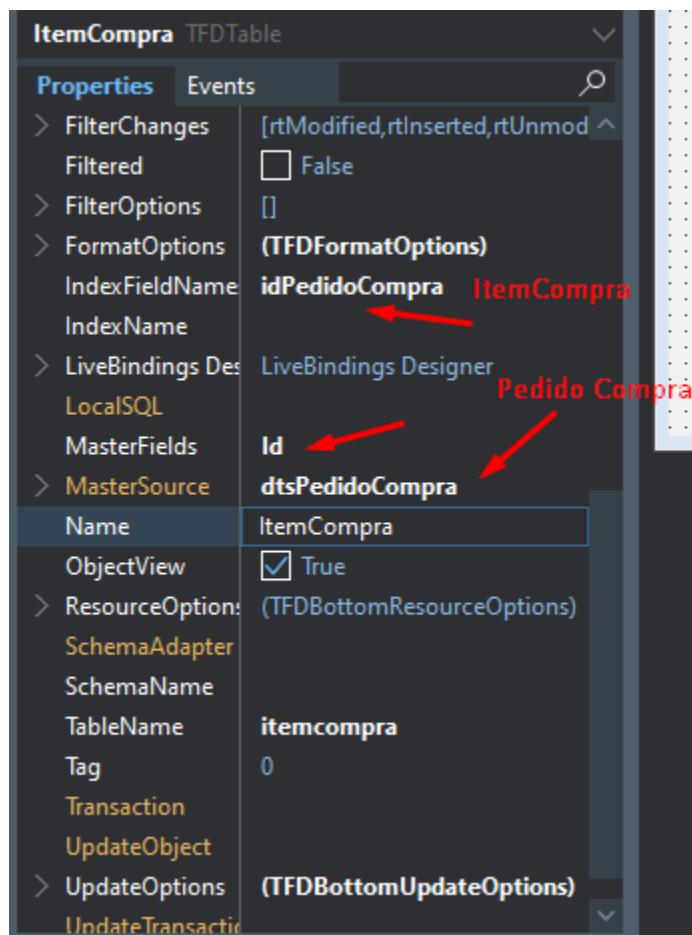
PK nos Produtos

Resultado para Usuário

Campo no ItemCompra

# Delphi com Banco de Dados

Mestre Detalhe (Ligação de uma relacionamento 1 x N) - Pai e Filhos



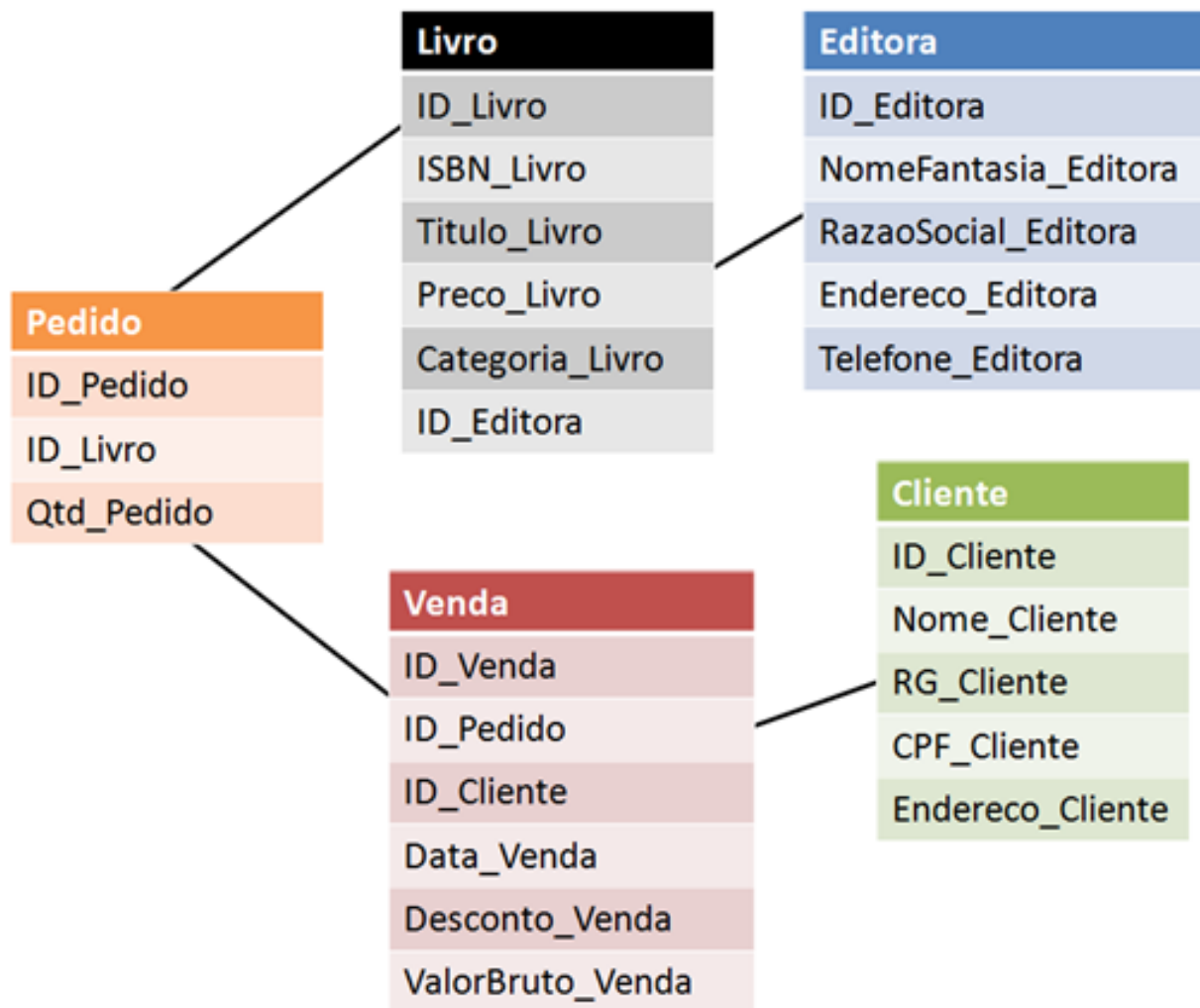
# Tipos de Bancos de Dados

- Relacionais x Não Relacionais

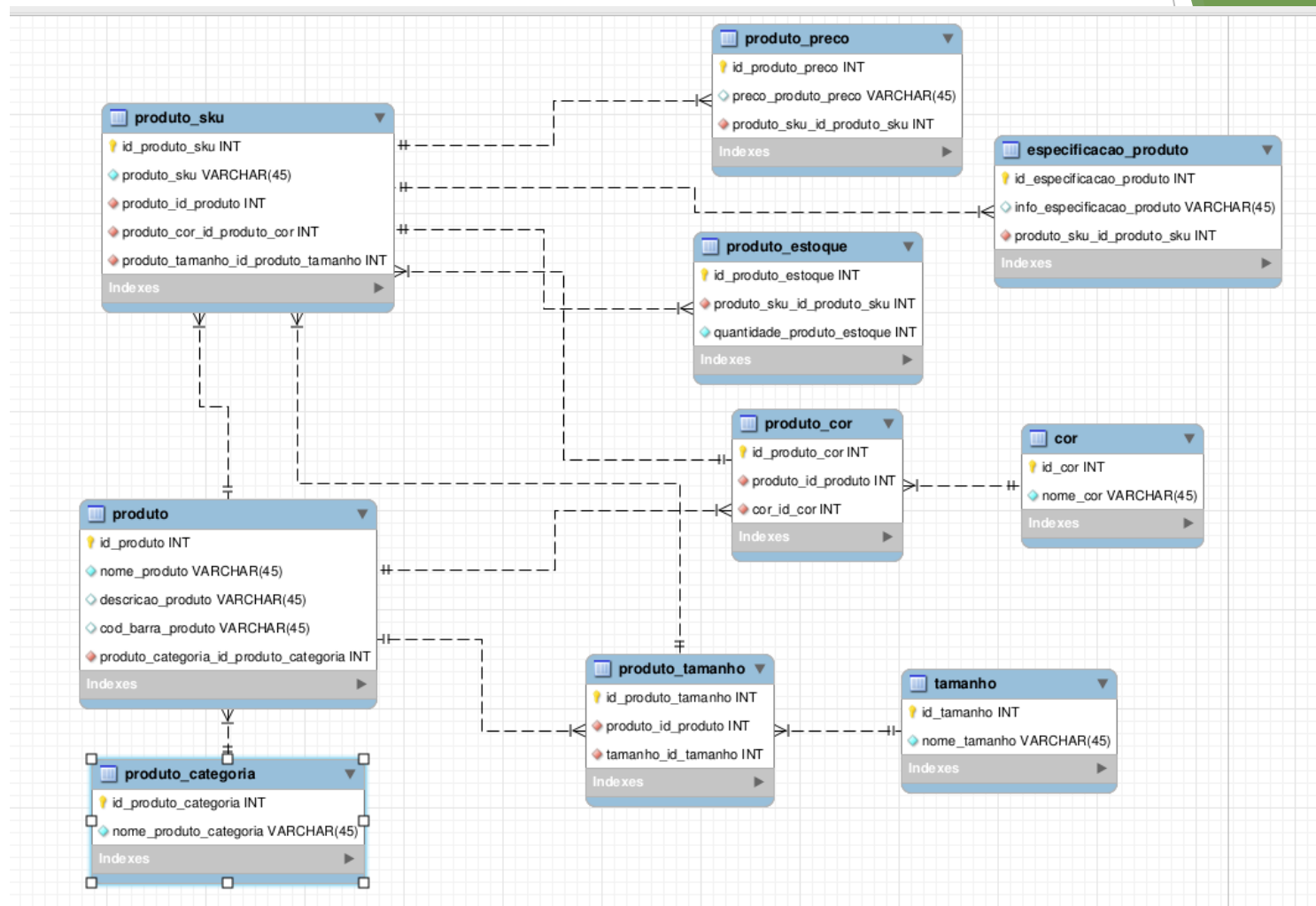
# Tipos de Bancos de Dados

- **Relacionais:** É um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. São baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas. Cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados.

# Tipos de Bancos de Dados



# Tipos de Bancos de Dados





# Tipos de Bancos de Dados

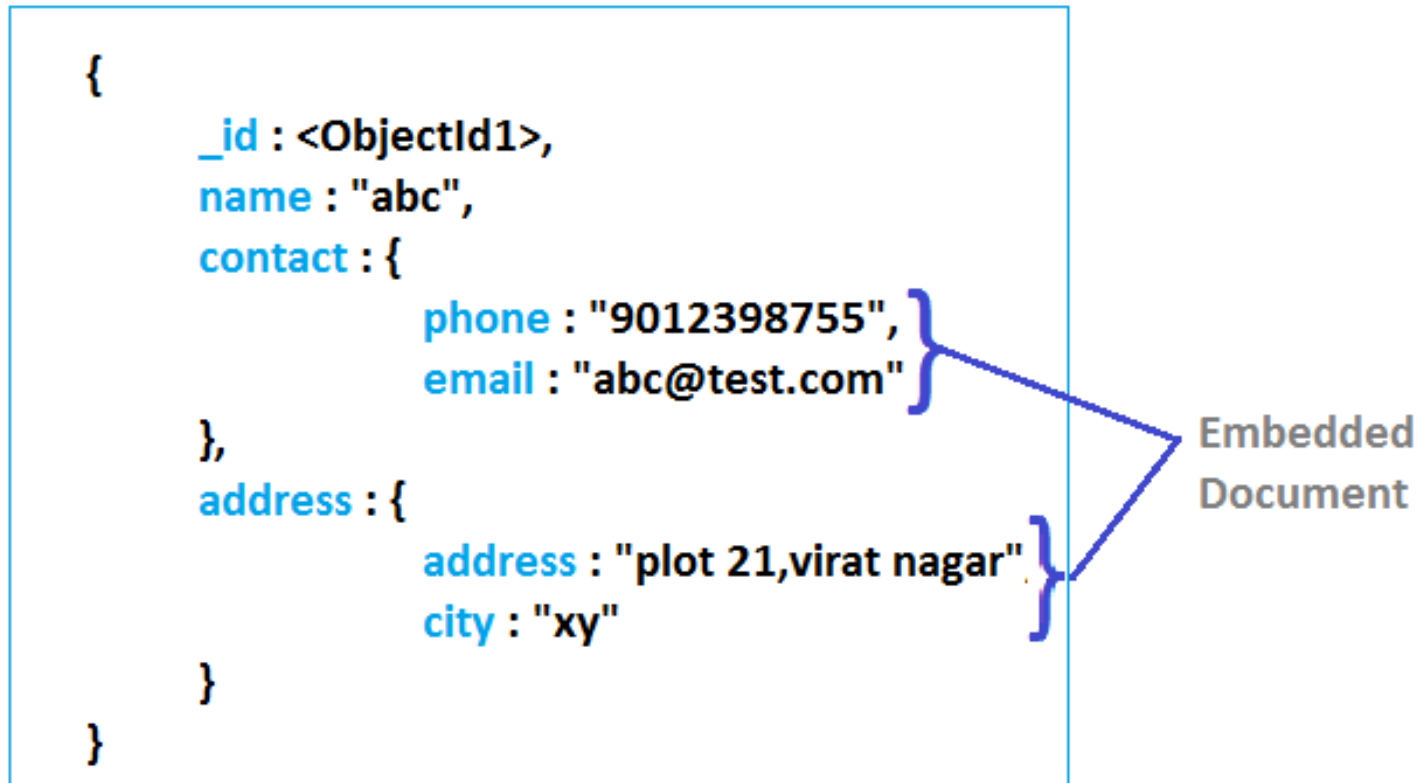
## ► Principais Bancos de Dados Relacionais:



# Tipos de Bancos de Dados

- ▶ **Não Relacionais (NoSQL):** É um banco de dados que não usa o esquema de tabela, linhas e colunas como encontrado nos bancos de dados relacionais (tradicionais). Usam um modelo de armazenamento otimizado para os requisitos específicos do tipo de dados que está sendo armazenado.
- ▶ Por exemplo, os dados podem ser armazenados como pares de chave/valor simples, como um documento JSON (JavaScript Object Notation).

# Tipos de Bancos de Dados



Powered By : [pingax.com](http://pingax.com)

# Tipos de Bancos de Dados

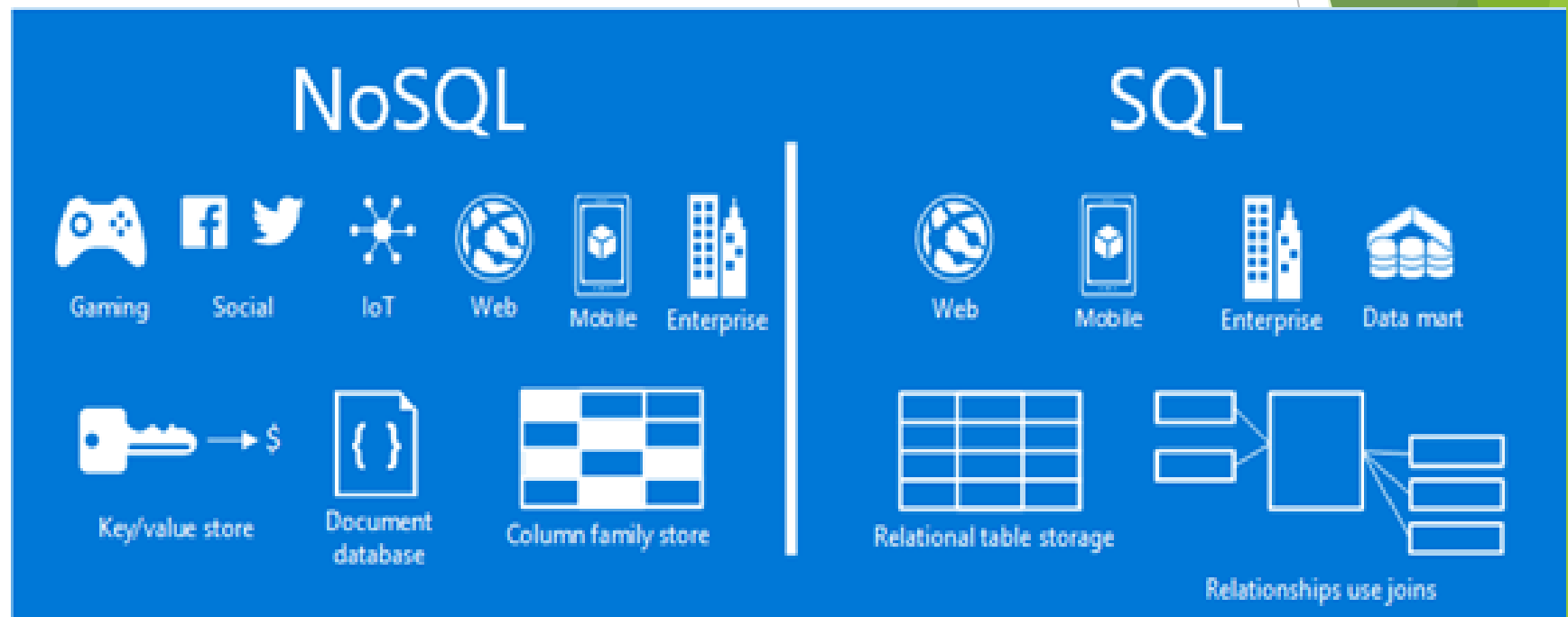
## ► Principais Bancos de Dados NoSQL:



# Tipos de Bancos de Dados

- Mas Armandinho qual tipo de banco vou usar?

# Tipos de Bancos de Dados



# Atividade Final

- ▶ O Sr. Manuel das Couves decidiu automatizar sua oficina mecânica. Hoje ele realiza os orçamentos via planilha Excel e para cada cliente ele tem alguns retrabalhos, fora as informações redundantes que muitas vezes são geradas.
- ▶ Você como Analista Desenvolvedor tem o trabalho de propor uma modelagem e informatizar seus orçamentos via aplicativo de computador.
- ▶ Com base na foto de um orçamento disponibilizado pelo Sr. Manuel crie um modelo conceitual, lógico e um aplicativo funcional para informatização da oficina.

# Atividade Final

Número: 47067 Data da Entrada: 25/10/2022 às \_\_\_\_\_ Prisma: \_\_\_\_\_  
 Cliente: \_\_\_\_\_  
 Endereço: 4122-FRANCIELI MIURI YAMASAKI  
 Complemento: \_\_\_\_\_  
 Cidade: Blumenau UF: SC  
 CEC/CEP: \_\_\_\_\_  
 Número do R.G.: \_\_\_\_\_ Fone: \_\_\_\_\_  
 e-mail: \_\_\_\_\_ Fone Celular: 018 9913 6626

---

Placa: CYL7C92 Ano: 2006/2007  
 Veículo: FIESTA 1,5 8 VALVULAS / PRATA / Et-Combustível / Freio: \_\_\_\_\_  
 Km Atual: 184.276, Chassi: 9BZF16P268469929

---

Peca	Descrição	Qtde	Unit	Vi.Total
02545	PIVO INFERIOR LADO ESQUERDO	1,00	99,00	99,00
03362	PIVO INFERIOR LADO DIREITO	1,00	99,00	99,00
07711	OLEO MOTOR 10 W 40	4,00	26,00	104,00
03062	FILTRO OLEO FORD FIESTA	1,00	26,00	26,00
04828	FILTRO AR FORD MOTOR ZETEC ROCAN	1,00	28,23	28,23
01797	FILTRO COMB. FORD FIESTA	1,00	21,00	21,00
04842	FILTRO CABINE FORD FIESTA/KA	1,00	34,72	34,72
06543	JG PASTILHA FREIO FORD	1,00	94,40	94,40
04706	JUNTA CARTER FORD KA/FIESTA	1,00	98,00	98,00
07451	CARCAÇA VALVULA TERMOSTATICA FORD	1,00	443,20	443,20
1722	AGUA DESMINERALIZADA	3,00	8,41	25,23
0278	ADITIVO RADIADOR	1,00	30,00	30,00
01217	MATERIAL DE LIMPEZA	1,00	40,00	40,00
02018	JG VELAS E IGNICAO	1,00	129,54	129,54
04817	JG CABO VELA	1,00	150,93	150,93
Total de Pecas		20,00		1.423,25

---

Servicos	Qtde	Vi.Unit	Vi.Total
01455 ALINHAMENTO	1,00	60,00	60,00
00004 MECANICA	6,00	100,00	600,00
Total de Servicos	7,00		660,00
Total Geral			2.083,25

AUTORIZO A EXECUCAO DOS SERVICOS ACIMA DISCRIMINADOS  
 \_\_\_\_\_