Comunicação com SGBD

Academia Indiana

O que é SGBD ?

Data Base Management System ou Sistema de Gerenciamento de Banco de Dados (SGBD) é um conjunto de software utilizado para o gerenciamento de uma base de dados, responsáveis por controlar, acessar, organizar e proteger as informações de uma aplicação, tendo como principal objetivo gerenciar as bases de dados utilizadas por aplicações clientes e remover esta responsabilidade das mesmas.

O que é SGBD ?

Alguns SGBDs :

- Oracle
- MySQL
- SQL Server
- MongoDB
- PostgreSQL

Introdução a Banco de Dados

Definindo como os dados serão armazenados no banco de dados, podemos destacar os modelos de SGBDs mais utilizados como os seguintes: Relacionais, não-relacionais (NoSQL), hierárquico, de rede e o orientado a objetos

Introdução a Banco de Dados

Relacionais : Os SGBDS relacionais são banco de dados que modelam os dados no formato de tabelas, que podem se relacionar entre si. Cada tabela pode possuir diversos atributos, com diversos tipos de dados.

Não-relacionais (NoSQL) : NoSQL (Not Only SQL) é o termo utilizado para banco de dados não relacionais de alto desempenho, onde geralmente não é utilizado o SQL como linguagem de consulta. Estes bancos utilizam diversos modelos de dados, incluindo documentos, gráficos, chave-valor e colunares. São amplamente reconhecidos pela facilidade em seu desenvolvimento, desempenho escalável, alta disponibilidade e resiliência.

Hierárquico : Modelo hierárquico de banco de dados consiste em uma coleção de arquivos que são conectados entre si por meio de ligações, baseando a sua base de dados em um modelo de entidades e relacionamentos.

Introdução a Banco de Dados

De rede : Possuindo uma organização semelhante ao modelo de banco de dados hierárquico, o modelo de rede possui uma estrutura mais completa, possuindo como principal diferença entre eles o fato de não existir restrição hierárquica. É um modelo que permite a organização dos dados em uma estrutura formada por várias listas, formada por um conjunto complexo de ligações.

Orientado a objetos : Baseado no paradigma da programação orientada a objetos, neste modelo as funcionalidades de orientação a objetos são integradas aos bancos de dados, onde cada informação é armazenada em forma de um objeto e seus registros em forma de tuplas.

Nesse curso vamos utilizar o MySQL que é Relacional.

Tipos de Dados

Esses são os tipos numéricos do MySQL :

- **TINYINT -** É usado quando precisamos armazenar valores inteiros que se encaixem na faixa de -128 a 127. Este tipo ocupa 1 byte de espaço (8 bits).
- **SMALLINT -** Especifica os números inteiros entre -32768 e 32767. O tipo DECIMAL especifica valores decimais de representação exata e ponto fixo. O número total de dígitos é especificado pela precisão que pode variar entre 0 e 18 (se for omitida, é igual a 18.
- **MEDIUMINT -** É um inteiro de valor medido entre -8388608 e 8388607.
- **INT -** Números inteiros entre -2147483648 e 2147483647.
- **BIGINT -** Deve ser usado quando valores inteiros podem exceder o intervalo ao qual tipo de dados int dá suporte.

Tipos de Dados

Esses são os tipos de datas do MySQL :

- **DATETIME -** A função DATETIME é uma combinação das funções DATE e TIME. DATE retorna o número de série sequencial que representa uma data específica.
- **DATE -** Datas entre 01/Jan/1000 até 31/Dez/9999. Formato padrão: “aaaa-mm-dd”.
- **TIMESTAMP -** No geral, usamos o tipo TIMESTAMP para armazenar informação sobre o momento em que um registro é inserido ou atualizado na tabela. Sempre que uma coluna do tipo TIMESTAMP é alterada, a hora e data atuais são armazenadas, automaticamente.
- **TIME -** Este dado é utilizado para armazenar um range (faixa) de tempo, é exibido pelo MySQL no formato ‘HH:MM:SS[.fração] e seu valor pode estar entre -838:59:59.000000 e 838:59:59.000000.
- **YEAR -** O ano, representado por 4 dígitos. É exibido no formato ‘AAAA’ e pode ter valores entre 1901 e 2155, além do valor 0000. Em versões mais antigas do MySQL ainda tínhamos a opção de YEAR(2), como vocês podem imaginar ele era exibido no formato ‘AA’.

Tipos de Dados

Esses são os tipos de strings do MySQL :

- **CHAR -** Retorna um caracter contido dentro da tabela ASCII, conforme valor informado. Para quem não conhece, existe uma tabela numerada de 1 a 255, uma matriz, que determina todos os caracteres válidos.
- **VARCHAR -** Texto de comprimento variável.

Esses são os tipos de texto do MySQL :

- **TEXT -** Armazenas textos longos ou curtos tenho alguns tipos como : **TINYTEXT, TEXT, MEDIUMTEXT,** e **LONGTEXT**.

Tabelas

Tabelas são objetos de banco de dados que contêm todos os dados em um banco de dados. Nas tabelas, os dados são organizados de maneira lógica em um formato de linha-e-coluna semelhante ao de uma planilha. Cada linha representa um registro exclusivo e cada coluna representa um campo no registro. Através das tabelas podemos acessar e manipular os dados utilizando a linguagem SQL.

Chave Primária

A chave primária, ou Primary key (PK) é o identificador único de um registro na tabela. Pode ser constituída de um campo (chave simples) ou pela combinação de dois ou mais campos (chave composta), de tal maneira que não existam dois registros com o mesmo valor de chave primária.

Essa chave pode ser incrementada, exemplo na tabela aluno teremos com campo id, nome etc. O campo id pode ser gerado automaticamente através do auto-increment.

Chave Estrangeira

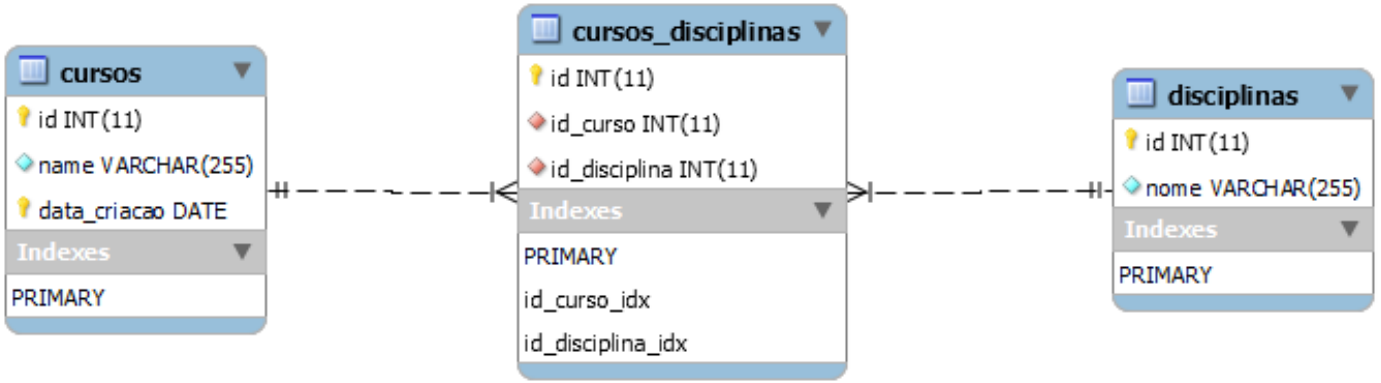
No contexto dos banco de dados, o conceito de chave estrangeira ou chave externa se refere ao tipo de relacionamento entre distintas tabelas de dados do banco de dados. Uma chave estrangeira é chamada quando há o relacionamento entre duas tabelas.

Exemplo : Existe a tabela lojas com os campos id, nome etc… e existe da tabela detalhes_lojas com os campos id, descrição, loja_id(FK) o campo loja_id é uma FK ligada ao id da tabela lojas. Assim posso ter vários detalhes para às lojas.

Índices

Os índices são utilizados para encontrar registros com um valor específico de uma coluna rapidamente. Sem um índice o MySQL tem de iniciar com o primeiro registro e depois ler através de toda a tabela até que ele encontre os registros relevantes. Quanto maior a tabela, maior será o custo.

MER


cursos
id INT(11)
name VARCHAR(255)
data_criacao DATE
Indexes
PRIMARY
cursos_disciplinas
id INT(11)
id_curso INT(11)
id_disciplina INT(11)
Indexes
PRIMARY
id_curso_idx
id_disciplina_idx
disciplinas
id INT(11)
nome VARCHAR(255)
Indexes
PRIMARY

Overview da ferramenta de SGDB

Um banco de dados é uma coleção organizada de dados bem estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (Data Base Management System - DBMS).

Nesse curso vamos trabalhar com o DBMS:

- HeidiSQL

HeidiSQL

HeidiSQL é uma excelente ferramenta para gerenciar uma base de dados MySQL e uma alternativa séria para aplicativos como phpMyAdmin.

Ele traz uma interface gráfica completa para gerenciar tabelas, registros e usuários de uma base de dados MySQL.

Além de ser gratuito.

Criação de bancos de dados

Após conectar no seu gerenciador de banco de dados através do HeidiSQL você pode rodar esse comando para criar uma base de dados ou criar visualmente pela interface do HeidiSQL. Nesse exemplo estamos criando um banco de dados de nome cursos.

CREATE DATABASE universidades;

Para usar o banco de dados e criar tabelas nele você pode usar o comando :

USE universidades;

Criação de tabelas

SQL para se criar uma tabela de cursos com uma Primary Key:

```
CREATE TABLE cursos (
    id int NOT NULL AUTO_INCREMENT,
    nome varchar(255) NOT NULL,
    data_criacao DATE,
    PRIMARY KEY (Id)
);
```

Criação de tabelas

SQL para se criar uma tabela de disciplinas com uma Primary Key:

```
CREATE TABLE disciplinas (
    id int NOT NULL AUTO_INCREMENT,
    nome varchar(255) NOT NULL,
    PRIMARY KEY (Id)
);
```

Criação de tabelas

SQL para se criar uma tabela de cursos_disciplinas com uma Primary Key e Foreign Keys:

```
CREATE TABLE cursos_disciplinas (
    id int NOT NULL AUTO_INCREMENT,
    id_curso int,
    id_disciplina int,
    PRIMARY KEY (Id),
    FOREIGN KEY (id_curso) REFERENCES cursos(id),
    FOREIGN KEY (id_disciplina) REFERENCES disciplinas(id)
);
```

Criação de colunas

As colunas podem ser criadas quando você usa o create table ou adicionadas depois que a tabela for criada. Também é possível adicioná-las visualmente através do seu gerenciador de banco de dados.

- ALTER TABLE cursos ADD carga_horaria varchar(255);

Criação de chaves

Você pode criar as chaves diretamente através do create table ou posteriormente dessa forma :

Criando primary key :

ALTER TABLE lojas ADD column `id` int(10) unsigned primary KEY AUTO_INCREMENT;

Criando foreign key :

ALTER TABLE cursos_disciplinas ADD COLUMN `id_disciplina` INT;

ALTER TABLE cursos_disciplinas ADD CONSTRAINT FK_Disciplinas FOREIGN KEY (id_disciplina) REFERENCES disciplinas(id);

Inserção de dados

Código para inserir registros no banco de dados :

```
INSERT INTO cursos (nome, data_criacao) VALUES ('Sistemas de Informação', '2023-07-10');
INSERT INTO cursos (nome, data_criacao) VALUES ('Ciência da Computação', '2023-07-11');
INSERT INTO disciplinas (nome) VALUES ('Lógica de Programação');
INSERT INTO disciplinas (nome) VALUES ('Matemática');
INSERT INTO cursos_disciplinas (id_curso, id_disciplina) VALUES (1, 1);
INSERT INTO cursos_disciplinas (id_curso, id_disciplina) VALUES (1, 2);
INSERT INTO cursos_disciplinas (id_curso, id_disciplina) VALUES (2, 1);
INSERT INTO cursos_disciplinas (id_curso, id_disciplina) VALUES (2, 2);
```

Seleção

Vamos rodar alguns selects para entender seu funcionamento :

Lista todos os campos e cursos da tabela curso :
SELECT * FROM cursos;

Lista apenas o nome de todos cursos:
SELECT nome FROM cursos;

Outras seleções :
SELECT * FROM disciplinas;
SELECT * FROM cursos_disciplinas WHERE id_curso = 2;

Ordenação

Seleciona todas disciplinas e ordena por id em ordem crescente :

SELECT * FROM disciplinas ORDER BY id ASC;

Seleciona todas disciplinas e ordena por id em ordem decrescente:

SELECT * FROM disciplinas ORDER BY id DESC;

Agrupamento

Insira mais uma disciplinas com o nome de Lógica de programação :

INSERT INTO disciplinas (nome) VALUES ('Lógica de Programação');

Esse select retorna duas disciplinas cadastradas, apesar de ter três, porque ele agrupou pelo nome e temos duas disciplinas com o nome Lógica de programação :

SELECT nome FROM disciplinas GROUP BY nome;

Junção de tabelas

LEFT JOIN : Retorna todas as linhas da tabela à esquerda, mesmo se não houver nenhuma correspondência na tabela à direita.

```
SELECT cursos_disciplinas.id_curso, cursos.nome AS nome_curso, cursos_disciplinas.id_disciplina,
disciplinas.nome AS nome_disciplina FROM cursos_disciplinas
LEFT JOIN cursos ON cursos.id = cursos_disciplinas.id_curso
LEFT JOIN disciplinas ON disciplinas.id = cursos_disciplinas.id_disciplina;
```

Junção de tabelas

RIGHT JOIN: Retorna todas as linhas da tabela à direita, mesmo se não houver nenhuma correspondência na tabela à esquerda.

```
SELECT cursos_disciplinas.id_curso, cursos.nome AS nome_curso, cursos_disciplinas.id_disciplina,
disciplinas.nome AS nome_disciplina FROM cursos_disciplinas
RIGHT JOIN cursos ON cursos.id = cursos_disciplinas.id_curso
RIGHT JOIN disciplinas ON disciplinas.id = cursos_disciplinas.id_disciplina;
```

Junção de tabelas

UNION ALL: O operador UNION ALL combina os resultados de duas ou mais queries, retornando todas as linhas pertencentes a todas as queries envolvidas na execução.

SELECT nome FROM cursos UNION ALL SELECT nome FROM disciplinas;

Update

Essa é a estrutura de código para rodar uma atualização no banco :

UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;

Altere dados da tabela cursos e disciplinas para entendimento do update.

Delete

Essa é a estrutura de código deletar um registro no banco :

DELETE FROM table_name WHERE condition;

Delete dados da tabela cursos e disciplinas para entendimento do delete.

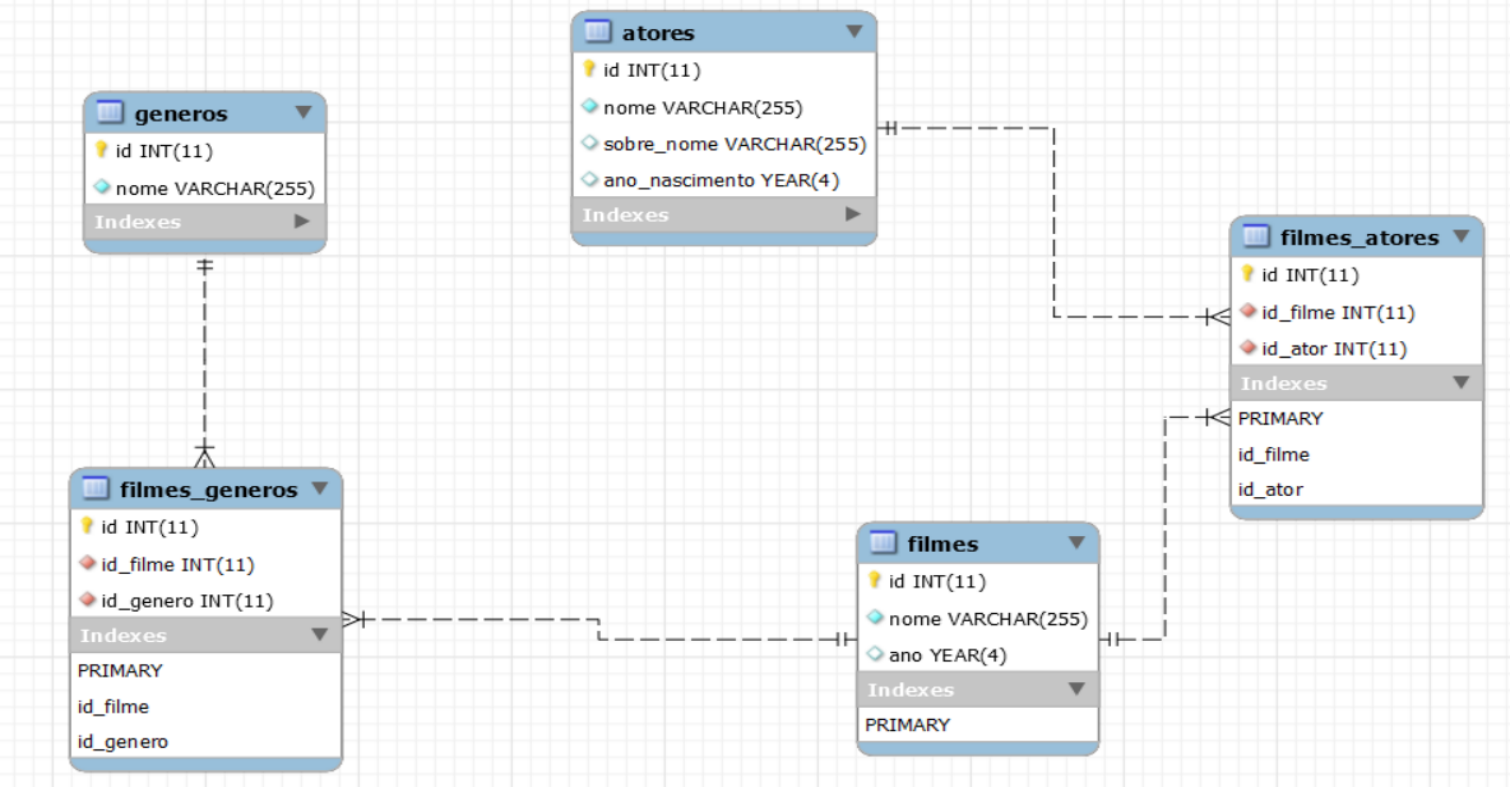
Exercício

Vamos pensar como seria o relacionamento entre essas tabelas, no caso estamos modelando um Estúdio de filmes.

atores, filmes, generos, filmes_generos, filmes_atores

- Busca, ao ir digitando o nome, retorna os filmes.
- Retornar filmes por genero Ficção Cientifica ou outro.
- Retornar todos os filmes e gênero do filme.
- Retornar todos os filmes e gênero do filme, ordenado pelo nome da tabela genero.

MER

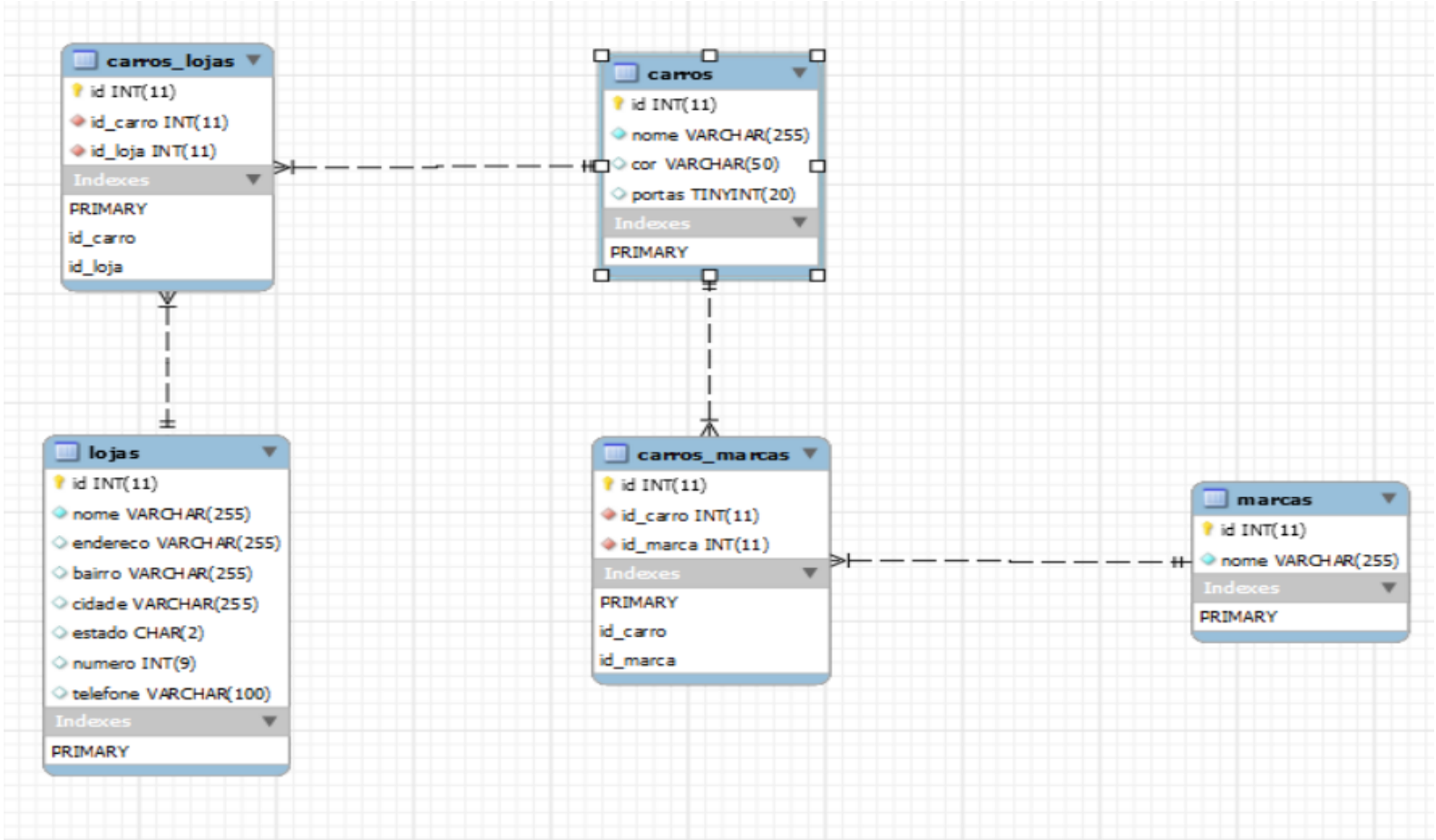

atores
id INT(11)
nome VARCHAR(255)
sobre_nome VARCHAR(255)
ano_nascimento YEAR(4)
Indexes
generos
id INT(11)
nome VARCHAR(255)
Indexes
filmes_atores
id INT(11)
id_filme INT(11)
id_ator INT(11)
Indexes
PRIMARY
id_filme
id_ator
filmes_generos
id INT(11)
id_filme INT(11)
id_genero INT(11)
Indexes
PRIMARY
id_filme
id_genero
filmes
id INT(11)
nome VARCHAR(255)
ano YEAR(4)
Indexes
PRIMARY

Exercício

Vamos pensar como seria o relacionamento entre essas tabelas, no caso estamos modelando uma Concessionária.

lojas, carros, marcas, carros_marcas, carros_lojas

- Busca, ao ir digitando o nome do carro, retorna os carros.
- Busca, ao ir digitando o nome do marca, retorna os carros.
- Retornar carros de determinada loja
- Retornar marca de determinado carro


carros_lojas
id INT(11)
id_carro INT(11)
id_loja INT(11)
Indexes
PRIMARY
id_carro
id_loja
carros
id INT(11)
nome VARCHAR(255)
cor VARCHAR(50)
portas TINYINT(20)
Indexes
PRIMARY
lojas
id INT(11)
nome VARCHAR(255)
endereco VARCHAR(255)
bairro VARCHAR(255)
cidade VARCHAR(255)
estado CHAR(2)
numero INT(9)
telefone VARCHAR(100)
Indexes
PRIMARY
carros_marcas
id INT(11)
id_carro INT(11)
id_marca INT(11)
Indexes
PRIMARY
id_carro
id_marca
marcas
id INT(11)
nome VARCHAR(255)
Indexes
PRIMARY

Exercício

Vamos pensar como seria o MER e a criação do SQL de um banco de dados, no caso somos uma empresa de recursos humanos que administra diversas empresas e seus funcionários.

Preciso ter os endereços das empresas e de seus funcionários e também preciso ter os cargos de cada funcionário.

Conectando ao MySQL com PDO

O PDO (PHP Data Objects) define uma interface de conexão a banco de dados leve e consistente para PHP. Há a possibilidade de utilização de diversos drivers de conexão que implementam a interface do PDO para vários tipos de bancos de dados.

Como o PDO representa uma camada de abstração de acesso aos dados, as mesmas funções utilizadas para manipular dados ou recuperar informações do banco serão as mesmas, independentemente do banco de dados que esteja sendo usado.

Nessa aula vamos demonstrar uma conexão PDO com PHP.