



High performance. Delivered.

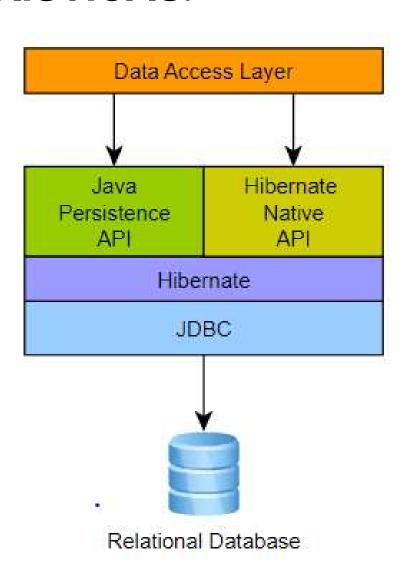
Application Delivery Fundamentals: SPRING BOOT

Module 8: ORM

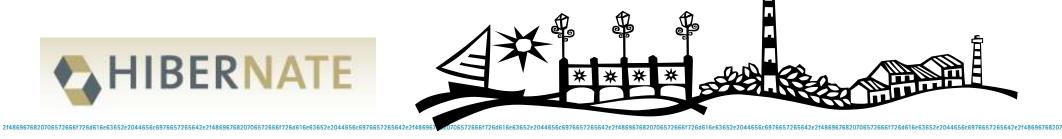




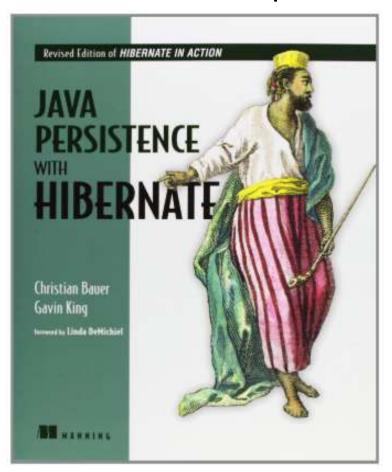
### • CARACTERISTICAS:







- DICA DE LIVRO:
- Java Persistence with Hibernate: Revised Edition of Hibernate in Action Paperback – November 24, 2006;









- Como iremos mapear o mundo relacional com o mundo Orientado a objeto?
- No mundo relacional os dados estão em forma de tabelas que contém linhas e colunas.
- No mundo OO os dados estão em forma de objetos.

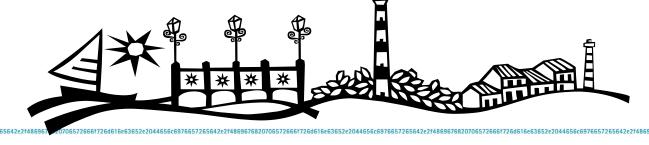




### Object Relational Mapping - ORM - Possui:

- Uma API para montar o CRUD;
- Uma query para especificar as classes e suas propriedades;
- Um Local para especificação dos metadados;
- Uma técnica para interagir com objetos relacionais para executar otimizações, recuperação de dados, etc..
- Proteje o desenvolvedor de SQL confusos;
- O Desenvolvedor n\u00e3o precisa entender de BD relacionais;





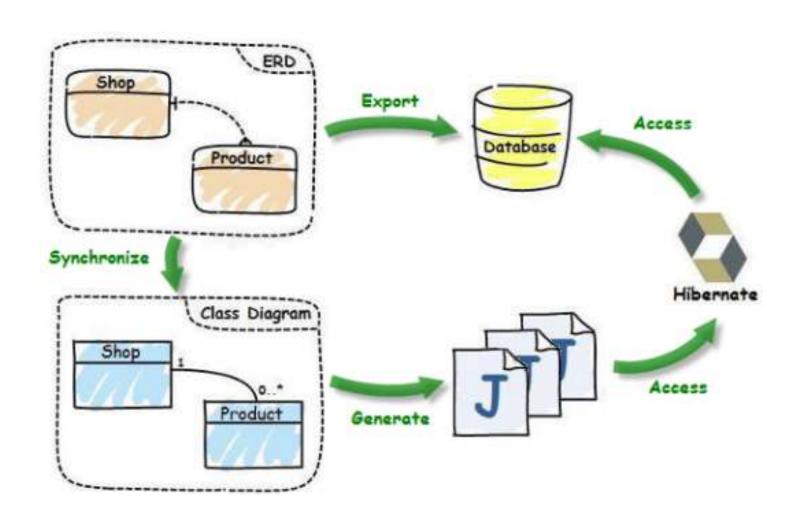
# O que é Mapeamento Objeto-Relacional?



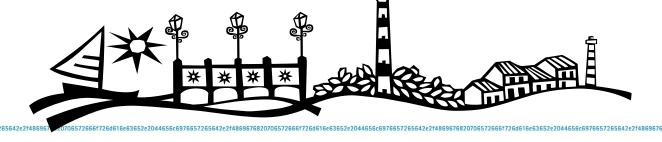
 ORM é uma técnica que consiste da conversão das classes da aplicação para tabelas do banco de dados e vice-versa.

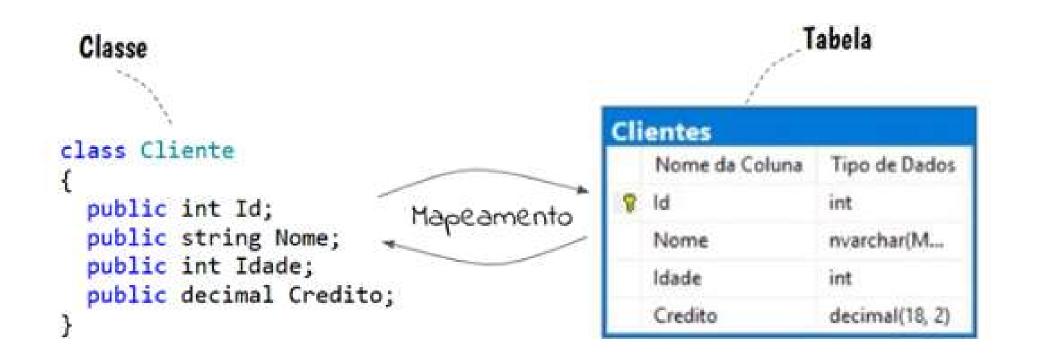
















 Da mesma forma, também faz parte desse processo a conversão entre os objetos da aplicação e as linhas da tabela Ou seja, enquanto no código trabalhamos com classes e objetos, esses são gravados e recuperados do banco de dados na forma de registros/linhas.



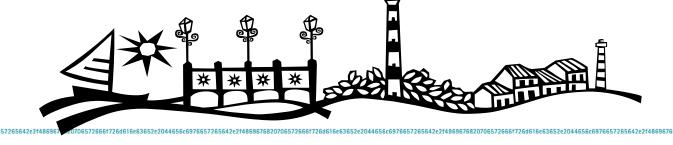




### Associação?

- Entidades podem ser associadas à outras entidades;
- Ou Collections;
- One-to-one, one-to-many, many-to-one, e many-to-many;
- Tecnicas de carregamento Lazy e Eager;

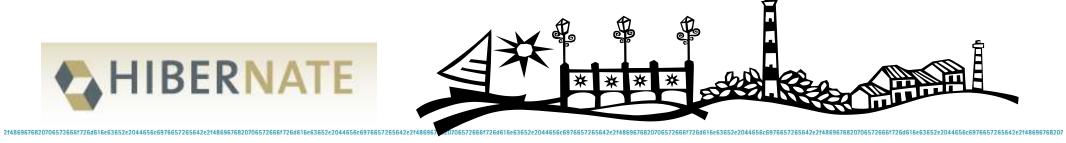




### Annotations

- https://hibernate.org/
  - @Entity
  - @Table
  - @Column (length = 100)
  - @Id @GeneratedValue(strategy = GenerationType.AUTO)
  - OneToOne(fetch = FetchType.LAZY, mappedBy = "stock", cascade = CascadeType.ALL)
  - @OneToMany
  - @ManyToOne
  - @ManyToMany
  - @Inheritance
  - @PreUpdate
  - @PrePersist





Estratégia	Descrição
GenerationType.AUTO	Valor padrão, deixa com o provedor de persistência a escolha da estratégia mais adequada de acordo com o banco de dados.
GenerationType.IDENTITY	Informamos ao provedor de persistência que os valores a serem atribuídos ao identificador único serão gerados pela coluna de auto incremento do banco de dados. Assim, um valor para o identificador é gerado para cada registro inserido no banco. Alguns bancos de dados podem não suportar essa opção.
GenerationType.SEQUENCE	Informamos ao provedor de persistência que os valores serão gerados a partir de uma sequence. Caso não seja especificado um nome para a sequence, será utilizada uma sequence padrão, a qual será global, para todas as entidades. Caso uma sequence seja especificada, o provedor passará a adotar essa sequence para criação das chaves primárias. Alguns bancos de dados podem não suportar essa opção.
GenerationType.TABLE  Copyright © 2022 Accenture All Rights Reserved.	Com a opção TABLE é necessário criar uma tabela para gerenciar as chaves primárias. Por causa da sobrecarga de consultas necessárias para manter a tabela atualizada, essa opção é pouco recomendada.

### **Hibernate**

# Aplicação One To One

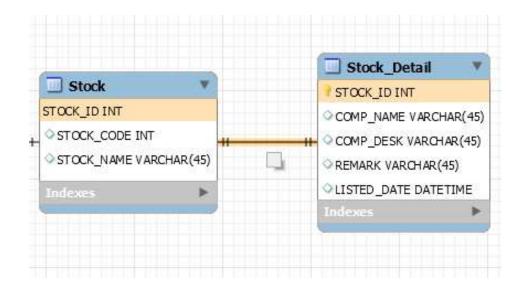
2468696768207065726661726661665765365262044656c697665726564262748696768707065726661726661666365262044656c697665726564262748696768207065726661726661666365262044656c6976657265642627486967682070657266617266772666172666172666172666172666172666172666172666172667726661726661726677266617266772666772666772666772666772667726667726





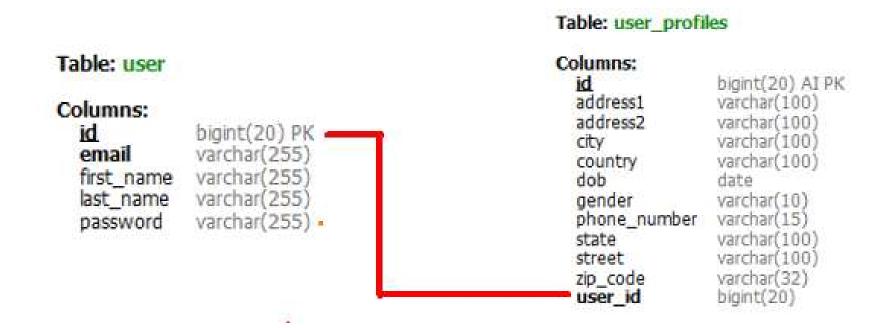
### Annotations

— @OneToOne(fetch = FetchType.LAZY, mappedBy = "stock", cascade = CascadeType.ALL)



- @OneToOne(fetch = FetchType.LAZY)
- @PrimaryKeyJoinColumn

### Relacionamento user vs user\_profile



### **Mapeamento**

```
9 @Entity
10 @Table(name = "users")
11 public class User implements Serializable {
12⊖
       @Id
       @GeneratedValue(strategy = GenerationType. IDENTITY)
13
14
       private Long id;
       @OneToOne(fetch = FetchType. LAZY,
35⊖
36
               cascade = CascadeType.ALL,
               mappedBy = "user")
37
       private UserProfile userProfile;
38
               @OneToOne(fetch = FetchType.LAZY, optional = false)
        49⊖
               @JoinColumn(name = "user id", nullable = false)
        50
               private User user;
        51
        52
```

#### Table: user

#### Columns:

email bigint(20) PK varchar(255) first\_name varchar(255) last\_name varchar(255) password varchar(255)

#### Table: user\_profiles

#### Columns:

bigint(20) AI PK address1 varchar(100) address2 varchar(100) varchar(100) city country varchar(100) dob date gender varchar(10) phone\_number varchar(15) varchar(100) state street varchar(100) zip code varchar(32) user id bigint(20)

# Aplicação One To Many

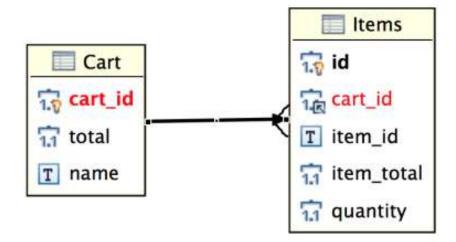
2468696768207065726661726661665765365262044656c697665726564262748696768707065726661726661666365262044656c697665726564262748696768207065726661726661666365262044656c6976657265642627486967682070657266617266772666172666172666172666172666172666172666172666172667726661726661726677266617266772666772666772666772666772667726667726





### One-to-many

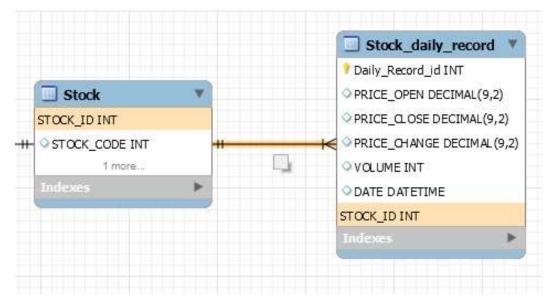
— @OneToMany(mappedBy="cart")



- @ManyToOne
- @JoinColumn(name="cart\_id", nullable=false)

### One-to-many:

— @OneToMany(fetch = FetchType.LAZY, mappedBy = "stock")



- @ManyToOne(fetch = FetchType.LAZY)
- @JoinColumn(name="STOCK\_ID", nullable=false)

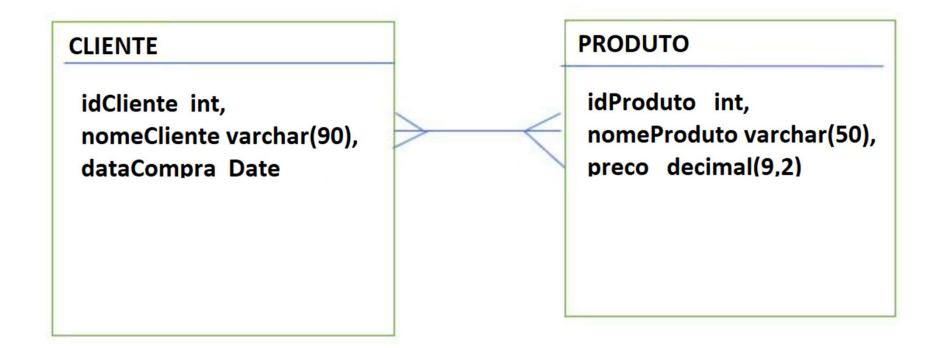
# Aplicação Many To Many

2468696768207065726661726661665765365262044656c697665726564262748696768707065726661726661666365262044656c697665726564262748696768207065726661726661666365262044656c6976657265642627486967682070657266617266772666172666172666172666172666172666172666172666172667726661726661726677266617266772666772666772666772666772667726667726





### Many-to-many:



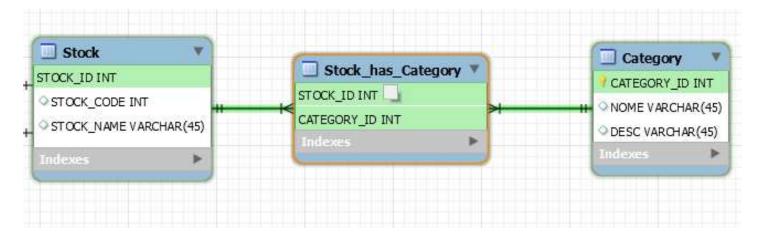




```
Funcionario.java
       BEntity
       @Table (name="funcionario")
       public class Funcionario implements Serializable (
 10
           Bld
 11
           @Column (name="id")
           @GeneratedValue(strategy = GenerationType.AUTO, generator = "funcionario seg gen")
           @SequenceGenerator(name = "funcionario seq gen", sequenceName = "funcionario id seq")
 13
 14
           private Long id;
 15
           private String nome;
 17
           @ManyToMany
           @JoinTable(name = "depto funcionario", joinColumns = (@JoinColumn(name = "id funcionario")),
 18
           inverseJoinColumns =(@JoinColumn(name = "id depto")))
 19
 20
           private List (Depto) (deptos;)
 21
                                                               Depto.java
    public List < Depto > getDeptos () (
 23
               return deptos;
 24
                                          7
                                               @Entity
                                               @Table (name="depto")
                                          9
                                               public class Depto implements Serializable {
                                         10
                                                    RId
                                         11
                                                    @Column(name="id")
                                         12
                                                    @GeneratedValue(strategy = GenerationType.AUTO, generator = "depto seq gen"
uso do mapped By
                                         13
                                                    @SequenceGenerator(name = "depto seq gen", sequenceName = "depto id seq")
                                          14
                                                    private Long id;
                                          15
                                          16
                                                    private String nome;
                                          13
                                                    @ManyToMany(mappedBy = "deptos")
                                         18
                                         19
                                         20
                                                    private List (Funcionario) funcionarios;
                                         21
                                          22
                                                    public List<Funcionario> getFuncionarios() (
                                          23
                                                        return funcionarios:
                                                                                                                                 22
```

### Annotations

- @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
- **@JoinTable**(name = "stock\_category", catalog = "mydb", joinColumns = {
- @JoinColumn(name = "STOCK\_ID", nullable = false, updatable = false) },
- inverseJoinColumns = {
- @JoinColumn(name = "CATEGORY\_ID", nullable = false, updatable = false) })



— @ManyToMany(fetch = FetchType.LAZY, mappedBy = "categories")





## **Dúvidas?**



