FIAP GRADUAÇÃO

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#07 – MAPEAMENTO AVANÇADO

 thiagoyama@gmail.com  thiagoyama

TRAJETÓRIA

- JPA Introdução
- JPA API
- Design Patterns e JUnit
- Relacionamentos
- JPQL
- Mapeamento Avançado

#07 - AGENDA

- Chave Composta
- Múltiplas Tabelas
- Herança entre Entidades

CHAVE COMPOSTA

CHAVE COMPOSTA

- Tomemos o modelo de **reserva de pacotes** abaixo;
- Nele, podemos verificar a existência de uma tabela associativa **T_RESERVA** que possui uma **chave primária** composta por: **CD_RESERVA**, **CD_CLIENTE** e **CD_PACOTE**;
- Além, disso, existe um campo **NR_DIAS**, próprio da tabela e, portanto, neste caso não podemos utilizar a anotação **@ManyToMany**;

CHAVE COMPOSTA

Para mapear **chaves compostas** você deve:

1. Criar uma classe que conterá apenas os **atributos correspondentes** aos atributos da **chave estrangeira;**
2. Na entidade em si utilizar a anotação **@IdClass** para indicar a classe criada em 1;
3. Criar na entidade os **mesmos atributos** que a classe de id (criada em 1) com anotações **@Id**;
4. Para os atributos com anotações @Id que também forem chaves estrangeiras utilizar as anotações **@ManyToOne** e **@JoinColumn** para indicar a coluna de chave estrangeira;

CLASSE DE CHAVE PRIMÁRIA

```
public class ReservaPK implements Serializable {
        private int id;
        private int cliente;
        private int pacote;
        ...
}
```

Mesmo **nome dos atributos** da Entidade, note que para o cliente e pacote o tipo de dado é **int**, para armazenar somente a chave e não o objeto inteiro;

Gere os métodos **equals**() e **hashCode**(), para determinar se dois objetos são iguais pelos valores dos três atributos (id, cliente e pacote) e não pelo endereço de memória;

CHAVE COMPOSTA - ENTIDADE

```
@Entity
@Table(name="TB_RESERVA")
@IdClass(ReservaPK.class)
public class Reserva implements Serializable {

    @Id
    @SequenceGenerator(name="seqReserva", allocationSize=1)
    @GeneratedValue(generator="seqReserva",strategy=GenerationType.SEQUENCE)
    @Column(name="CD_RESERVA")
    private int id;

    @Column(name="NR_DIAS")
    private int numeroDias;
    ...
}
```

CHAVE COMPOSTA - ENTIDADE

- Os atributos que fazem parte tanto da chave estrangeira quanto da primária ficam assim:

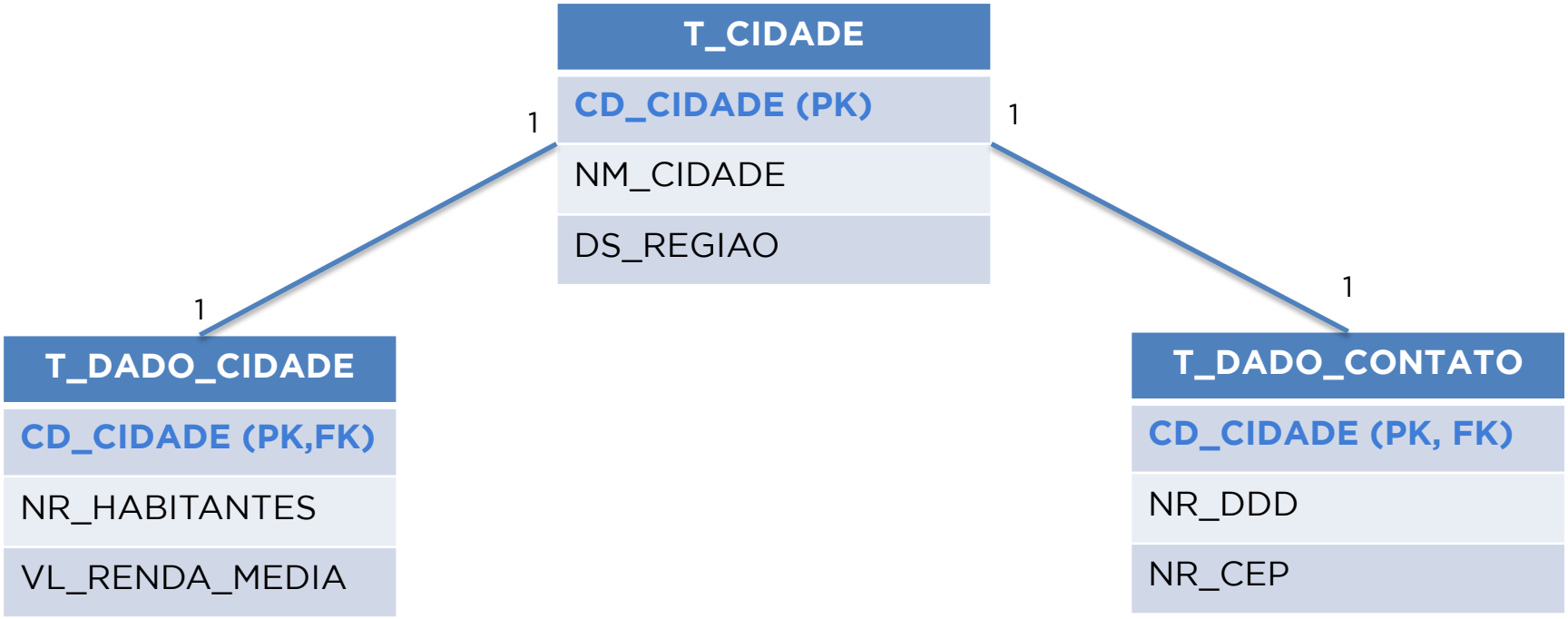
```
@Id
@ManyToOne
@JoinColumn(name="CD_CLIENTE")
private Cliente cliente;

@Id
@ManyToOne
@JoinColumn(name="CD_PACOTE")
private Pacote pacote;
```

MÚLTIPLAS TABELAS

MÚLTIPLAS TABELAS

- Uma **única entidade** pode ser representada por **mais de uma tabela** no banco de dados:



Observe que **CD_CIDADE** é **PK** e **FK**

MÚLTIPLAS TABELAS

FIAP

1. Criar **uma entidade** com **todos os atributos** de todas as tabelas;
2. A entidade deve possuir a anotação **@SecondaryTable** ou **@SecondaryTables** , com os parâmetros:
   - **name**: nome da tabela secundária
   - **pkJoinColumns**: colunas envolvidas na chave primária:
     - **@PrimaryKeyJoinColumn**: nome da coluna de chave primária
3. Cada atributo da tabela secundária anotadas com **@Column** deverão apontar para a tabela secundária por meio do parâmetro **table**.

```
@SecondaryTable(name="T_DADO_CIDADE",
pkJoinColumns={@PrimaryKeyJoinColumn(name="CD_CIDADE")})
public class Cidade {
        @Column(name="NR_HABITANTES", table="T_DADO_CIDADE")
        private int totalHabitantes;
}
```

MÚLTIPLAS TABELAS

- Existindo mais de uma tabela secundária, basta utilizar a anotação **@SecondaryTables**:

```
@SecondaryTables(value={
        @SecondaryTable(name="TAB_DADO_CIDADE",
        pkJoinColumns={@PrimaryKeyJoinColumn(name="COD_CIDADE")})
        ,
        @SecondaryTable(name="TAB_DADO_CONTATO",
        pkJoinColumns={@PrimaryKeyJoinColumn(name="COD_CIDADE")})
})
public class Cidade {

        @Column(name="NUM_HABITANTES", table="TAB_DADO_CIDADE")
        private int totalHabitantes;

        @Column(name="NUM_DDD", table="TAB_DADO_CONTATO")
        private int ddd;
}
```

MÚLTIPLAS TABELAS

- Ao persistir uma entidade Cidade parte dos dados será armazenado na **TAB_DADO_CIDADE** e parte na **TAB_DADO_CONTATO**:

```
Cidade c = new Cidade();
c.setNome("BAURU");
c.setDdd(14);
c.setTotalHabitantes(4500000);

... persist(c);
```

T_CIDADE	
CD_CIDADE	NM_CIDADE
1	BAURU

T_DADO_CIDADE		
CD_CIDADE	NR_HABITANTES	...
1	450000	...

T_DADO_CONTATO		
CD_CIDADE	NR_DDD	...
1	14	...

HERANÇA

HERANÇA

- A **herança** entre classes pode ser também **mapeada** para o modelo de **dados**;

- Para tanto existe uma anotação **@Inheritance** que define, **na classe pai**, a estratégia de mapeamento a ser utilizada (parâmetro **strategy**);

- Existem três estratégias possíveis (definidas no *enum* **InheritanceType**);

HERANÇA - ESTRATÉGIAS

- **SINGLE_TABLE: Uma única tabela para toda a hierarquia da herança;**
  - Mais eficiente;
  - Pode produzir uma tabela com muitos campos;

- **JOINED: Uma tabela separada para cada subclasse;**
  - Necessita de uma operação de *join* entre as tabelas pai e filho;
  - Cada tabela terá apenas campos específicos da subclasse representada;

- **TABLE_PER_CLASS: Uma única tabela para cada classe concreta;**
  - Repete-se em cada tabela os atributos da classe filho e pai;
  - Das três estratégias é a menos utilizada (não estudaremos);

HERANÇA - ESTRATÉGIAS

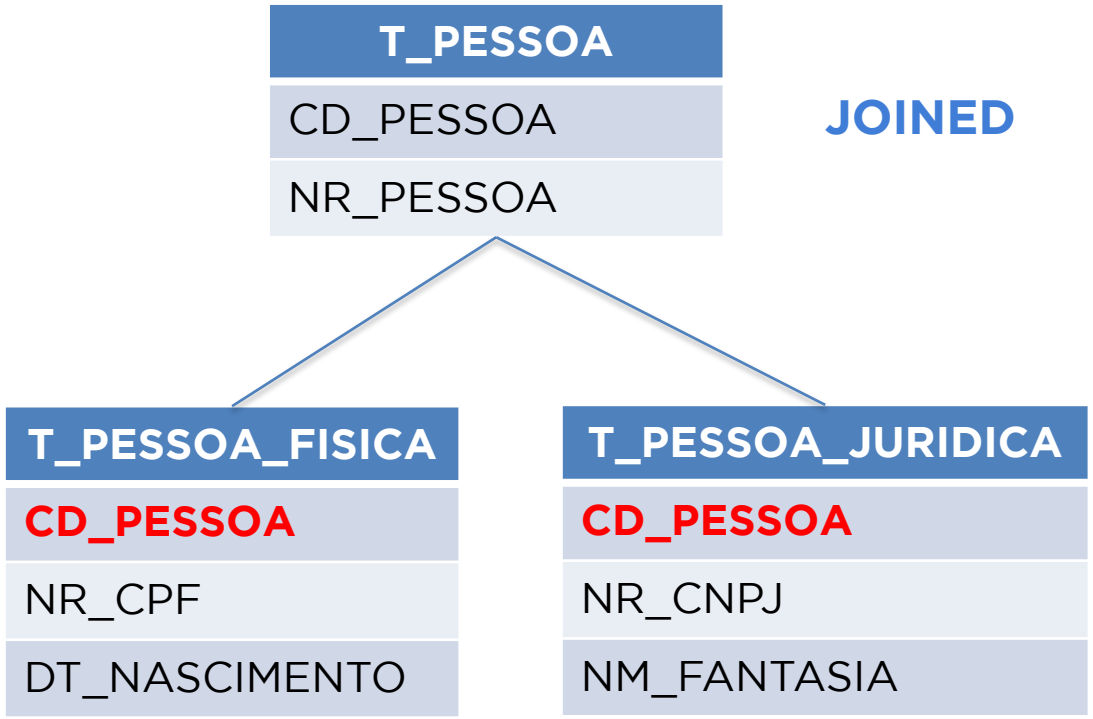
- Observe os exemplos de **SINGLE_TABLE** e **JOINED:**

SINGLE_TABLE

T_PESSOA
CD_PESSOA
NR_PESSOA
NR_CPF
NR_CNPJ
DT_NASCIMENTO
NM_FANTASIA
DTYPE

DTYPE (“Pessoa”, “PessoaFisica”, “PessoaJuridica”)

JOINED



HERANÇA – SINGLE_TABLE

FIAP

- Na classe pai definimos o tipo da **estratégia**:
  - **@Inheritance(strategy = InheritanceType.*SINGLE_TABLE*)**

- Na entidade pai pode existir uma **anotação** indicando o campo na tabela que identificará o tipo de classe representada pelo registro:
  - **@DiscriminatorColumn**
    - **name**: nome da coluna;
    - **discriminatorType**: tipo de dado da coluna;

- Nas entidades (pai e filhas) pode existir uma anotação para indicar qual o **valor** que deve assumir para representar a entidade:
  - **@DiscriminatorValue**
    - **value**: valor que tipifica a entidade no banco de dados

SINGLE_TABLE - EXEMPLO

FIAP

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="DS_TIPO")
@DiscriminatorValue("Pessoa")
public class Pessoa {
   //...
}
```

```
@Entity
@DiscriminatorValue(value="PF")
public class PessoaFisica extends Pessoa {
  //...
}
```

HERANÇA – JOINED

- As **tabelas filhas** devem possui uma **chave primária** que também será chave estrangeira, apontando para a chave primária da tabela pai;
- Nas entidades filhas pode existir uma anotação para indicar o **nome** campo de chave primária na tabela pai:
  - **@PrimaryKeyJoinColumn**
    - **name**: nome do campo de chave primária da tabela pai;

```
@Entity
@Table(name="T_PESSOA_FISICA")
@PrimaryKeyJoinColumn(name="CD_PESSOA")
public class PessoaFisica extends Pessoa {
    //...
}
```

VOCÊ APRENDEU..

- Mapear **chave composta** de uma tabela, criando uma classe com os atributos da chave e mapeando essa classe na tabela com **@IdClass** e adicionando **@Id** em todos os atributos chave da entidade;

- Mapear múltiplas tabelas de uma única entidade com a anotação **@SecondaryTable**;

- Trabalhar com herança de classes utilizando as estratégias **Single Table** e **Joined**;

FIAP

"O sucesso normalmente vem para quem está ocupado demais para pensar nele"