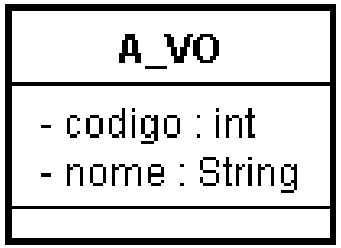
Mapeamento Objeto/Relacional JPA/Hibernate

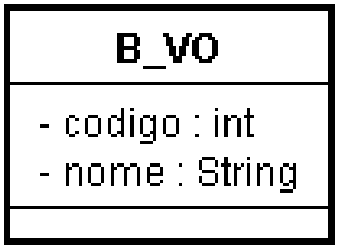
Evandro César Freiberger

evandrofreiberger@gmail.com

Mapeamento de Relações

Entidades que não se relacionam não geram relação no BD.





Mapeamento de Relações

```
@Entity
@Table(name = "a")

public class A_VO {

   @Id
   @GeneratedValue(strategy=GenerationType.SEQUENCE)
   private int codigo;

   @Column(length=40, nullable=false)
   private String nome;

}
```

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

Mapeamento de Relações

```
@Entity
@Table(name = "b")
public class B_VO {

   @Id
   @GeneratedValue(strategy=GenerationType.SEQUENCE)
   private int codigo;

   @Column(length=40, nullable=false)
   private String nome;

}
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

Mapeamento de Relações

- ✓ B_VO referencia uma instância de A_VO
- ✓ B_VO agrega uma instância de A_VO
- ✓ A_VO não referencia B_VO

Mapeamento de Relações

```java
@Entity
@Table(name = "a")
public class A_VO {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;

    @Column(length=40, nullable=false)
    private String nome;
}
```

```cpp
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

Mapeamento de Relações

```java
@Entity
@Table(name = "b")
public class B_VO {

   @Id
   @GeneratedValue(strategy=GenerationType.SEQUENCE)
   private int codigo;

   @Column(length=40, nullable=false)
   private String nome;

   @OneToOne(fetch=FetchType.EAGER)
   private A_VO a_VO;
}
```

```sql
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  a_vo_codigo integer,
  CONSTRAINT b_pkey PRIMARY KEY (codigo),
  CONSTRAINT fk62e6f1c60d FOREIGN KEY (a_vo_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
6   public class Principal1 {
7       public static void main(String args[]) {
8           try {
9               EntityManager entityManager = FabricaEntityManager.getEntityManager();
10
11              A_VO a = new A_VO();
12              a.setNome("a");
13
14              entityManager.getTransaction().begin();
15              entityManager.persist(a);
16              entityManager.getTransaction().commit();
17          } catch (PersistenciaException ex) {
18              System.out.println(ex.toString());
19          }
20      }
21  }
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
8   public class Principal2 {
9       public static void main(String args[]) {
10          try {
11              EntityManager entityManager = FabricaEntityManager.getEntityManager();
12              entityManager.getTransaction().begin();
13              Query query = entityManager.createQuery("SELECT a FROM A_VO a WHERE a.nome = 'a' ");
14              A_VO a = (A_VO) query.getSingleResult();
15
16              B_VO b = new B_VO();
17              b.setNome("b");
18              b.setA_VO(a);
19
20              entityManager.persist(b);
21              entityManager.getTransaction().commit();
22          } catch (PersistenciaException ex) {
23              System.out.println(ex.toString());
24          }
25      }
26  }
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
7   public class Principal3 {
8       public static void main(String args[]) {
9           try {
10              EntityManager entityManager = FabricaEntityManager.getEntityManager();
11              entityManager.getTransaction().begin();
12              Query query = entityManager.createQuery("SELECT b FROM B_VO b WHERE b.nome = 'b' ");
13              B_VO b = (B_VO) query.getSingleResult();
14
15              if(b != null){
16                  System.out.println("Codigo B: "+b.getCodigo());
17                  System.out.println("Nome B: "+b.getNome());
18                  System.out.println("Codigo A: "+b.getA_VO().getCodigo());
19                  System.out.println("Nome A: "+b.getA_VO().getNome());
20              }
21              entityManager.getTransaction().commit();
22          } catch (PersistenciaException ex) {
23              System.out.println(ex.toString());
24          }
25      }
26  }
```

Mapeamento de Relações

- ✓ Muitas instâncias de B_VO referenciam uma instâncias de A_VO
- ✓ Cada B_VO agrega uma instância de A_VO
- ✓ A_VO não referencia B_VO

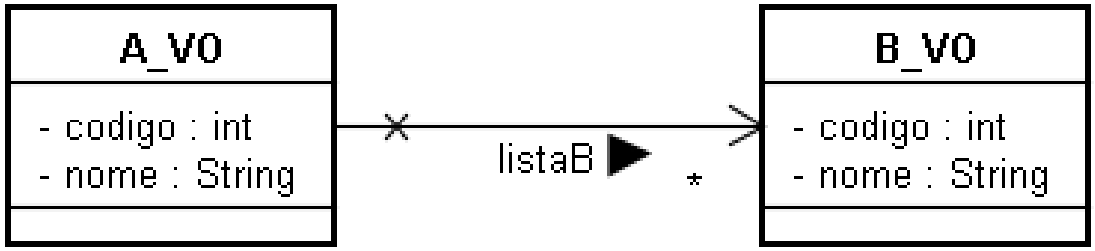
Mapeamento de Relações

```java
@Entity
@Table(name = "b")
public class B_VO {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;

    @Column(length=40, nullable=false)
    private String nome;

    @ManyToOne(fetch=FetchType.EAGER)
    private A_VO a_VO;
}
```

```sql
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  a_vo_codigo integer,
  CONSTRAINT b_pkey PRIMARY KEY (codigo),
  CONSTRAINT fk62e6f1c60d FOREIGN KEY (a_vo_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Relações

- ✓ Uma instância de A_VO referencia uma lista de instâncias de B_VO
- ✓ Cada A_VO agrega N instâncias de B_VO
- ✓ B_VO não referencia A_VO



Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;
    @Column(length=40, nullable=false)
    private String nome;

    @OneToMany(fetch=FetchType.LAZY)
    private Collection<B_VO> listaB;
}
```

Mapeado com a estrutura Collection

```
@Entity
@Table(name = "b")
public class B_VO {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;

    @Column(length=40, nullable=false)
    private String nome;
}
```

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE a_b
(
  a_codigo integer NOT NULL,
  listab_codigo integer NOT NULL,
  CONSTRAINT fk17804c0244384 FOREIGN KEY (listab_codigo)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17804cf2d71c3 FOREIGN KEY (a_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT a_b_listab_codigo_key UNIQUE (listab_codigo)
)
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
6   public class Principal1 {
7       public static void main(String args[]) {
8           try {
9               EntityManager entityManager = FabricaEntityManager.getEntityManager();
10              entityManager.getTransaction().begin();
11              B_VO b = new B_VO();
12              b.setNome("b1");
13              entityManager.persist(b);
14              b = new B_VO();
15              b.setNome("b2");
16              entityManager.persist(b);
17              b = new B_VO();
18              b.setNome("b3");
19              entityManager.persist(b);
20              entityManager.getTransaction().commit();
21          } catch (PersistenciaException ex) {
22              System.out.println(ex.toString());
23          }
24      }
25  }
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
9   public class Principal2 {
10      public static void main(String args[]) {
11          try {
12              EntityManager entityManager = FabricaEntityManager.getEntityManager();
13              entityManager.getTransaction().begin();
14              A_VO a = new A_VO();
15              a.setNome("a1");
16
17              List<B_VO> colecaoB;
18              Query query = entityManager.createQuery("SELECT b FROM B_VO b ");
19              colecaoB = query.getResultList();
20              a.setListaB(colecaoB);
21
22              entityManager.persist(a);
23              entityManager.getTransaction().commit();
24          } catch (PersistenciaException ex) {
25              System.out.println(ex.toString());
26          }
27      }
28  }
```

Mapeamento de Relações

```
1   package aplicacao;
2   import ...
8   public class Principal3 {
9       public static void main(String args[]) {
10          try {
11              EntityManager entityManager = FabricaEntityManager.getEntityManager();
12              entityManager.getTransaction().begin();
13              Query query = entityManager.createQuery("SELECT a FROM A_VO a WHERE a.nome = 'a1' ");
14              A_VO a = (A_VO) query.getSingleResult();
15              if (a != null) {
16                  System.out.println("Codigo A: " + a.getCodigo());
17                  System.out.println("Nome A: " + a.getNome());
18                  for (B_VO b : a.getListaB()) {
19                      System.out.println("Codigo B: " + b.getCodigo());
20                      System.out.println("Nome B: " + b.getNome());
21                      System.out.println("-------------------------------");
22                  }
23              }
24              entityManager.getTransaction().commit();
25          } catch (PersistenciaException ex) {
26              System.out.println(ex.toString());
27          }
28      }
29  }
```

Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO implements Serializable {
   @Id
   @GeneratedValue(strategy = GenerationType.SEQUENCE)
   private int codigo;

   @Column(length = 40, nullable = false)
   private String nome;

   @OneToMany(fetch= FetchType.LAZY)
   @JoinColumn(name="codigo_a")
   private Collection<B_VO> listaB;
}
```

Uso do @JoinColumn para evitar a tabela de relacionamento a_b

```
@Entity
@Table(name = "b")
public class B_VO implements Serializable {

   @Id
   @GeneratedValue(strategy = GenerationType.SEQUENCE)
   private int codigo;

   @Column(length = 40, nullable = false)
   private String nome;
}
```

Cria uma chave estrangeira no lado N da relação

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  codigo_a integer,
  CONSTRAINT b_pkey PRIMARY KEY (codigo),
  CONSTRAINT fk_b_codigo_a FOREIGN KEY (codigo_a)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Não representa uma relação onde B_VO não referencia A_VO.

Cada B_VO consegue referenciar apenas um A_VO

Mapeamento de Relações

```
try {
    EntityManager entityManager = FabricaEntityManager.getEntityManager();
    entityManager.getTransaction().begin();
    B_VO b = new B_VO();
    b.setNome("b1");
    entityManager.persist(b);
    b = new B_VO();
    b.setNome("b2");
    entityManager.persist(b);
    b = new B_VO();
    b.setNome("b3");
    entityManager.persist(b);
    entityManager.getTransaction().commit();
} catch (PersistenciaException ex) {
    System.out.println(ex.toString());
}
```

	codigo [PK] integer	nome character var	codigo_a integer
1	2851	b1	
2	2852	b2	
3	2853	b3	
*			

Mapeamento de Relações

```
try {
    EntityManager entityManager = FabricaEntityManager.getEntityManager();
    entityManager.getTransaction().begin();
    A_VO a = new A_VO();
    a.setNome("a1");

    List<B_VO> colecaoB;
    Query query = entityManager.createQuery("SELECT b FROM B_VO b ");
    colecaoB = query.getResultList();
    a.setListaB(colecaoB);

    entityManager.persist(a);
    entityManager.getTransaction().commit();
} catch (PersistenciaException ex) {
    System.out.println(ex.toString());
}
```

	codigo [PK] integer	nome character var
1	2951	a1
*		

	codigo [PK] integer	nome character var	codigo_a integer
1	2851	b1	2951
2	2852	b2	2951
3	2853	b3	2951
*			

Mapeamento de Relações

```
try {
    EntityManager entityManager = FabricaEntityManager.getEntityManager();
    entityManager.getTransaction().begin();
    A_VO a = new A_VO();
    a.setNome("a2");

    List<B_VO> colecaoB;
    Query query = entityManager.createQuery("SELECT b FROM B_VO b ");
    colecaoB = query.getResultList();
    a.setListaB(colecaoB);

    entityManager.persist(a);
    entityManager.getTransaction().commit();
} catch (PersistenciaException ex) {
    System.out.println(ex.toString());
}
```

	codigo [PK] integer	nome character var
1	2951	a1
2	3051	a2
*		

	codigo [PK] integer	nome character var	codigo_a integer
1	2851	b1	3051
2	2852	b2	3051
3	2853	b3	3051
*			

Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;
    @Column(length=40, nullable=false)
    private String nome;

    @OneToMany(fetch=FetchType.LAZY)
    private Set<B_VO> listaB;
}
```

Mapeado com a estrutura Set

```
@Entity
@Table(name = "b")
public class B_VO {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;

    @Column(length=40, nullable=false)
    private String nome;
}
```

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE a_b
(
  a_codigo integer NOT NULL,
  listab_codigo integer NOT NULL,
  CONSTRAINT a_b_pkey PRIMARY KEY (a_codigo, listab_codigo),
  CONSTRAINT fk17804c0244384 FOREIGN KEY (listab_codigo)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17804cf2d71c3 FOREIGN KEY (a_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT a_b_listab_codigo_key UNIQUE (listab_codigo)
)
```

Mapeamento de Relações

```
6   public class Principal1 {
7       public static void main(String args[]) {
8           try {
9               EntityManager entityManager = FabricaEntityManager.getEntityManager();
10              entityManager.getTransaction().begin();
11              B_VO b = new B_VO();
12              b.setNome("b1");
13              entityManager.persist(b);
14              b = new B_VO();
15              b.setNome("b2");
16              entityManager.persist(b);
17              b = new B_VO();
18              b.setNome("b3");
19              entityManager.persist(b);
20              entityManager.getTransaction().commit();
21          } catch (PersistenciaException ex) {
22              System.out.println(ex.toString());
23          }
24      }
25  }
```

Mapeamento de Relações

```
10  public class Principal2 {
11      public static void main(String args[]) {
12          try {
13              EntityManager entityManager = FabricaEntityManager.getEntityManager();
14              entityManager.getTransaction().begin();
15              A_VO a = new A_VO();
16              a.setNome("a1");
17
18              List<B_VO> colecaoB;
19              Query query = entityManager.createQuery("SELECT b FROM B_VO b ");
20              colecaoB = query.getResultList();
21              a.setConjuntoB(new HashSet(colecaoB));
22              entityManager.persist(a);
23              entityManager.getTransaction().commit();
24          } catch (PersistenciaException ex) {
25              System.out.println(ex.toString());
26          }
27      }
28  }
```

Mapeamento de Relações

```
9   public class Principal3 {
10      public static void main(String args[]) {
11          try {
12              EntityManager entityManager = FabricaEntityManager.getEntityManager();
13              entityManager.getTransaction().begin();
14              Query query = entityManager.createQuery("SELECT a FROM A_VO a WHERE a.nome = 'a1' ");
15              A_VO a = (A_VO) query.getSingleResult();
16              if (a != null) {
17                  System.out.println("Codigo A: " + a.getCodigo());
18                  System.out.println("Nome A: " + a.getNome());
19                  for (B_VO b : a.getConjuntoB()) {
20                      System.out.println("Codigo B: " + b.getCodigo());
21                      System.out.println("Nome B: " + b.getNome());
22                      System.out.println("------------------------------");
23                  }
24              }
25              entityManager.getTransaction().commit();
26          } catch (PersistenciaException ex) {
27              System.out.println(ex.toString());
28          }
29      }
30  }
```

Mapeamento de Relações

- ✓ Muitas instâncias de A_VO referenciam muitas instâncias de B_VO
- ✓ B_VO não referencia A_VO

Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;
    @Column(length=40, nullable=false)
    private String nome;

    @ManyToMany(fetch=FetchType.LAZY)
    private Set<B_VO> listaB;
}
```

```
@Entity
@Table(name = "b")
public class B_VO {

    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;

    @Column(length=40, nullable=false)
    private String nome;
}
```

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE a_b
(
  a_codigo integer NOT NULL,
  listab_codigo integer NOT NULL,
  CONSTRAINT a_b_pkey PRIMARY KEY (a_codigo, listab_codigo),
  CONSTRAINT fk17804c0244384 FOREIGN KEY (listab_codigo)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17804cf2d71c3 FOREIGN KEY (a_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Relações

- ✓ Muitas instâncias de A_VO referenciam muitas instâncias de B_VO
- ✓ Muitas instâncias de B_VO referenciam muitas instâncias de A_VO

Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;
    @Column(length=40, nullable=false)
    private String nome;

    @ManyToMany(fetch=FetchType.LAZY)
    private Set<B_VO> listaB;
}
```

Mapeamento feito nas duas entidades de forma independente

```
@Entity
@Table(name = "b")
public class B_VO {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE)
    private int codigo;
    @Column(length=40, nullable=false)
    private String nome;

    @ManyToMany(fetch=FetchType.LAZY)
    private Set<A_VO> listaB;
}
```

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE a_b
(
  a_codigo integer NOT NULL,
  listab_codigo integer NOT NULL,
  CONSTRAINT a_b_pkey PRIMARY KEY (a_codigo, listab_codigo),
  CONSTRAINT fk17804c0244384 FOREIGN KEY (listab_codigo)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17804cf2d71c3 FOREIGN KEY (a_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Relações

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b_a
(
  b_codigo integer NOT NULL,
  listab_codigo integer NOT NULL,
  CONSTRAINT b_a_pkey PRIMARY KEY (b_codigo, listab_codigo),
  CONSTRAINT fk17bc4370f1301 FOREIGN KEY (b_codigo)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17bc4c023cf25 FOREIGN KEY (listab_codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Relações

```
@Entity
@Table(name = "a")
public class A_VO {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int codigo;
    @Column(length = 40, nullable = false)
    private String nome;
    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "a_b",
    joinColumns = {
        @JoinColumn(name = "a_id")},
    inverseJoinColumns = {
        @JoinColumn(name = "b_id")})
    private Set<B_VO> listaB;
}
```

Mapeamento feito nas duas entidades com referência entre eles

```
@Entity
@Table(name = "b")
public class B_VO {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int codigo;
    @Column(length = 40, nullable = false)
    private String nome;

    @ManyToMany(mappedBy="listaB", fetch = FetchType.LAZY)
    private Set<A_VO> listaA;
}
```

Mapeamento de Relações

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)
```

```
CREATE TABLE a_b
(
  a_id integer NOT NULL,
  b_id integer NOT NULL,
  CONSTRAINT a_b_pkey PRIMARY KEY (a_id, b_id),
  CONSTRAINT fk17804225dfb45 FOREIGN KEY (a_id)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk17804225ee403 FOREIGN KEY (b_id)
      REFERENCES b (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Mapeamento de Herança

- ✓ Tabela Única
  - Uma tabela por hierarquia de classe
- ✓ Vantagens
  - É mais simples de implementar (banco)
  - É a estratégia melhor para performance (uma única tabela sem junções)
- ✓ Desvantagem
  - Não é normalizada
  - Exige que todas as propriedades das subclasses permitam valores nulos

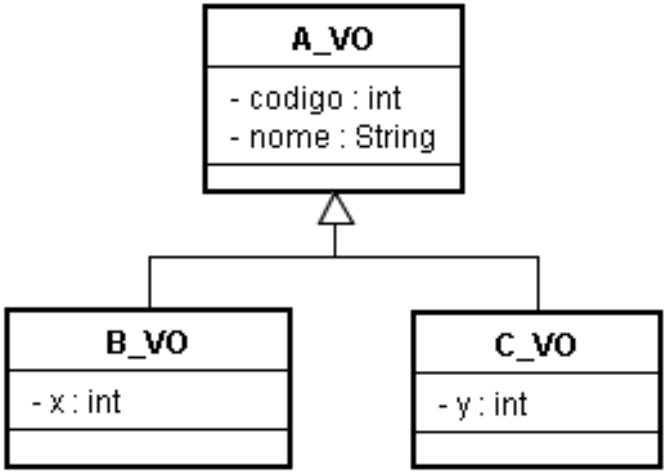
Mapeamento de Herança

```
@Entity
@Table(name = "a")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="discriminador", discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue("A_VO")
public class A_VO {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int codigo;
    @Column(length = 40, nullable = false)
    private String nome;
}
```

```
@Entity
@DiscriminatorValue("B_VO")
public class B_VO extends A_VO{

    @Column(nullable=false)
    private int x;
}
```

```
@Entity
@DiscriminatorValue("C_VO")
public class C_VO extends A_VO{

    @Column(nullable = false)
    private int y;
}
```

Mapeamento de Herança




```
CREATE TABLE a
(
  discriminador character varying(31) NOT NULL,
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  x integer NOT NULL,
  y integer NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)
```

Mapeamento de Herança

- ✓ Tabela por Classe Concreta
  - Uma tabela por cada classe concreta da hierarquia
- ✓ Vantagens
  - Permite que propriedades das subclasses possuam restrições de não nulo
- ✓ Desvantagem
  - Não é normalizada (colunas redundantes)
  - É menos performática que a simples tabela
  - Produz redundância de dados

Mapeamento de Herança

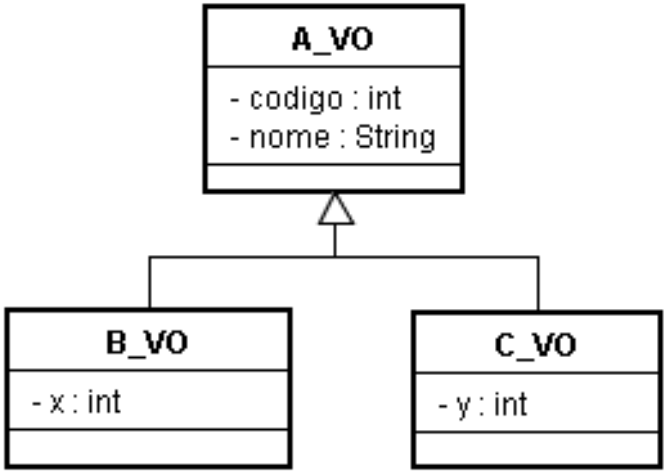
```
@Entity
@Table(name = "a")
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS)
public class A_VO {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int codigo;
    @Column(length = 40, nullable = false)
    private String nome;
}
```

```
@Entity
@Table(name = "b")
public class B_VO extends A_VO{

    @Column(nullable=false)
    private int x;
}
```

```
@Entity
@Table(name = "c")
public class C_VO extends A_VO{

    @Column(nullable = false)
    private int y;
}
```

Mapeamento de Herança

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)

CREATE TABLE b
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  x integer NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo)
)


CREATE TABLE c
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  y integer NOT NULL,
  CONSTRAINT c_pkey PRIMARY KEY (codigo)
)
```



Mapeamento de Herança

- ✓ Tabela por SubClasse
  - Cada subclasse tem sua própria tabela
- ✓ Vantagens
  - É normalizada
  - Não tem redundância de dados
- ✓ Desvantagem
  - Desempenho menor que a estratégia Simples Tabela

Mapeamento de Herança

```
@Entity
@Table(name = "a")
@Inheritance(strategy=InheritanceType.JOINED)
public class A_VO {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int codigo;
    @Column(length = 40, nullable = false)
    private String nome;
}
```

```
@Entity
@Table(name = "b")
public class B_VO extends A_VO{

    @Column(nullable=false)
    private int x;
}
```

```
@Entity
@Table(name = "c")
@PrimaryKeyJoinColumn(name="codigoA")
public class C_VO extends A_VO{

    @Column(nullable = false)
    private int y;
}
```

Mapeamento de Herança

```
CREATE TABLE a
(
  codigo integer NOT NULL,
  nome character varying(40) NOT NULL,
  CONSTRAINT a_pkey PRIMARY KEY (codigo)
)

CREATE TABLE b
(
  x integer NOT NULL,
  codigo integer NOT NULL,
  CONSTRAINT b_pkey PRIMARY KEY (codigo),
  CONSTRAINT fk62d16b7425 FOREIGN KEY (codigo)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE c
(
  y integer NOT NULL,
  codigoa integer NOT NULL,
  CONSTRAINT c_pkey PRIMARY KEY (codigoa),
  CONSTRAINT fk635a55c274 FOREIGN KEY (codigoa)
      REFERENCES a (codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```