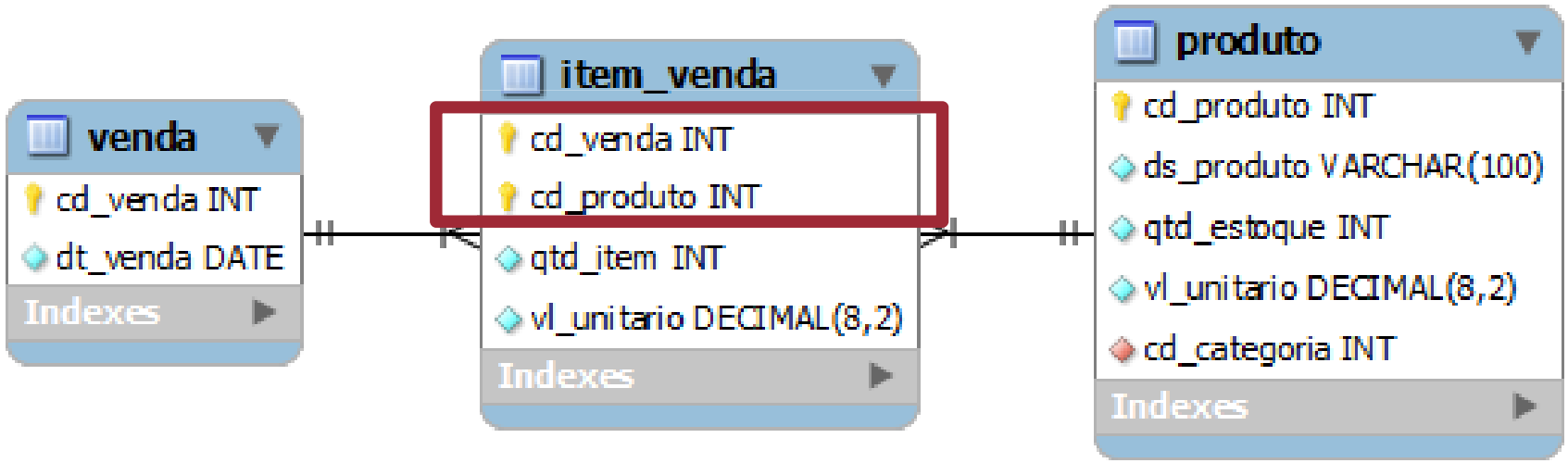
Java Persistence API (JPA)
Mapeamento de Chave Primária Composta

Fernando dos Santos
fernando.santos@udesc.br

UDESC
Ibirama

Chave Primária Composta

- Considere a tabela **item_venda**



- Chave primária é uma composição das PKs de **produto** e **venda**

Chave Primária Composta

- Em JPA, mapeamento de PK composta é feito em duas etapas:

1. Criar uma classe para representar a PK composta
   - definir os atributos para mapear os campos da chave

2. Usar a classe de PK para definir a chave na entidade.

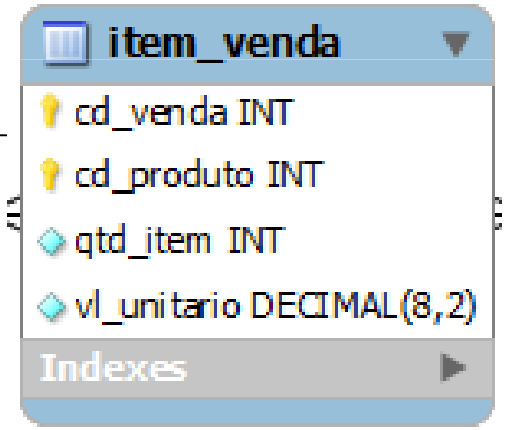
Etapa 1: classe de PK composta

- Criar uma classe para representar a PK, anotar c/ **@Embeddable**
- Mapear as colunas normalmente
  - Não usar anotação @Entity, @Table e @Id


```
@Embeddable
public class ItemVendaPK {

    @Column(name="cd_venda")
    private int idVenda;

    @Column(name="cd_produto")
    private int idProduto;

    // gets e sets
}
```

Etapa 2: usar PK para definir chave

- Declarar um atributo do tipo da PK, anotar c/ **@EmbeddedId**
- Fazer o mapeamento dos relacionamentos
  - **@ManyToOne ou @OneToOne**
  - Atenção ao insertable / updatable, devem ser false




```
@Entity
@Table(name="item_venda")
public class ItemVenda {
  @Column(name="qtd_item")        // atributos
  private int quantidade;         // simples do item
  @Column(name="vl_unitario")
  private double valorUnitario;

  @EmbeddedId
  private ItemVendaPK itenVendaPK;

  @ManyToOne
  @JoinColumn(name = "cd_venda", insertable = false, updatable = false)
  private Venda venda;

  @ManyToOne
  @JoinColumn(name="cd_produto", insertable = false, updatable = false)
  private Produto produto;
```

Manipulação de entidade com PK composta (1)

- Ao persistir, é necessário instanciar e definir o objeto PK

```
Produto prod = em.find(Produto.class, 1);

Venda ven = new Venda();
ven.setDataVenda(new Date());

ItemVenda item = new ItemVenda();
item.setProduto(prod);
item.setVenda(ven);
item.setQuantidade(3);
item.setValorUnitario(prod.getValorUnitario());

em.getTransaction().begin();
em.persist(ven);
em.flush();

ItemVendaPK itemPK = new ItemVendaPK();
itemPK.setIdProduto(prod.getId());
itemPK.setIdVenda(ven.getId());
item.setItenVendaPK(itemPK);

em.persist(item);
em.getTransaction().commit();
```

faz associação do item com a Venda e Produto, para manter o relaciona-mento coerente

Venda ainda não possui o valor da chave. flush() é necessário para o banco gerar este valor da chave na Venda

tendo o ID de produto e venda, é possível criar o objeto PK e setar na entidade item

Manipulação de entidade com PK composta (2)

- Buscas por chave (find ou select) deve utilizar **objeto da PK**

```
ItemVendaPK itemPK = new ItemVendaPK();
itemPK.setIdProduto(1);
itemPK.setIdVenda(1);

ItemVenda item = em.find(ItemVenda.class, itemPK);
System.out.println("Prod: "+item.getProduto().getDescricao());
System.out.println("Dt Venda: "+item.getVenda().getDataVenda());
```

- Consultas devem navegar pelo relacionamento p/ acessar PK.

```
Query cons = em.createQuery("select iv from ItemVenda iv
                        where iv.produto.id = 1 and iv.venda.id = 1") ;

ItemVenda item = (ItemVenda) cons.getSingleResult();
System.out.println("Prod: "+item.getProduto().getDescricao());
System.out.println("Dt Venda: "+item.getVenda().getDataVenda());
```

Bibliografia

- BURKE, Bill; MONSON-HAEFEL, Richard. **Enterprise JavaBeans 3.0**. 5.ed. São Paulo: Prentice Hall, 2007. 538 p.