

# Banco de Dados II

Persistência Aula 10

Vanessa Cristina Oliveira de Souza



## **HIBERNATE**





- Desenvolvimentos:
  - □ Top Down
    - Modelo de domínio -> MR
    - Botton Up
      - MR -> Modelo de domínio





- Criar projeto
- 2. Arquivo de configuração
  - Configura a conexão com o banco de dados
- 3. Arquivo de engenharia reversa
  - Arquivo xml com os dados do banco (tabelas)
- 4. Mapeamento Objeto-Relacional
  - POJOs de todas as tabelas do banco
- Arquivo HibernateUtil
  - Para abrir as sessões







## Hibernate Inserir um objeto no banco

```
import Mapeamento.*;
import org.hibernate.*;
public class Main {
   public static void main(String[] args) {
        //Cria objeto disciplina
        Disciplinas disc = new Disciplinas();
        disc.setCodigo("SIT430");
        disc.setNome("Data Warehouse");
        disc.setCreditos("3");
        //cria uma sessão
        Session sessao = HibernateUtil.getSessionFactory().openSession();
        //salva objeto disc - Objeto transiente
        sessao.save(disc);
        //abre uma transação
        Transaction tr = sessao.beginTransaction();
        //comita no banco o que foi salvo até agora
        //disc passa a ser um objeto persistente
        tr.commit();
        //fecha a sessão
        sessao.close();
```



# Hibernate Inserir um objeto no banco

```
mysql> select * from disciplinas;
 codigo | nome
                                          | creditos
 CC0013 | Fundamentos de Programação
 CCT620 | Banco de Dados II
        l Computador e Sociedade
        l Estrutura de Dados I
    T420 | Laboratório de Banco de Dados
5 rows in set (0.00 sec)
mysql> select * from disciplinas;
 codigo | nome
                                          | creditos
        l Fundamentos de Programação
     620 | Banco de Dados II
         l Computador e Sociedade
           Estrutura de Dados I
          Data Warehouse
 rows in set (0.00 sec)
```



## Hibernate Apagar um objeto do banco

```
package universidade;
import Mapeamento.*;
import org.hibernate.*;
public class Main {
    public static void main(String[] args) {
        //Cria objeto disciplina
        Disciplinas disc = new Disciplinas();
        //cria uma sessão
        Session sessao = HibernateUtil.getSessionFactory().openSession();
        //LOAD - Recupera um objeto do banco e salva em disc.
        //LOAD recebe o objeto e a chave primária
        //conversão de objeto persistente em transiente.
        sessao.load(disc, "SIT430");
        //apaga o objeto do banco
        sessao.delete(disc);
        //abre uma transação
        Transaction tr = sessao.beginTransaction();
        //comita no banco o que foi salvo até agora
        //disc passa a ser um objeto persistente
        tr.commit();
        //fecha a sessão
        sessao.close();
```



```
mysql> select * from disciplinas;
 codigo | nome
                                           creditos
  CC0013 | Fundamentos de Programação
         l Banco de Dados II
         l Computador e Sociedade
           Estrutura de Dados I
           Data Warehouse
 rows in set (0.00 sec)
mysql> select * from disciplinas;
  codigo l
                                           creditos
          nome
        l Fundamentos de Programação
         l Banco de Dados II
          Computador e Sociedade
           Estrutura de Dados I
           Laboratório de Banco de Dados
  rows in set (0.00 sec)
```



### Exemplo



- Cadastrar uma nova turma
  - □ Disciplina "CCO013"
  - □ Codigo "Teste"
  - □ Vagas 30
  - □ Professor Maria -> matricula 1

mysql> select +   disciplina	+	+-		+	professor
CSP410	CC0_2007_1 CC0_2008_1 SIT_207_1	ł	25		2   2   1
3 rows in set (0.00 sec)					



### Exemplo



- Cadastrar uma nova turma
  - □ Disciplina "CCO013"
  - □ Codigo "Teste"
  - □ Vagas 30
  - □ Professor Maria -> matricula 1

```
public class Turmas implements java.io.Serializable {
    private TurmasId id;
    private Disciplinas disciplinas;
    private Professores professores;
    private Short vagas;
```

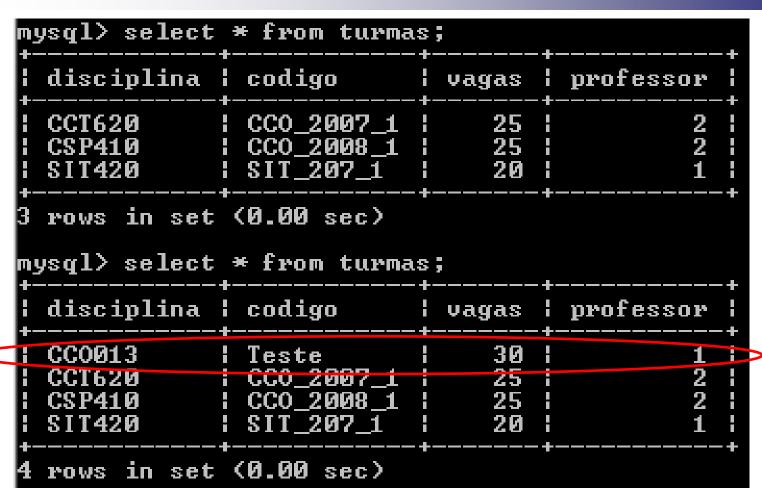


### Hibernate Inserir uma nova Turma

```
public static void main(String[] args) {
    //Cria objeto TurmasId já inicializado com os valores de referência.
    TurmasId id = new TurmasId("CCO013", "Teste");
    //Cria objeto disciplina
    Disciplinas disc = new Disciplinas();
    //Cria objeto professor
    Professores prof = new Professores();
   //Cria objeto turma
    Turmas turma = new Turmas();
    //cria uma sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //Recupera a disciplina que é FK
    sessao.load(disc, "CC0013");
    //Recupera o professor que é FK
    sessao.load(prof, (short)1);
    //Atualizar os valores do objeto turma
    turma.setId(id);
    turma.setDisciplinas(disc);
    turma.setProfessores(prof);
    turma.setVagas((short)30);
    //salva o objeto
    sessao.save(turma);
    //abre uma transação
    Transaction tr = sessao.beginTransaction();
    //comita no banco o que foi salvo até agora
    tr.commit();
    //fecha a sessão
    sessao.close();
```



### Hibernate Inserir uma nova Turma





### Update



- Alterar a titulação do professor João para Doutor
  - ☐ Matricula 2

```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor
    Professores prof = new Professores();
   //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
   //iniciar a transação
   Transaction tr = sessao.beginTransaction();
   //carrega o professor
   prof = (Professores) sessao.load(Professores.class, (short)2);
   //altera o objeto
   prof.setTitulacao("Doutor");
    //update
    sessao.update(prof);
    //comitt
   tr.commit();
    //fechar a sessão
    sessao.close();
```



### saveOrUpdate

- Alterar o professor da turma CCO013/Teste para o João
  - Matricula 2

O que acontece?

```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor, turmas e turmasId
    Professores prof = new Professores();
    Turmas turmas = new Turmas();
    TurmasId id = new TurmasId();
    //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //iniciar a transação
    Transaction tr = sessao.beginTransaction();
    //carrega o NOVO professor
    prof = (Professores) sessao.load(Professores.class, (short)2);
    //setar o id da turma
    id.setCodigo("Teste");
    id.setDisciplina("CC0013");
    turmas.setId(id);
    //setar o novo professor da turma
    turmas.setProfessores(prof);
    sessao.saveOrUpdate(turmas);
    tr.commit();
    //fechar sessao
    sessao.close();
```



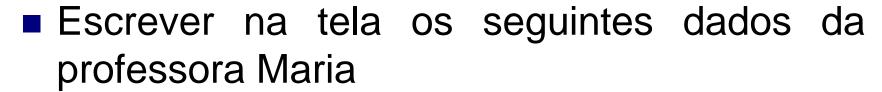
#### Exercício



Cadastrar um novo pedido (northwind.order)



### Exemplo



- Nome
- □ Número de turmas
- □ Código das turmas



### Exemplo

```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor, turmas e turmasId
    Professores prof = new Professores();
    Turmas turma = new Turmas();
    //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //carrega o NOVO professor
   prof = (Professores) sessao.load(Professores.class, (short)1);
    System.out.print(prof.getNome());
    System.out.print(prof.getNroTurmas());
    Iterator i = prof.getTurmases().iterator();
    while(i.hasNext())
    4
        turma = (Turmas) i.next();
        System.out.print(turma.getId().getCodigo());
    //fechar sessao
    sessao.close();
```



#### Consultas



#### HQL

- ☐ Hibernate Query Language
- □ Linguagem de consulta que se parece muito com a SQL
- □ A HQL é totalmente orientada a objeto, incluindo os paradigmas de herança, polimorfismo e encapsulamento.
- Executar os pedidos SQL sobre as classes de persistência do Java ao invés de tabelas no banco de dados, aumentando, assim, a distância entre o desenvolvimento da regras de negócio e o banco de dados.
- □ <a href="http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html">http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html</a>



#### Consultas



#### CRITERIA

- □ API alternativa à HQL
- □ Voltado para consultas onde o número de parâmetros não é conhecido, como aquelas que contém N filtros
- Nesses casos, agiliza a consulta na base de dados.
- □ Consulta por Critério (QBC)
- □ Consulta por Exemplo (QBE)
- http://docs.jboss.org/hibernate/core/3.3/reference/en/html/querycriteria.html



### Consulta Simples



- Selecionar todos os registros de professoresHQL
- public static void main(String[] args) throws Exception{
   Professores prof = new Professores();
   Session sessao = HibernateUtil.getSessionFactory().openSession();
   Iterator i = sessao.createQuery("from Professores").list().iterator();
   while(i.hasNext())
   {
   prof = (Professores) i.next();
   System.out.printf("%s\t%s\n", prof.getNome(), prof.getTitulacao());
   }
   sessao.close();
  }



### Consulta Simples



- Selecionar as turmas da professora Maria
  - □ SQL
    - SELECT \* from Turmas, Professores WHERE Turmas.professor = Professores.matricula AND Professores.nome like 'MARIA%';
  - HQL

```
public static void main(String[] args) throws Exception{
   Turmas turma = new Turmas();
   Session sessao = HibernateUtil.getSessionFactory().openSession();
   Iterator i = sessao.createQuery("from Turmas where professores.nome like '%Maria%'").list().iterator();
   while(i.hasNext())
   {
      turma = (Turmas) i.next();
      System.out.printf("%s\t%s\n", turma.getId().getCodigo(), turma.getId().getDisciplina());
   }
   sessao.close();
}
```



### Projeção

Problema?



□ HQL

```
public static void main(String[] args) throws Exception{
   Turmas turma = new Turmas();
   Session sessao = HibernateUtil.get@essionFactory().openSession();
   Iterator i = sessao.createQuer(("codigo, disciplina from Turmas where professores.nome like '%Maria%'").list().iterator();
   while(i.hasNext())
   {
     turma = (Turmas) i.next();
        System.out.printf("%s\t%s\n", turma.getId().getCodigo(), turma.getId().getDisciplina());
   }
   sessao.close();
}
```



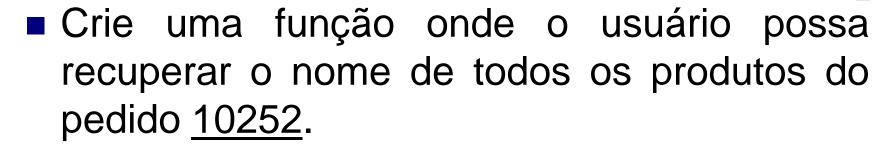
### Projeção



- Selecionar código, disciplina das turmas da professora Maria
  - □ HQL



#### Exercício





#### Exercício

- Na tabela northwind.products, crie as chaves estrangeiras para os atributos supplierid e categoryid.
- Faça um pequeno programa que realiza um CRUD nessa tabela, usando:
  - □ JDBC
  - ☐ Hibernate
  - \*No update, pode considerar a alteração de apenas um campo da tabela para facilitar.
  - □ \*Da mesma forma, a consulta pode ser limitada a uma única chave primária.
  - \*Não se preocupe com a interface!