
Prog. Orientada a Objetos e Mapeamento Objeto-Relacional – IMD0104

Consulta JPQL

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

Java Persistence Query Language

- Java Persistence Query Language (JPQL) é uma **linguagem de consulta**:
 - ❖ **Orientada a objeto**;
 - ❖ Opera sobre **classes e objetos**, diferente do SQL que opera sobre tabelas;
 - ❖ **Independente** de SGBD;
 - ❖ Definida como parte da especificação Java Persistence API (JPA);
 - ❖ Fortemente **inspirada na linguagem SQL**.

Java Persistence Query Language

❑ Exemplos de consultas:

```
// Busca todos os alunos da tabela aluno
SQL: select * from aluno
JPQL: from Aluno
JPQL: from Aluno a
JPQL: from Aluno as alu
```

```
// Busca todos os alunos matricula >= 35
SQL: select * from aluno where matricula >= 35
JPQL: from Aluno a where a.matricula >= 35
```

```
// Todos os professores com nome começando com João
SQL: select * from professor where nome like "João%"
JPQL: from Professor p where p.nome like "João%"
```

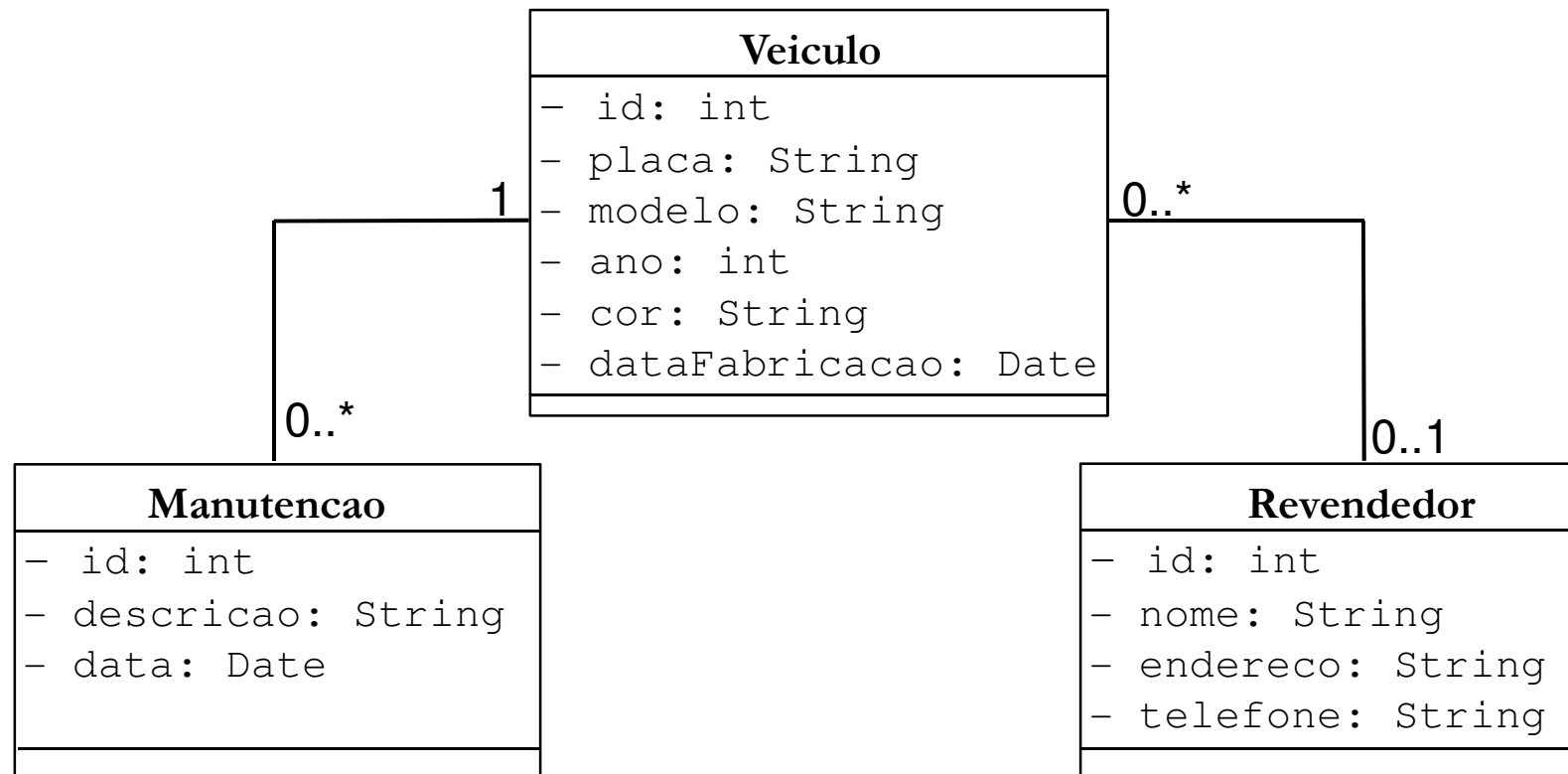
Java Persistence Query Language

❑ Exemplos de consultas:

```
// Busca todos os alunos ordenados por nome (asc)  
SQL: select * from aluno order by nome asc  
JPQL: from Aluno order by nome asc
```

Java Persistence Query Language

❑ Exemplo de domínio:



Java Persistence Query Language

❑ Usando mapeamento @ManyToOne:

```
// Manutencoes de veículos do ano 2009
SQL: select m.* from veiculo v, manutencao m
      where v.id_veiculo = m.id_veiculo
      and v.ano = 2009
JPQL: select m from Manutencao m
      where m.veiculo.ano = 2009
```

```
// Veículos com nome do revendedor começando com A%
SQL: select r.* from veiculo v, revendedor r
      where v.id_revendedor = r.id_revendedor
      and r.nome like 'A%'
JPQL: from Veiculo v
      where v.revendedor.nome like 'A%'
```

Java Persistence Query Language

❑ Usando mapeamento @ManyToOne:

```
// Manutenções com revendedor dos veículos com nome  
começando com A%
```

```
SQL: select m.* from manutencao m, veiculo v, revendedor r  
      where v.id_revendedor = r.id_revendedor  
      and m.id_veiculo = v.id_veiculo  
      and r.nome like 'A%'
```

```
JPQL: select m from Manutencao m  
      where m.veiculo.revendedor.nome like 'A%'
```

```
JPQL: select m from Manutencao m join m.veiculo v  
      where v.revendedor.nome like 'A%'
```

```
JPQL: select m from Manutencao m  
      join m.veiculo v join v.revendedor r  
      where r.nome like 'A%'
```

Java Persistence Query Language

❑ Usando mapeamento @OneToMany:

```
// Revendedores de veiculos do ano = 2009
SQL: select distinct r.*
      from veiculo v, revendedor r
      where v.id_revendedor = r.id_revendedor
      and v.ano = 2009

JPQL: select distinct r from Revendedor r
      join r.veiculo v
      where v.ano = 2009
```


Java Persistence Query Language

❑ Outras consultas:

```
// Conta quantos .....?
```

```
SQL: select count(v.id_veiculo), v.ano  
      from veiculo v  
      group by v.ano
```

```
JPQL: select count(v.id), v.ano  
        from Veiculo v  
        group by v.ano
```

```
SQL: select max(v.ano) from veiculo v
```

```
JPQL: select max(v.ano) from Veiculo v
```

Java Persistence Query Language

❑ Outras consultas:

```
JPQL: select v from Veiculo v  
      where year(dataFabricacao) = 2008
```

```
JPQL: select v from Veiculo v  
      where month(dataFabricacao) = 03
```

```
JPQL: select v from Veiculo v  
      where day(dataFabricacao) = 30
```

Java Persistence Query Language

❑ Outras consultas:

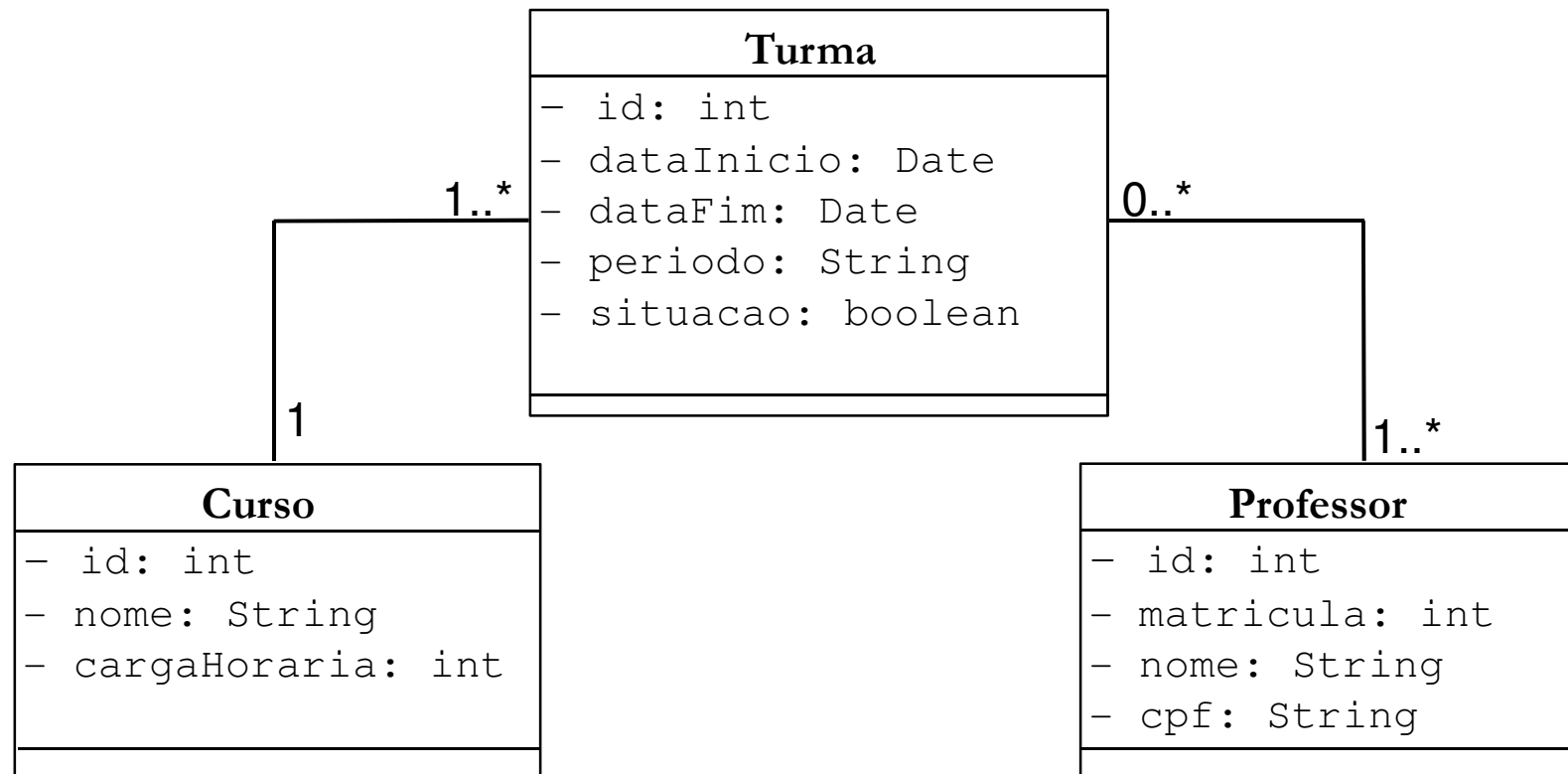
```
JPQL: select avg(p.peso), sum(p.peso), max(p.peso)
       from Pessoa p
```

```
JPQL: select max(p.idade), min(p.idade)
       from Pessoa p
```

```
JPQL: select p.idade, sum(p.peso), count(p)
       from Pessoa p
       group by p.idade
       having p.idade >= 12
       order by p.idade
```

Java Persistence Query Language

❑ Exemplo de domínio 02:



Java Persistence Query Language

❑ Usando mapeamento @ManyToMany:

```
// Todos os professores de turmas com periodo =  
// "Tarde" e com cursos de cargaHoraria > 40  
SQL: select p.* from professor p  
      join professor_turma pt on (pt.id_professor =  
      p.id_professor)  
      join turma t on (t.id_turma = pt.id_turma)  
      join curso c on (c.id_curso = t.id_curso)  
      where t.periodo = "Tarde"  
      and c.carga_horaria > 40  
  
JPQL: select p from Professor p  
      join p.turma t  
      where t.periodo = "Tarde"  
      and t.curso.cargaHoraria > 40
```

Java Persistence Query Language

❑ Usando JPQL na aplicação:

❖ Interface Query para listar todos os registros:

```
// Todos os alunos  
Query q = em.createQuery("from Aluno");  
q.getResultList();
```

```
// Todos os cursos começando pela letra A:  
String s = "from Curso c where c.nome like 'A%'";  
Query q = em.createQuery(s);  
q.getResultList();
```

```
// Todos os cursos com que tem o departamento = 2  
String s = "from Curso c where c.departamento.id = 2";  
Query q = em.createQuery(s);  
q.getResultList();
```

Java Persistence Query Language

- ❑ É possível dar nomes aos parâmetros:

```
// Lista de objetos Aluno
Query q = em.createQuery("from Aluno a where a.nome = :nome");
q.setParameter("nome", "João");
List result = q.getResultList();
```

```
// Lista de objetos Empregado
Query q = em.createQuery("from Empregado e
    where e.empresa.tipo = :tipo
    and e.salario > :salario
    and e.idade > e.esposa.idade");

q.setParameter("tipo", 1);
q.setParameter("salario", 1500);
List result = q.getResultList();
```

Java Persistence Query Language

❑ Retorno de um objeto único:

```
// Único objeto como resultado
Query q = em.createQuery("
    select a from Aluno a where a.id = 109");
Aluno a = (Aluno)q.getSingleResult();
```


Java Persistence Query Language

- ❑ Usando **JPQL** na prática:
 - ❖ Listando objetos do tipo Professor:

```
Query q = em.createQuery("from Professor");
List <Professor> results = q.getResultList();

for(Professor p: results){
    System.out.println(p.getId() + " " +
        p.getNome() + " " + p.getMatricula() + " " +
        p.getCpf());
}
```

Java Persistence Query Language

- ❑ Usando **JPQL** na prática:
 - ❖ Listando objetos do tipo Professor:

```
Hibernate:
  select
    professor0_.id as id1_2_,
    professor0_.cpf as cpf2_2_,
    professor0_.matricula as matricul3_2_,
    professor0_.nome as nome4_2_
  from
    Professor professor0_
10 João Carlos Xavier Júnior 4351681 423.555.789-20
11 Francisco dos Anjos 3254178 963.852.741-00
12 Maria Eduarada da Silva 654789 987.258.745-33
```

Java Persistence Query Language

❑ Projeções em consulta:

- ❖ Usadas para retornar um **subconjunto** das propriedades de uma classe.
- ❖ Os **valores dos campos** são extraídos dos **objetos** de uma classe.

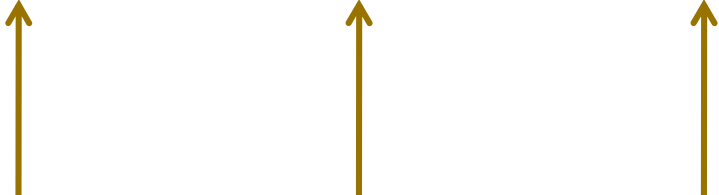
❑ Exemplo:

```
// A query seguinte retorna o objeto endereço  
// de uma pessoa especificada:  
"select p.endereco from Pessoa p  
  where p.matricula = :mat"
```

Java Persistence Query Language

❑ Exemplo de Projeção em uma consulta:

```
public class Main03 {  
  
    static EntityManager em;  
  
    public static void main(String[] args) {  
  
        em = Conexao.getInstance();  
        Query q = em.createQuery("select p.nome, p.cpf,"  
                                + "p.matricula from Professor p");  
        List <Object[]> results = q.getResultList();  
        for(Object[] obj: results){  
            System.out.println(obj[0] + " " + obj[1] + " " + obj[2]);  
        }  
        em.close();  
    }  
}
```



Java Persistence Query Language

❑ Exibindo uma Projeção:

```
INFO: HHH000126: Indexes: []
Set 27, 2015 6:53:00 PM org.hibernate.tool.hbm2ddl.SchemaUpdate execute
INFO: HHH000232: Schema update complete
Hibernate:
  select
    professor0_.nome as col_0_0_,
    professor0_.cpf as col_1_0_,
    professor0_.matricula as col_2_0_
  from
    Professor professor0_
João Carlos Xavier Júnior 423.555.789-20 4351681
Francisco dos Anjos 963.852.741-00 3254178
Maria Eduarda da Silva 987.258.745-33 654789
```

Java Persistence Query Language

❑ Outro exemplo de projeção:

```
String jpql = "select c.id, c.nome, a.id, a.nome  
from Curso c, Aluno a where a.curso.id = c.id";  
Query q = em.createQuery(jpql);  
ArrayList resultado = new ArrayList();  
// Lista com vetor de dados  
Iterator it = q.getResultList().iterator();  
while ( it.hasNext() ) {  
    Object[] obj = (Object[]) it.next();  
    Curso curso = new Curso();  
    curso.setId((Integer)obj[0]);  
    curso.setNome((String)obj[1]);  
    Aluno aluno = new Aluno();  
    aluno.setId((Integer)obj[2]);  
    aluno.setNome((String)obj[3]);  
    aluno.setCurso(curso);  
    resultado.add(aluno);  
}
```

Java Persistence Query Language

- ❑ Outro exemplo de projeção @ManyToMany:

```
System.out.println("Consultas: ");
@SuppressWarnings("unchecked")
String jpql = "select a.id, a.nome, a.cpf, l.titulo, l.isbn "
            + "from Autor a join a.livros l";
List<Object[]> lista = em.createQuery(jpql).getResultList();
for (Object[] obj : lista ){
    System.out.println(obj[0] + "\t" + obj[1] + "\t" +
                       obj[2] + "\t" + obj[3] + "\t" + obj[4]);
}
em.close();
```

Java Persistence Query Language

❑ Exibindo outra projeção @ManyToMany:

Consultas:

Hibernate:

```
select
    autor0_.id as col_0_0_,
    autor0_.nome as col_1_0_,
    autor0_.cpf as col_2_0_,
    livro2_.titulo as col_3_0_,
    livro2_.isbn as col_4_0_
from
    Autor autor0_
inner join
    autor_livro livros1_
        on autor0_.id=livros1_.id_autor
inner join
    Livro livro2_
        on livros1_.id_livro=livro2_.id
```

32	Angelina Jolie Fields	632.417.258-11	Malévola for Kids	4567-6333
30	Francisco Pereira	125.478.365-33	Malévola for Kids	4567-6333
31	Roberto Carlos da Silva	521.587.745-47	Second World War	4714-5877

Java Persistence Query Language

- ❑ **Consulta** que recupera do banco apenas as linhas dentro de uma **faixa de resultados**:

```
Query q = em.createQuery("from Aluno");  
q.setFirstResult(10);  
q.setMaxResults(30);  
q.getResultList();
```

Java Persistence Query Language

❑ Atualizações de valores (Update):

```
// ...  
em.getTransaction().begin();  
Query q = em.createQuery(  
    "update Curso set nome = :novoNome "  
    + "where id = :idCurso");  
q.setParameter("novoNome", "BTI");  
q.setParameter("idCurso", 150);  
  
q.executeUpdate();  
em.getTransaction().commit();  
// ...
```

Java Persistence Query Language

❑ Atualizações de valores (Delete):

```
// ...
em.getTransaction().begin();
Query q = em.createQuery(
    "delete from Aluno where curso.id = :idCurso");
q.setParameter("idCurso", 150);

q.executeUpdate();

em.getTransaction().commit();
// ...
```

Dúvidas...

