02

Resolvendo o Problema do N+1

Dica: Caso precise, você pode baixar o projeto até o momento aqui (https://github.com/alura-cursos/Nhibernate/archive/capitulo4.zip)

Quando desenvolvemos software utilizando o NHibernate, não precisamos nos preocupar tanto com o banco de dados relacional, o que facilita muito o desenvolvimento, mas devemos cuidar para que essas facilidades não acabem prejudicando a performance do sistema.

Como aprendemos anteriormente, o NHibernate carrega todos os relacionamentos de forma lazy, ou seja, os relacionamentos são carregados apenas quando necessário. Imagine que em nossa loja, queremos imprimir a lista com o nome de todos os produtos junto com o nome da categoria de cada produto.

```
IQuery query = session.CreateQuery("from Produto");
IList<Produto> produtos = query.List<Produto>();
foreach(var produto in produtos)
{
    Console.WriteLine(produto.Nome + " - " + produto.Categoria.Nome);
}
```

No código acima, fazemos uma query para recuperar a lista de todos os produtos e depois para cada produto imprimimos seu nome e o nome de sua categoria, porém a categoria é um relacionamento, então o NHibernate só a carrega quando necessário (quando acessamos seu nome), ou seja, para cada produto estamos enviando uma query para carregar sua categoria. Esse problema é conhecido como N + 1 queries.

Evitando o problema do N+1

Quando estamos utilizando a SQL, podemos resolver o problema das N + 1 queries utilizando um join:

```
select p.*, c.*
from Produto p join Categoria c on p.CategoriaId = c.Id
```

Podemos fazer algo parecido utilizando a HQL. Para especificar que a query que traz a lista de produtos deve buscar também as categorias utilizamos um join especial do NHibernate, o fetch join:

```
select p from Produto p join fetch p.Categoria
```

Agora que estamos utilizando o fetch join, essa query carrega a lista de produtos com as categorias já carregadas.

N + 1 em relacionamentos to many

Vamos agora pensar numa query que devolve a lista de categorias.

```
string hql = "from Categoria";
IQuery query = session.CreateQuery(hql);
```

```
IList<Categoria> categorias = query.List<Categoria>();
```

Para cada categoria queremos imprimir o tamanho de sua lista de produtos:

```
string hql = "from Categoria";
IQuery query = session.CreateQuery(hql);
IList<Categoria> categorias = query.List<Categoria>();
foreach(var categoria in categorias)
{
    Console.WriteLine(categoria.Nome + " - " + categoria.Produtos.Count);
}
```

Como a lista de produtos é um relacionamento do tipo one to many, ela é carregada de forma lazy pelo NHibernate. Quando executamos `categoria.Produtos.Count` , o NHibernate executa uma query que busca os produtos relacionados a categoria. Temos novamente o problema de N + 1 Queries.

Para resolver o problema de N + 1 queries no relacionamento to many, também utilizamos o fetch join:

```
string hql = "select c from Categoria c join fetch c.Produtos";
```

Porém, quando colocamos o fetch join na query, o NHibernate gera uma SQL que contém um join em uma coleção, ou seja, o resultado da query conterá categorias duplicadas.

Para removermos as duplicações do resultado da query, utilizamos o `select distinct` da HQL:

```
string hql = "select distinct c from Categoria c join fetch c.Produtos";
```

Com isso resolvemos o problema das N + 1 queries, porém se a aplicação estivesse acessando um banco de dados com 100 categorias cada uma com 100 produtos, então o fetch join traria 10000 linhas de resultado do banco de dados!

Quando trabalhamos com bancos muito grandes, algumas vezes o join fetch pode aumentar muito o número de linhas do resultado. Não queremos executar uma query por categoria mas também não queremos um resultado muito grande, estamos interessados em uma solução intermediária que consiga trazer a lista de produtos de, por exemplo, 10 categorias por vez.

Para resolver esse problema na SQL, utilizaríamos a seguinte query:

```
select * from Produto where CategoriaId in (lista_de_ids_de_categorias)
```

Podemos configurar o NHibernate para fazer uma query parecida com a exemplificada acima para trazer a coleção do relacionamento. Essa configuração é feita através do atributo `batch-size` da tag `bag` no arquivo de mapeamento da entidade.

Para configurarmos o `batch-size` da lista de produtos da categoria, vamos abrir o `Categoria.hbm.xml` e dentro da tag `bag` vamos colocar o atributo `batch-size` com o número de categorias que devem ter sua lista de produtos carregadas por vez:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
  assembly="Loja" namespace="Loja.Entidades">
```

```
    <class name="Categoria">
      <!-- mapeamento do nome e do id da categoria -->
      <bag name="Produtos" batch-size="10">
        <key column="CategoriaId"/>
        <one-to-many class="Produto"/>
      </bag>
    </class>
  </hibernate-mapping>
```

Agora que colocamos o `batch-size` com o valor 10, o número de queries que serão executadas será `(número de elementos da lista) / (batch-size)` para nosso exemplo de uma lista com 100 categorias, com o batch-size configurado com o valor 10, executaríamos 10 querys ao invés das 100 iniciais.