MAPEAMENTO OBJETO RELACIONAL: UM ESTUDO DE CASO UTILIZANDO O *HIBERNATE*

Rafael Laurino GUERRA, Dra. Luciana Aparecida Martinez ZAINA
Faculdade de Tecnologia de Indaiatuba – FATEC-ID

RESUMO

Este artigo apresenta os conceitos sobre mapeamento objeto relacional e implementa um estudo de caso que utiliza uma linguagem de programação orientada a objetos e um banco de dados relacional. O objetivo é mostrar como esta ferramenta pode auxiliar no desenvolvimento de sistemas, facilitando o trabalho do desenvolvedor.

Palavras Chave: Persistência de dados; Modelo relacional; Modelo orientado a objetos.

1 Introdução

O desenvolvimento de sistemas cada vez mais complexos fez com que a linguagem de programação orientada a objetos venha sendo uma das mais utilizadas no desenvolvimento de sistemas. Por outro lado, o modelo relacional é hoje o modelo mais aceito e utilizado quando se deseja armazenar dados.

Mapeamento objeto relacional surge como uma alternativa para os desenvolvedores de sistemas que não querem abrir mão dos benefícios que a linguagem de programação orientada a objetos possui, e que também sabem que um banco de dados puramente orientado a objetos está longe de conseguir uma boa aceitação no mercado. A idéia é ter um mecanismo que faça a conversão entre objetos do sistema e as tabelas do banco de dados relacional.

O objetivo principal deste trabalho é prover uma abordagem sobre os processos que ocorrem durante a implementação de um mapeamento objeto relacional, tal como mostrar o funcionamento do *Hibernate*, que é considerada pelos especialistas da área como uma das ferramentas mais utilizadas para realizar o mapeamento objeto relacional.

Para ilustrar os processos que ocorrem na realização de um mapeamento objeto relacional foram utilizadas neste artigo, as tecnologias Java como a linguagem de programação orientada a objetos, o *SQL server* como banco de dados relacional, o *Hibernate* como ferramenta de mapeamento e o padrão UML para representar os estágios de desenvolvimento do estudo de caso.

Este trabalho está organizado em 5 seções. Esta seção exibe a motivação, objetivos e metodologias adotadas. A seção 2 aborda os conceitos que envolvem o mapeamento objeto relacional. A seção 3 fala sobre o funcionamento do *Hibernate*. A seção 4 traz um estudo de caso de empréstimos de livros de uma biblioteca. E a seção 5 traz as conclusões, objetivos alcançados, limitações e trabalhos futuros.

2 Mapeamento objeto relacional

Segundo Bauer (2005, p.23) Mapeamento Objeto Relacional é "persistir de maneira automática e transparente, os objetos de um aplicativo para tabelas em um banco de dados relacional. Em essência, transforma dados de uma representação para a outra".

No desenvolvimento de um sistema, muitas vezes o programador dedica boa parte do tempo de desenvolvimento construindo comandos de instruções SQL para realizar a persistência dos dados no banco de dados relacional. Como ilustra a Figura 1, o aplicativo precisará de uma camada de mapeamento objeto relacional, que irá traduzir as estruturas e operações do sistema orientado a objetos para o banco de dados relacional.

Figura 1. Mapeamento de um aplicativo Orientado a Objetos, Adaptado de: (DOEDERLEIN, 2006, p.25).

O mapeamento objeto relacional faz a persistência automática de dados de uma representação para outra. Persistência se trata do armazenamento de dados que estão em meio volátil, como a memória RAM, para dispositivos de memória secundária, o disco rígido, por exemplo. Consiste em manter em meio físico recuperável, como banco de dados, arquivo etc. Quando se fala de persistência em linguagem de programação orientada a objetos, normalmente a preocupação é de como armazenar dados em um banco de dados relacional (BAUER, 2007, p. 05).

Persistência em Banco de Dados Relacionais: O banco de dados relacional suporta criação e alteração de tabelas, inserção, atualização e exclusão de dados, agrupamentos, ordenação e agregação. Sua conexão com o sistema se dá através de uma interface de programação (API - *Application Programming Interface*) de conectividade do banco de dados, que é responsável em fazer o envio de instruções SQL para o banco de dados relacional. A tarefa de persistir objetos utilizando instruções SQL através da API de conectividade é muito trabalhosa e tediosa, o que pode gerar erros no desenvolvimento do sistema (BAUER, 2007, p. 22).

Persistência em Banco de Dados Orientado a Objetos: persistência em banco de dados orientados a objetos, trabalha com um banco de dados puramente orientado a objetos. Seu funcionamento se dá com o armazenamento de dados organizados em hierarquias de classes e não em tabelas. O problema é que este tipo de banco de dados não possui uma utilização muito difundida, devido a sua falta de padronização. Suas APIs possuem diferenças de um banco de dados para outro, não permitindo assim uma portabilidade do sistema, o que não é bom, pois dentro da comunidade de desenvolvedores de sistemas orientado a objetos o que mais é valorizado é justamente a padronização e portabilidade de um sistema (DOEDERLEIN, 2005, p.22).

2.1 Vantagens do Mapeamento Objeto relacional

De acordo com Esjug (2007, p.14) implementar um mapeamento objeto relacional pode ser considerada uma tarefa complexa, porém esta tecnologia possui algumas vantagens:

- **Produtividade** – com a eliminação dos códigos SQL no código fonte, as classes passam a ser mais simples e com isso o sistema é desenvolvido em menor tempo.

- **Manutenibilidade** – por reduzir o número de linhas do código fonte do sistema, menor será o trabalho de manutenção do sistema.
- **Desempenho** – o tempo economizado no desenvolvimento, pode ser dedicado a programar otimizações do sistema.
- **Independência de Fornecedor** – por mais que um banco de dados utilize a mesma linguagem SQL, alguns comandos, tipos de dados, podem ser diferentes de um banco para outro.

O objetivo é construir um código mais uniforme e que utilize de fato as características da linguagem orientada a objetos. Sistemas que programem a maior parte da lógica da aplicação no banco de dados, não irão se beneficiar de mapeamentos objeto relacional, pois não há necessidade de um modelo de objetos para trabalhar a lógica no sistema (LINHARES, p. 01).

3 *Hibernate*

O *Hibernate* é uma ferramenta de mapeamento objeto relacional de grande aceitação entre os desenvolvedores de sistemas orientados a objetos. Esta é uma ferramenta gratuita e é considerada uma das mais utilizadas por especialistas da área, e por estes motivos ela foi adotada para o desenvolvimento deste artigo.

Toda a configuração do *Hibernate* é feita através de arquivos em XML, os quais contêm detalhes sobre o mapeamento de dados e detalhes sobre as conexões com bancos de dados. Uma nova versão do *Hibernate*, o *Hibernate Annotations* permite fazer anotações sobre o mapeamento em cada classe que se deseja mapear no sistema, substituindo assim os arquivos XML de mapeamento que cada classe deve possuir para realizar o mapeamento, exceto o arquivo de configuração do *Hibernate* (FERNANDES, 2005).

3.1 Funcionamento do *Hibernate*

De acordo com Bauer (2007, p.44) para que o mapeamento objeto relacional ocorra, o *Hibernate* precisa de informações de como as classes do sistema serão persistidas e recuperadas do banco de dados. Além de informações sobre como mapear as classes, o *Hibernate* precisa também de informações para sua configuração e quais arquivos de mapeamento serão

implementados. Estas informações são dados que serão escritos em arquivos XML (*EXtensible Markup Language*). XML é uma linguagem de marcação de dados que permite armazenar dados através de campos no formato de texto.

Para apresentar as etapas do funcionamento do *Hibernate*, foram desenvolvidos diagramas no padrão UML de desenvolvimento de sistemas. UML é um padrão de modelagem que indica as formas que podem ser utilizadas para representar um sistema em diversos estágios de seu desenvolvimento (MEDEIROS, pg.10). Os diagramas representados neste artigo são os de atividades e de classes.

O diagrama de atividades ilustrado pela figura 2 a seguir, mostra as atividades percorridas pelo sistema, para realizar o mapeamento objeto relacional de uma classe para o banco de dados. Todas as atividades mostradas no diagrama ocorrem através de linhas de comandos no código fonte do sistema:

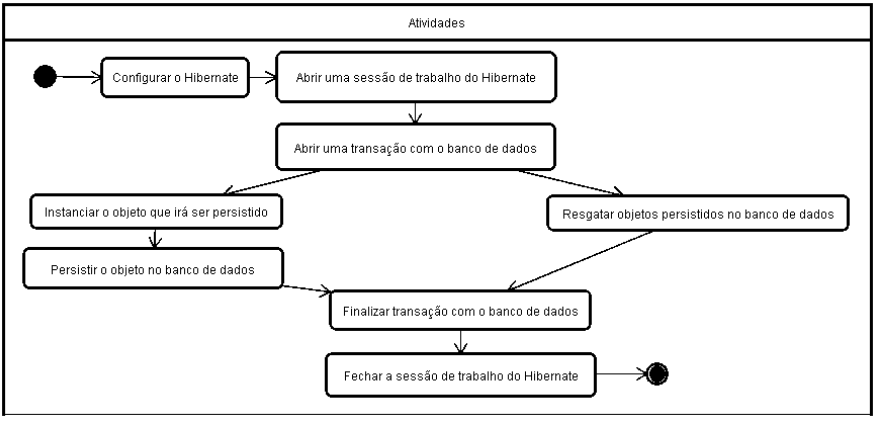

Figura 2. Diagrama de Atividades de um mapeamento objeto relacional, Adaptado de: (Bauer, 2007).

Configurar o *Hibernate*: sempre que iniciada uma classe que irá realizar um mapeamento objeto relacional, o sistema busca o arquivo XML que contém informações para realizar a configuração do *Hibernate*. Configurar o *Hibernate* é a primeira tarefa executada pelo sistema, pois ao ser configurado o *Hibernate* é informado sobre os dados da conexão com o banco de dados a ser utilizado, quais classes a serem persistidas, e outras funcionalidades.

Abrir sessão de trabalho do *Hibernate*: depois de ser configurado, o *Hibernate* abre uma sessão de trabalho. Uma sessão é uma classe que contém métodos utilizados para realizar o mapeamento objeto relacional.

Abrir uma transação com o banco de dados: é aberta uma transação com o banco de dados para que ocorram as operações com o banco de dados. Estas operações são solicitadas quando forem utilizados os métodos responsáveis por realizarem o mapeamento objeto relacional.

Instanciar classes: Caso o objetivo seja persistir um objeto, deve-se criar uma instância de sua classe, para que esta instância possa ser enviada para o método responsável pela persistência. Para maior portabilidade do sistema, as classes que deverão ser persistidas, devem evitar possuir dependência de bibliotecas.

Persistir o objeto: o objeto instanciado na atividade anterior é passado como parâmetro para o método responsável por persistir objetos. Ao ser enviado o objeto para a persistência, o *hibernate* busca os atributos da classe na qual pertence o objeto e cria um comando de inserção de dados, e o executa no banco de dados.

As atividades ocorridas no processo de persistência são ilustradas pela figura 3 a seguir:

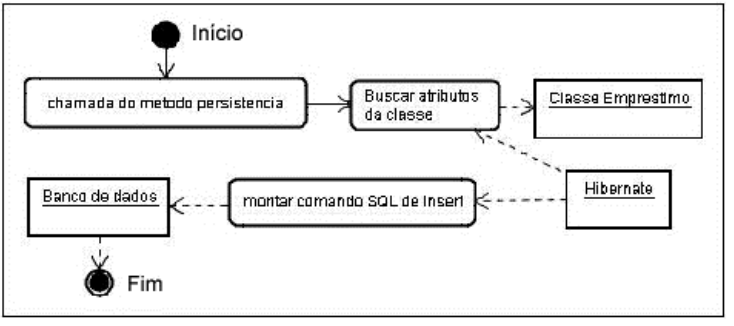

Figura 3. Atividades do método de persistência.

Resgatar um objeto: para resgatar um objeto o *Hibernate* utiliza sua linguagem própria de consulta a HQL (*Hibernate Query Language*). Para se resgatar um objeto do banco de dados é utilizada a API Criteria, que é a responsável por traduzir as execuções de uma representação HQL para o SQL do banco de dados.

Finalizar transação com o banco de dados: nesta etapa ocorre o encerramento de transações com o banco de dados. As transações que resultarem em erros, o banco de dados desfaz a transação e retorna uma mensagem de erro que deve ser tratada pelo sistema.

Fechar sessão de trabalho do *Hibernate*: assim como a transação, devemos fechar também a sessão de trabalho do *Hibernate*. Fechar a sessão de trabalho do Hibernate é importante, pois quando uma sessão é iniciada ela fica armazenada em memória, prejudicando assim o desempenho do sistema.

3.2 Mapeamento com *Hibernate Annotations*

De acordo com Bauer (2007, p.33), esta é uma das maneiras de se realizar um mapeamento objeto relacional. Este método trabalha com anotações feitas direto nas classes do código java. O *Hibernate Annotations* é um pacote que deve ser utilizado junto com o *Hibernate* para reduzir as linhas de códigos em comparação com os arquivos XML, pois ele substitui os arquivos XML de mapeamento de cada classe por anotações feitas dentro das classes, O *Hibernate Annotations* é a mais nova tendência de mapeamento objeto relacional, pois esta tecnologia quebra um pouco da complexidade do desenvolvimento dos arquivos XML utilizados para realizar o mapeamento.

4 Estudo de caso

Este estudo de caso tem como objetivo ilustrar o funcionamento e colocar em prática alguns conceitos sobre mapeamento objeto relacional vistos no decorrer deste trabalho. A ferramenta de mapeamento objeto relacional escolhida para trabalhar no desenvolvimento do sistema de biblioteca, é a e *Java Persistence API* (JPA), implementada pelo *Hibernate*.

A tecnologia JPA sendo implementada pelo *Hibernate* vem sendo uma das soluções de mapeamento objeto relacional mais utilizada, devido a sua facilidade de implementação comparada a complexidade da utilização do *Hibernate* com arquivos XML para mapear cada classe.

4.1 Mapeamento Objeto Relacional do sistema de biblioteca

O sistema utilizado para demonstrar o estudo de caso deve automatizar os empréstimos de livros e revistas de uma biblioteca. Para que o cenário escolhido possa ser implementado foram realizadas as etapas:

- Definição dos objetos do sistema: aluno, funcionário, livro, revista, empréstimo e devolução;
- Definição e detalhamento do estudo de caso: cadastrar alunos, funcionários, livros, revistas, efetuar empréstimos, definir data da devolução, efetuar devoluções e emitir relatórios;
- Definição das classes: pessoa, aluno, funcionário, título, livro, revista, empréstimo, devolução e *Hibernate*.

4.2 Diagrama de classes

O diagrama de classes é um dos diagramas que fazem parte do padrão UML já comentado na seção 3.1, ele representa a estrutura e relações das classes. Este modelo está organizado através de níveis de hierarquia onde as classes Aluno e Funcionário herdam atributos e métodos da classe Pessoa, assim como as classes Livro e Revista herdam os atributos e métodos da classe Título. As classes Empréstimo e Devolução estão relacionadas através de associação para interagir com as classes Títulos e Pessoa. O *Hibernate* foi representado no diagrama para ilustrar a interação de suas classes com as classes que irão ser mapeadas.

As simbologias utilizadas no diagrama de classes serão descritas na Figura 4 a seguir:

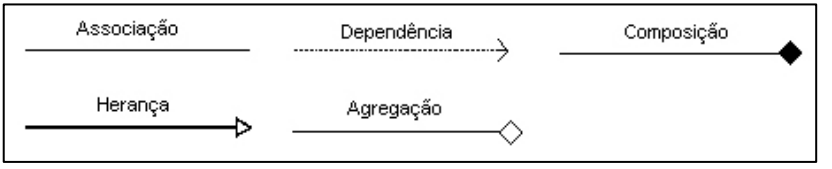

Figura 4. Simbologias de relacionamento de Classes.

Associação: são relacionamentos que especificam que objetos de uma classe estão ligados a objetos de outras classes.

Herança: relacionamento que especifica que uma classe denominada subclasse herda atributos e métodos da classe denominada superclasse.

Dependência: a dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe, uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente.

Agregação: é uma associação em que um objeto é parte de outro, de tal forma que a parte pode existir sem o objeto no qual ela pertence. Tipo de associação onde o objeto parte é um atributo do todo, por exemplo, Empréstimo (o todo) é composto por Títulos e Pessoa (as partes).

Composição: relacionamento onde um objeto contém uma lista de outros objetos. Os objetos contidos não fazem sentido fora do contexto do objeto que os contém, por exemplo, uma Devolução não faz sentido se não existir um Empréstimo.

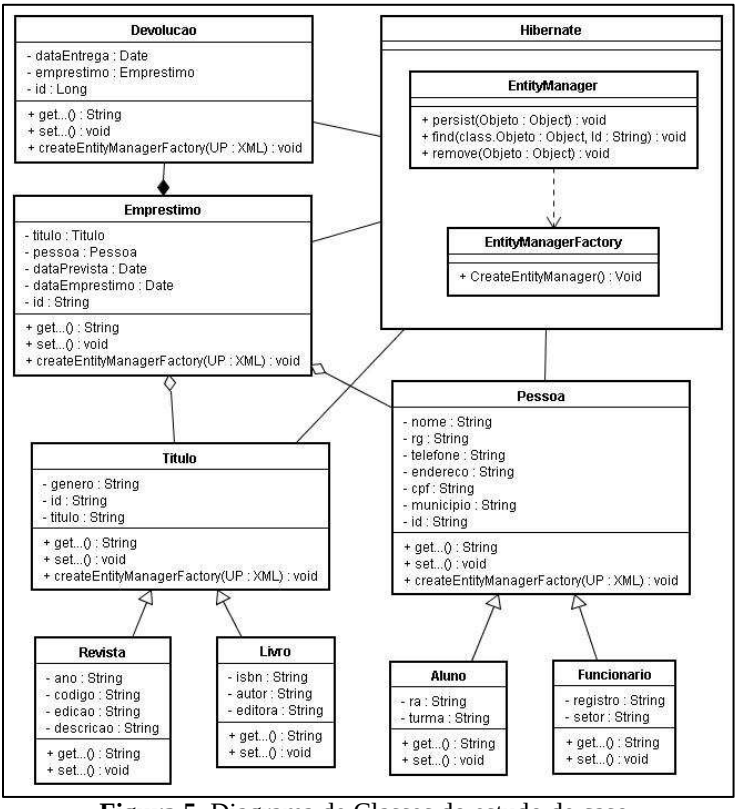

Figura 5. Diagrama de Classes do estudo de caso.

O método *createEntityManagerFactory* está representado em todas as classes que irão utilizar as funcionalidades do mapeamento objeto relacional. Este método recebe como parâmetro o nome da unidade de persistência a ser utilizada pelo *Hibernate* para realizar o mapeamento das classes. A unidade de persistência é um arquivo XML, que contém informações utilizadas para realizar a configuração do *Hibernate*. Configurar o *Hibernate* é necessário para que esteja especificado as informações do banco de dados utilizado e classes a serem mapeadas.

4.3 Modelo Classe - Tabela

O Modelo Classe–Tabela foi desenvolvido neste artigo, como sugestão do autor, para ilustrar os resultados de mapeamentos de heranças existentes no projeto da biblioteca. As metodologias utilizadas para mapear as classes do sistema para as tabelas no banco de dados podem parecer simples, mas quando há a necessidade de mapear heranças, existe uma dificuldade devido a uma visível incompatibilidade formal do modelo orientado a objetos e do modelo relacional. Dentre as metodologias existentes para realizar o mapeamento de heranças, foi escolhido para trabalhar neste projeto a abordagem de tabela por hierarquia de classe.

Tabela por hierarquia de classe consiste em mapear toda a hierarquia de classes para uma única tabela do banco de dados. O problema desta abordagem é a não aplicação da terceira forma normal no banco de dados. A não adoção da forma normal visa otimizar o sistema e não o banco de dados, uma vez que toda a lógica do funcionamento do sistema estará no aplicativo e não no sistema gerenciador de banco de dados. Segundo Bauer (2007, pg.199), esta é a melhor abordagem para sistemas que possuem uma modelagem da aplicação mais simples, como é o caso da modelagem deste projeto.

A Figura 6 a seguir, ilustra o resultado do processo de mapeamento de herança por hierarquia de classe. É possível observar na figura que a tabela recebe o nome referente a superclasse da hierarquia, e que cada subclasse é identificada por uma coluna chamada “Herdeira”, esta coluna é adicionada automaticamente pelo *Hibernate*, e é ela que identifica de qual subclasse pertence o objeto armazenado na tabela do banco de dados. A coluna Herdeira é utilizada pelo *Hibernate*, para poder identificar a qual objeto um registro do banco de dados pertence, permitindo assim resgatar somente as informações pertinentes a este objeto.

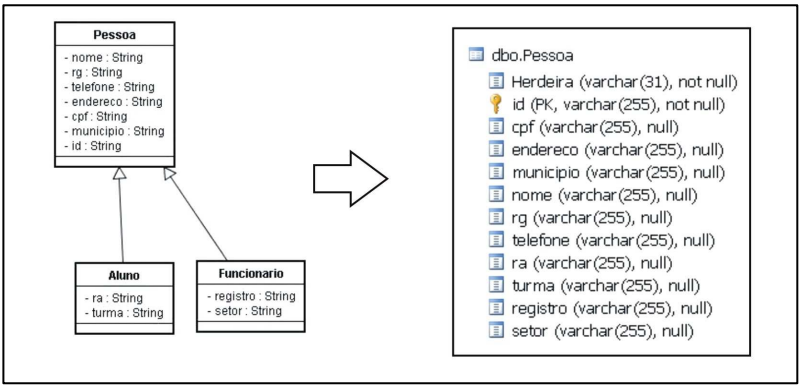

Figura 6. Mapeamento Herança de Pessoa.

Para realizar o armazenamento das informações sobre empréstimo, o sistema cria uma tabela Empréstimo que irá armazenar informações como as datas e as chaves que fazem a ligação desta tabela com as tabelas Pessoa e Título. Os atributos que fazem as ligações da tabela Empréstimo com as tabelas Pessoas e Titulo são respectivamente “id_pessoa” e “id_titulo”. Este modelo pode ser observado na Figura 7.

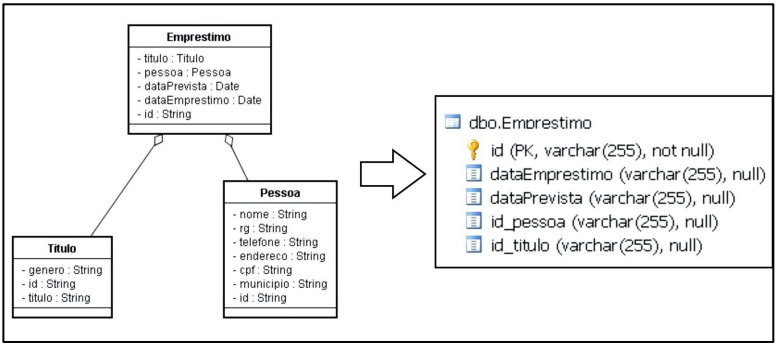

Figura 7. Mapeamento Relacionamento de Empréstimo.

O mapeamento da classe Devolução ilustrado na Figura 8, segue a mesma abordagem do mapeamento da classe Empréstimo, embora Empréstimo tenha uma associação de agregação com Pessoa e Título, a associação de composição que liga Empréstimo a Devolução tem o mesmo mecanismo de funcionamento. A tabela Devolução irá possuir o atributo

“id_emprestimo”, que irá trabalhar como chave de ligação com a tabela de Empréstimo.

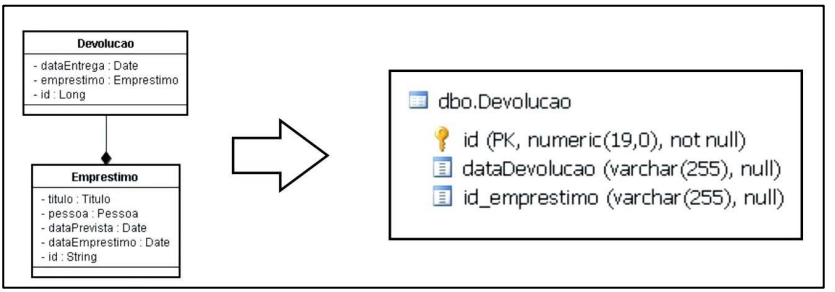

Figura 8. Mapeamento Relacionamento de Devolução.

5 Considerações Finais

As ferramentas de mapeamento objeto relacional surgiram para auxiliar no trabalho de desenvolvedores de *softwares* que utilizam linguagem de programação orientada a objetos para o desenvolvimento do sistema que adotem os bancos de dados relacionais para o armazenamento de dados.

Pode-se concluir ao realizar o desenvolver o projeto prático deste artigo que a idéia central é propiciar ao programador que ele reduza a utilização de comandos SQL em seu código, visando o desenvolvimento de um código mais enxuto e com menos probabilidade de erros, facilitando assim no desenvolvimento do sistema, e em futuras manutenções deste sistema. A ferramenta *Hibernate* substitui os comandos SQL por métodos que irão realizar todo o processo de armazenamento de dados, mas isso não significa que o programador não precise ter conhecimentos aprofundados em sistemas gerenciadores de banco de dados, muito pelo contrário, pois o desenvolvedor deve saber como que o mapeamento objeto relacional está sendo gerado dentro do banco de dados.

Ao realizar a implementação prática deste artigo, pode-se observar que ao realizar o processo de mapeamento de dados nos banco de dados SQL server e Postgree, os mesmo apresentaram algumas limitações quanto a agregações e composições, o que ocorre no banco de dados Oracle.

Mapeamento objeto relacional pode ser considerado uma tecnologia muito recente, que ainda necessita ser melhorado e principalmente explorado, pois existem muitas outras funcionalidades além das utilizadas neste artigo. Um trabalho futuro seria o aprofundamento nas funcionalidades do *Hibernate* e o tratamento de relacionamento entre objetos.

Referências Bibliográficas

BARTELS, Dirk .JDO- Java Data Objects, Disponível em :<http://www.mundooo.com.br/php/modules.php?name=News&file=article &sid=615> , Acesso em: 15 outubro 2007.

BAUER, Christian et al, *Java Persistence com Hibernate*, Rio de Janeiro: Ciência Moderna, 2007.

DOEDERLEIN, Osvaldo Pinali. *Dados e Mapeamento Explorando técnicas e tecnologias para persistência de dados.* **JAVA magazine**, Ed.42, ano.V, p. 22-30, 2006.

ESJUG, Tutorial *Hibernate* Básico, Disponível em: <www.inf.ufes.br/~vsouza/files/Tutorial*Hibernate*Basico.pdf>. Acesso em: 13 agosto 2007.

FERNANDEZ, Luiz Rafael . Construindo aplicações utilizando Thinlet e *Hibernate* - Parte 02, Disponível em : < http://www.imasters.com.br/artigo/3510/oracle/construindo_aplicacoes_utilizando_thinlet_e_*Hibernate*_-_parte_02/>, Acesso em: 17 outubro 2007.

LINHARES, Maurício. Introdução ao *Hibernate* 3, Disponível em : <http://www.guj.com.br/content/articles/*Hibernate*/intruducao_*Hibernate*3_guj.pdf>, Acesso em: 03 agosto 2007.

LOZANO, Fernando. *Persistência com Hibernate Conhecendo o Mais popular Framework O/R*. **JAVA magazine**, Grajaú, Ed.28, ano.IV, p. 18-28, setembro. 2005.

LOZANO, Fernando, Persistência Objeto Relacional com Java, Disponível em: <http://www.lozano.eti.br>. Acesso em: 03 outubro 2007.

MEDEIROS, Ernani, *Desenvolvendo software com UML definitivo 2.0,* São Paulo: Pearson.

SILVA, Edgar.A. *Dados na Web com o NetBeans 5.5*, **Java magazine**, Ed. 43, ano V, p. 36-39 , 2006.