ANÁLISE COMPARATIVA ENTRE FRAMEWORKS DE PERSISTÊNCIA DE DADOS

ALISON JOVELIANO FERRENHA – alisonjferrenha@gmail.com

Centro Universitário de Araraquara (Uniara)

Trabalho Orientado pelo Prof. Dr. Marcus Rogério de Oliveira

Resumo: Este artigo apresenta uma análise comparativa entre dois *frameworks* relacionados à persistência de dados usando *Hibernate* e *MyBatis*. A visão geral do trabalho é disponibilizar uma análise fiel e em tempo real da *performance* de cada uma destas ferramentas, evidenciando tais resultados através de uma pequena aplicação, desenvolvida exclusivamente para a análise em questão. A ideia para desenvolvimento do artigo surgiu da necessidade de escolha entre estas duas ferramentas para iniciar um projeto empresarial, e, devido à falta de artigos relacionados, assim proporcionar tais informações para desenvolvedores que estão em situações de decisão semelhantes.

Palavras-chave: Persistência de Dados, *Frameworks*, *Hibernate, MyBatis*, *Performance*

Abstract: This article presents a comparative analysis between two *frameworks* related to data persistence (*Hibernate* and *MyBatis)*. The overview of the work is to provide a reliable and real time performance analysis of each of these tools, showing such results through a small application, developed exclusively for the analysis in question. The idea for developing the article came from a personal need to choose between these two tools to start a business project, and due to lack of related articles, came true the idea of providing such information for developers who are in similar decision situations.

Keywords: Data Persistence, *Frameworks*, *Hibernate, MyBatis, Performance.*

1 INTRODUÇÃO

1.1 Apresentação e Delimitação do Tema

Devido à oposição de fluxo de dados existente entre o modelo Orientado a Objetos, mais utilizado no desenvolvimento de software, e o modelo Relacional dos Bancos de Dados, surgiram técnicas de mapeamento objeto-relacional (ORM) que permitem melhor interação entre os dois modelos. Contudo, existem várias especificações e frameworks de mapeamento objeto-relacional que implementam as técnicas de ORM, por isso se faz necessário decidir qual implementação utilizar ao

desenvolver um software. É notável o crescimento acelerado da indústria de software, devido principalmente às demandas crescentes do mercado por soluções inovadoras. A informação se tornou um artefato de extrema valia para a corporação moderna, os bancos, por exemplo, gravam constantemente registros dos dados financeiros de milhões de pessoas e empresas. A manutenção e o gerenciamento das informações contidas em seus bancos de dados são um dos pilares da área de Sistemas de Informação. As corporações armazenam grandes volumes de informações em Bancos de Dados Relacionais, e estes, surgidos por volta da década de 70, são baseados em fundamentos matemáticos que garantem uma ótima estruturação e confiabilidade aos dados armazenados. Em contrapartida, na década de 90, a indústria de software viu crescer o paradigma de orientação a objetos, que permitia maior abstração aos desenvolvedores de software, e por conta disso diminuía o tempo necessário para a construção de sistemas, que por sua vez eram mais robustos e rentáveis.

Até o final da década de 90, a maioria absoluta de empresas dentro da indústria de software já utilizava o paradigma de orientação a objetos, porém estas empresas desenvolviam software para corporações que, como dito anteriormente, utilizavam-se de bancos de dados relacionais para armazenar os dados. O Modelo Orientado a Objetos (OO) é muito diferente do Modelo Entidade/Relacionamento (ER) utilizado pelos bancos de dados relacionais. Enquanto o primeiro é baseado em técnicas de Engenharia de Software, o outro é fundamentado em princípios matemáticos de relacionamentos entre tabelas. A esta diferença entre os modelos se dá o nome de Impedância Objeto-Relacional (AMBLER, 2003a).

1.2 **Objetivo**

Este artigo tem por objetivo realizar uma análise comparativa dos frameworks de persistência, evidenciar o desempenho dos mesmos, apresentar técnicas de mapeamento, comparar a produtividade, detalhar suas especificações e apontar os pontos fracos e pontos fortes.

1.3 **Justificativa**

Analisando os sistemas desenvolvidos nos dias de hoje, é visível a necessidade de armazenamento de dados presente, sejam eles sistemas organizacionais, empresariais, pessoais, ou até mesmo acadêmicos. Neste sentido,

é necessário avaliar as ferramentas disponíveis para que este armazenamento e gerenciamento sejam realizados constantemente, de forma segura e performática, ou seja, buscando explorar os melhores recursos disponíveis no mercado.

Quando se trata de persistência de dados, os frameworks (abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica, auxiliando o desenvolvimento de software) se tornam necessários, e, graças à grande diversidade desses arcabouços, é possível optar por aquele que mais atende às necessidades do projeto.

Este artigo traz comparações entre os frameworks Hibernate e MyBatis, descrevendo detalhes importantes na hora de escolher tal recurso, como por exemplo: performance, complexidade de implementação, situações viáveis e inviáveis, entre outras informações relevantes na construção de um projeto.

O intuito do trabalho não é induzir à escolha de um determinado framework, mas sim evidenciar as características de cada um, de modo que seja possível tomar a decisão correta (no sentido de melhor custo-benefício), baseando-se nas informações descritas, pois o melhor recurso em uma determinada situação pode não ser assim caracterizado em outra circunstância.

1.4 Problema e hipótese da pesquisa

Existe um grande vício por parte das equipes no momento de escolher o framework mais adequado para o projeto, fazendo com que a mesma escolha um framework o qual já foi utilizado em um projeto anterior, ou aquele que é mais comentado nas comunidades da internet, o famoso “todo mundo usa”.

A análise comparativa fornecerá informações importantes, as quais mostram os valores reais de cada um destes frameworks, suas vantagens e desvantagens , como e quando devem ser utilizados, situações onde deve ser evitado o uso de determinados recursos oferecidos e estatísticas de uso ao redor do mundo.

Na prática, é notável a falta de pesquisa dos líderes quando o assunto é a busca por uma ferramenta mais adequada, os padrões estipulados muitas vezes fogem do princípio das boas práticas, e, o comodismo em utilizar recursos já presentes em outro momento não permite que a execução do projeto tenha um nível de qualidade superior.

1.5 Metodologia

O estudo realizado pode ser caracterizado como de natureza tecnológica, focado em apresentar os conceitos de Banco de Dados e Mapeamento Objeto Relacional, principalmente.

As informações presentes são resultado de observações, análise de dados obtidos através de constatações bibliográficas e testes realizados, incluindo levantamento de hipóteses e problemas.

Serão abordadas pesquisas bibliográficas, se utilizando de referência nomes como Java Persistence com Hibernate (Bauer e King), JPA Eficaz (Hebert Coelho), Hibernate in Action, entre outros.

As análises descritas no artigo envolvem desempenho, documentação, fácil acesso à documentação, ciclo de vida, mapeamento de classes, relacionamentos, gerenciadores de bancos de dados e controle dos esquemas destes bancos.

É analisado também o desempenho das combinações, paradigmas de bancos de dados, as tendências para o futuro, o porque da supervalorização do Hibernate (framework o qual se tornou um dos mais maduros e estáveis no mercado).

2 REVISÃO BIBLIOGRÁFICA

2.1 Hibernate

O Hibernate é um framework de persistência de dados, seguindo o conceito de mapeamento objeto-relacional, que visa ser uma solução eficiente e eficaz para o problema de gerenciamento de dados persistentes em Java. (BAUER e KING, 2007).

A utilização desse framework facilita o armazenamento e recuperação de objetos em Java com seu mapeamento objeto-relacional. Também trabalha com o modelo de objetos POJO que segundo Shaefer e Isaia (2006), são objetos Java, sem nenhuma lógica, contendo apenas atributos e seus respectivos métodos de acesso (*getters* e *setters*). Tratando-se de suporte a banco de dados, oferece um suporte considerável, e, mesmo que este não seja possível por algum motivo, existe a possiblidade da utilização de um driver para sanar tal problema.

Foi desenvolvido por uma equipe de programadores Java, tendo sua primeira versão divulgada em 2004. Posteriormente, JBoss Inc (empresa adquirida pela Red Hat) contratou os principais programadores para se juntar ao seu suporte. Segundo King, a intenção de criar o projeto se deu na tentativa de resolver problemas referentes à persistência, causados por outras ferramentas, e, considerados por ele muito complexos. Atualmente, é encontrado como software livre para as plataformas Java e .NET, podendo ser utilizado gratuitamente sob a licença LGPL (Lesser GPL). Só é possível estabelecer conexões com bancos de dados relacionais, mapeando as entidades na aplicação (objetos correspondentes às tabelas da base de dados).

Tornou-se a implementação mais utilizada para a especificação JPA (Java Persistence API), recebendo diversas melhorias com o passar do tempo, pela própria equipe do Hibernate, que são distribuídas na forma de plug-ins, como por exemplo Hibernate Validator, Hibernate OGM, entre outros. Seu principal objetivo se dá em substituir cerca de 90% das tarefas repetitivas da programação envolvendo persistência, evitando que eles gerem códigos SQL manualmente.

A partir da versão 3.5, foi declarado como uma implementação certificada para a especificação JPA2 (JSR 317), que se consagrou oficial em 2009. Com a evolução da ferramenta, surgiram novos recursos e muitos dos existentes foram aprimorados. A decisão de utilizar o Hibernate ao invés de outros frameworks envolve vários fatores, mas, principalmente pelo fato de ser o framework ORM mais presente no mercado e implementar a especificação Java EE para persistência, o JPA.

2.2 **MyBatis**

MyBatis é um framework *open source* de persistência de dados, desenvolvido em 2002 por Clinton Begin, e tem como objetivo simplificar a implementação da camada de persistência, abstraindo códigos JDBC e oferece uma API simples e de fácil interação com bancos de dados, podendo ser uma alternativa tanto ao JDBC quanto ao Hibernate.

Era conhecido como iBatis, sob licença da Apache, e após esse período, migrou para o Google Code, onde teve seu nome alterado para MyBatis, porém, manteve a configuração (exceto pelo schema dos arquivos de configuração).

Dispõe de uma performance otimizada, como suporte à pool de conexões (que impedem a criação de uma nova conexão à cada requisição) e mecanismo de cache para consultas. Oferece suporte à SQL customizado, stored procedures, mapeamento avançado e integração com frameworks como Spring, Guice, e bibliotecas de cache de terceiros (mesmo possuindo suporte para cache embutido). Pode ser configurado através de annotations ou XML e tem boa integração com bancos de dados legados, o que é um diferencial interessante, pois a dificuldade de se trabalhar com bancos legados utilizando, por exemplo, frameworks ORM, é muito grande, devido ao mapeamento entre a base de dados e os objetos da aplicação.

Dentre as principais razões da popularidade do MyBatis, uma delas se trata da simplicidade de aprendizagem e uso, tornando-se ainda mais simples caso o desenvolvedor tenha conhecimento de Java e SQL. Fornece também gerenciamento de transações para operações no banco de dados, porém, caso houver outro gerenciador disponível, delegará o gerenciamento à este.

2.3 **MySQL**

O SGBD (Sistema Gerenciador de Banco de Dados) utilizado para armazenar os dados utilizados nas comparações é o MySQL. Trata-se de um banco de dados relacional, o qual armazena os dados em tabelas separadas, oferecendo recursos para tornar o ambiente de desenvolvimento flexível.

É uma ferramenta open source, o que significa que é possível utilizar e também modificar o programa. Qualquer usuário pode baixar o software do site oficial, sem necessidade de licença.

Foi criado na Suécia pela empresa MySQL AB, empresa que veio a ser adquirida pela Sun Microsystems pelo valor de U$1 bilhão, e, nos dias de hoje, se tornou propriedade da Oracle. Seu objetivo era atender aplicações de pequeno/médio porte (em torno de centenas de milhões de registros por tabela, o equivalente a centenas de megabytes por tabela), porém, com a evolução decorrente, passou a comportar aplicações de grande porte.

Oferece portabilidade, multithreading, caching em consultas, criptografia, e utiliza o mecanismo MYISAM, o qual possui um método de armazenamento consideravelmente veloz, eliminando apenas o suporte à transações. É possível

também adquirir gratuitamente a ferramenta MySQL Workbench no site oficial (www.mysql.com), a qual auxilia na manutenção da base de dados, na manipulação dos registros e facilita a identificação de problemas e inconsistências. A versão 6.3 é a versão utilizada no desenvolvimento do trabalho.

3 DESENVOLVIMENTO

3.1 IDE

A IDE (*Integrated Development Environment*) é a ferramenta utilizada para desenvolver a aplicação, envolvendo plug-ins, bibliotecas e pacotes que auxiliam no processo. Nela podemos encontrar compiladores (recurso onde o código-fonte é interpretado e transformado em código binário para leitura do sistema operacional), editores de código, gerenciadores de dependências, console para logs, ou seja, tudo que a ferramenta pode oferecer ao desenvolvedor para auxiliá-lo no desenvolvimento, tornando-o produtivo e de qualidade.

No caso deste artigo, a IDE utilizada foi o Eclipse, pois se trata de uma ferramenta completa, eficiente, e que oferece diversas funcionalidades e atalhos, auxiliando na evolução do projeto. Possui vários recursos úteis tanto no aumento da produtividade quanto na organização das equipes de desenvolvimento e testes.

3.2 Contexto

O objetivo do desenvolvimento é levantar a forma de execução de ambos os frameworks Hibernate e MyBatis, como se comportam em determinadas operações, os resultados obtidos e a performance obtida durante a execução destas operações.

Os testes foram aplicados em uma máquina Intel i5 4210U, 1.7GHz, 6GB RAM, 500GB HDD e Sistema Operacional Linux Fedora 22.

Foram realizadas operações idênticas em ambos os frameworks e extraídas as informações de tempo gasto para cada uma destas operações. Lembrando que os resultados podem variar de acordo com vários fatores como: máquina utilizada, memória disponível no momento da execução, volume de registros no banco de dados, versão, configuração e parâmetros de inicialização da máquina virtual do java (JVM), entre outros.

3.3 Configuração

A configuração dos frameworks foi realizada por arquivos XML, mais especificamente, “sqlmapconfig.xml” (MyBatis) e hibernate.cfg.xml (Hibernate). As configurações envolvem: o endereço de acesso ao banco de dados, o dialeto ao ser utilizado (MySQL, neste caso), usuário e senha, entre outras configurações como é possível visualizar nas imagens abaixo:

```
<hibernate-configuration>
    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/tcc</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="show_sql">false</property>
        <property name="format_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">none</property>
        <property name="hibernate.jdbc.batch_size">20</property>
        <property name="hibernate.cache.use_second_level_cache false">false</property>
        <property name="order_inserts">true</property>
        <property name="hibernate.jdbc.batch_versioned_data">true</property>

        <!-- Names the annotated entity class -->
        <mapping class="br.com.artigo.tcc.entidades.Employee" />
        <mapping class="br.com.artigo.tcc.entidades.Department" />
        <mapping class="br.com.artigo.tcc.entidades.Job" />
        <mapping class="br.com.artigo.tcc.entidades.ClienteCrud" />

    </session-factory>

</hibernate-configuration>
```

Figura 1 – Configuração realizada no arquivo hibernate.cfg.xml para funcionamento do framework Hibernate. FONTE: Próprio Autor

```xml
<configuration>

    <typeAliases>
        <typeAlias alias="Employee" type="br.com.artigo.tcc.entidades.Employee"/>
        <typeAlias alias="Job" type="br.com.artigo.tcc.entidades.Job"/>
        <typeAlias alias="Department" type="br.com.artigo.tcc.entidades.Department"/>
        <typeAlias alias="ClienteCrud" type="br.com.artigo.tcc.entidades.ClienteCrud"/>
    </typeAliases>

    <environments default="development">
        <environment id="development">
          <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/tcc"/>
                <property name="username" value="root"/>
                <property name="password" value=""/>
            </dataSource>
       </environment>
    </environments>

    <mappers>
       <mapper resource="br/com/artigo/tcc/mybatis/Employee.xml"/>
       <mapper resource="br/com/artigo/tcc/mybatis/ClienteCrud.xml"/>
    </mappers>

</configuration>
```

Figura 2 – Configuração realizada no arquivo sqlmapconfig.xml para funcionamento do framework MyBatis. FONTE: Próprio Autor

3.4 **Consultas**

Neste tópico foram abordadas as operações de consulta ao banco de dados (selects). Uma consulta pode ser simples, retornando, por exemplo, todos os registros e todos os campos da tabela Empregados, utilizando o seguinte comando:

SELECT * FROM Empregados;

A consulta pode se tornar complexa conforme a query se estende, adicionando selects internos, funções variadas (soma, média, substituição) e joins (junções) com outras tabelas, como é o caso do segundo exemplo de consulta realizado:

SELECT (e.salary-1000), AVG(e.salary), REPLACE(e.full_name, 'a', 'o')

FROM employees e INNER JOIN department d ON e.department_id = d.department_id INNER JOIN job j ON d.department_id = j.department_id

WHERE e.full_name like 'A%'

AND e.age BETWEEN 30 AND 45

```
AND d.department_name LIKE '%a%'

AND j.job_title LIKE '%a%'

AND j.max_salary <= 25000;
```

O resultado obtido na primeira execução (consulta de 10.000 registros utilizando o primeiro exemplo de comando citado), levando em conta o tempo de execução, foi de 0.7 segundos para o MyBatis e de 1.6 segundos para o Hibernate.

Na segunda execução, o tempo de execução foi de 0.01 segundos para o Hibernate e 0.06 segundos para o MyBatis. Se observarmos a diferença de tempo entre a primeira e a segunda execução, o Hibernate teve uma melhor performance, e, isso acontece devido ao recurso de cache, onde os registros retornados na primeira consulta permanecem em memória, e, ao identificar que o comando executado é o mesmo, ocorre apenas uma consulta à estes registros na memória, e não mais no banco de dados. De acordo com as estatísticas, o Hibernate fez um melhor trabalho ao utilizar o recurso de cache.

Já no segundo exemplo de comando citado, o framework Hibernate atingiu um tempo de execução de 1.3 segundos, contra 0.2 segundos do concorrente MyBatis. Na segunda execução, a ferramenta ORM atingiu 0.001 segundos, contra 0.06 segundos do MyBatis.

É notável o ganho de performance do Hibernate quando se trata da tecnologia de cache, porém, quando uma aplicação tem a necessidade de executar comandos diferentes constantemente, não é possível utilizar os benefícios do cache, pois a consulta deve ser realizada diretamente no banco de dados, e, para ambos os cenários de consulta, o MyBatis demonstrou melhor desempenho nas primeiras execuções dos comandos.

3.5 **Inserções**

Foram realizadas operações de inserção simples, inserindo 1000 registros em ambos os casos, com todos os campos vazios. Para o Hibernate foi utilizado o método save de sua própria API, o qual executa um insert implicitamente, como é possível visualizar abaixo:

```
Session session = HibernateUtils.getSession();
session.beginTransaction();

try {
    for (int i=0; i<1000; i++) {
        session.save(new Employee());
    }

}

finally {
    session.getTransaction().commit();
    session.close();
}
```

Figura 3 – Código do método responsável por inserir 1000 registros na base utilizando o Hibernate. FONTE: Próprio Autor

Para o MyBatis foi utilizado o comando INSERT, o qual foi mapeado no arquivo Employee.xml, onde devem ficar todos os comandos referentes à esta entidade (selects, updates, inserts e deletes), no caso, Employee:

```
<insert id="inserir" parameterType="Employee">
INSERT INTO employees (PHONE, FULL_NAME, ADDRESS, EMAIL, ZIP_CODE, SALARY, MANAGER_ID, JOB_ID, DEPARTMENT_ID)
    VALUES (#{phone}, #{fullName}, #{address}, #{email}, #{zipCode}, #{salary}, #{managerId}, #{jobId}, #{departmentId});
  <selectKey keyProperty="id" resultType="int" order="AFTER">
    select last_insert_id() as id
  </selectKey>
</insert>
```

Figura 4 – Código INSERT mapeado no arquivo Employee.xml (arquivo de configuração de comandos SQL referentes à entidade Employee). FONTE: Próprio Autor

O resultado obtido na primeira execução (inserção de 1.000 registros) para o Hibernate foi de 0.8 segundos, e, para o MyBatis, 1.1 segundos. Na segunda execução (utilizando exatamente os mesmos comandos), o Hibernate atingiu 0.7 segundos e o MyBatis alcançou 0.8 segundos.

Ao analisar estes resultados, pode-se notar que a primeira inserção tem um tempo maior em ambos os casos, porém, o Hibernate teve melhor desempenho nas duas execuções, mesmo sendo perceptível uma evolução maior do MyBatis entre a

primeira e segunda execução. Não foram abordadas as operações de alteração e remoção, pois seguem o mesmo conceito da inserção, persistência, enquanto a consulta engloba um cenário diferente, o retorno de registros presentes na base de dados.

4 CONCLUSÃO

Neste trabalho, o assunto abordado se trata da comparação entre ferramentas de persistência de dados, onde suas respectivas performances foram colocadas em análise, e, evidenciadas através de resultados em tempo real, seguindo os padrões de comparação presentes nos trabalhos utilizados como referências para este artigo. O trabalho pode ser de valia para qualquer desenvolvedor/analista/arquiteto que esteja à procura de uma ferramenta qualificada para exercer a função citada.

Conclui-se que não há uma verdade absoluta, como por exemplo, caracterizar uma das duas ferramentas abordadas sendo melhor que a outra, ou vice-versa, o que se pode afirmar é que existem situações favoráveis para o uso de uma ou de outra, adequando-se à necessidade do projeto em questão.

O objetivo de analisar as performances especificamente em determinados tipos de operações (consulta simples, consultas com junções e inserções) foi atingido, porém, não podendo assumir que os mesmos resultados sejam obtidos nas mesmas proporções em outros ambientes, uma vez que os resultados se baseiam não só nas funcionalidades e recursos das ferramentas, mas também nos quesitos processamento, memória, sistema operacional, entre outros fatores contribuintes.

O trabalho é muito importante para o conhecimento de pessoas interessadas e que se utilizam dessas tecnologias, pois pode servir como base na decisão de escolha entre as mesmas, em situações com necessidades diferenciadas, conforme citado durante o artigo. O aprofundamento deste tema pode ser realizado de diversas maneiras, incluindo integrações com outras tecnologias, complementando toda uma aplicação, visto que a persistência de dados é realizada, geralmente, em aplicações web e desktop, independentemente da linguagem.

REFERÊNCIAS

AMBLER, S.W. **Agile Database Techniques: Effective Strategies for the Agile Software Developer**, 1st Edition, Ambysoft Inc., 2003a.

BAUER, C.; KING, G. **Java Persistence com Hibernate,** 1º ed., Editora Ciência Moderna Ltda., 2007.

BAUER, C.; KING, G. **Hibernate in Action,** First Edition, Manning Publications Co., 2004.

COELHO, Hébert. **JPA Eficaz: As melhores práticas de persistência de dados em Java,** Editora Casa do Código, 2009.

NEVES, Édina M.N; PACHER, Thyago H. **Análise e Comparação de Frameworks de Persistência.** Disponível em: <https://oldwolf1602.wordpress.com/2013/07/28/analise-e-comparacao-de-frameworks-de-persistencia>. Acesso em: 25 de set. de 2015.

JÚNIOR, Aurélio V.R; OLIVEIRA, Edson A.J. **Análise de Desempenho dos Frameworks de Persistência Hibernate e EclipseLink.** Disponível em: <http://www.espweb.uem.br/wp2/wp-content/themes/espweb/files/tcc/2012/Aurelio%20Vargas%20Ramos%20Junior%20-%20Analise%20de%20Desempenho%20dos%20Frameworks%20de%20Persistencia%20Hibernate%20e%20EclipseLink.pdf>. Acesso em: 30 de set. de 2015.

REDDY, S.; P.; K. **Java Persistence with MyBatis 3,** Packt Publishing, 2013.

SHAEFER, Henrique; ISAIA, Victor Ferreira. **Desenvolvimento das Camadas Lógicas e Persistência do Projeto SIAP-F.** Disponível em: <http://projetos.inf.ufsc.br/arquivos_projetos/projeto_316/artigo_tcc.pdf>. Acesso em: 25 de set.de 2015.