9º Simpósio da Pós-Graduação
INSTITUTO FEDERAL Sul de Minas Gerais
ISSN 2319-0124

MAPEAMENTO OBJETO-RELACIONAL: um comparativo de performance entre frameworks

Gleyson RIBEIRO[1]; Alessandro C. BORGES[2]

RESUMO

Considerando a técnica de mapeamento objeto-relacional (ORM), o presente artigo tem como objetivo realizar um comparativo de performance entre alguns frameworks de persistência de dados. Foram usados os frameworks Entity Framework Core, Fluent NHibernate e Dapper por estarem entre os mais utilizados no desenvolvimento de aplicações. Foram desenvolvidos projetos no Microsoft Visual Studio Community 2017 para os testes, com pequenas e com grandes quantidades de dados. Evidenciou-se, pelos dados e gráficos expostos, uma clara diferença performática entre os frameworks, mostrando a necessidade de estudo prévio para decidir a ferramenta apropriada em cada projeto específico.

Palavras-chave:
ORM; Entity-Framework Core; Fluent NHibernate; Dapper.

1. INTRODUÇÃO

Segundo Bauer e King (2007, p. xxi), enquanto as modernas linguagens de programação fornecem uma visão orientada a objetos das entidades de negócio em nível de aplicativo, os dados corporativos correspondentes a estas entidades são fortemente relacionais por natureza.

O Mapeamento Objeto-Relacional (ORM) cria uma camada de mapeamento entre o modelo de objetos (aplicação) e o modelo relacional (banco de dados) abstraindo o acesso ao mesmo – o programador não precisa escrever instruções SQL. Frameworks ORM ajudam desenvolvedores nas tarefas de persistência e recuperação de dados oferecendo-lhes maior produtividade (SATO, 2013).

Este artigo apresenta uma comparação de performance, com gravação e leitura de dados, entre os frameworks Entity Framework Core, Fluent NHibernate e Dapper, softwares livres que podem ser utilizados para criação de diversos tipos de sistemas, incluindo aplicações Web.

2. FUNDAMENTAÇÃO TEÓRICA

A seguir, uma contextualização sobre ORM e características dos frameworks comparados.

2.1. Mapeamento Objeto-Relacional

O ORM permite o mapeamento entre as classes de domínio da aplicação e o modelo relacional que trata da persistência dos dados. O desenvolvedor não necessita interagir com o banco de dados.

1Aluno IFSULDEMINAS – *Campus* Passos. E-mail: gleysonribeiromg@gmail.com.
2 Orientador, IFSULDEMINAS – *Campus* Passos. E-mail: alessandro.borges@ifsuldeminas.edu.br.

2.2. Entity Framework Core

Versão leve, extensível, de software livre e multiplataforma da tecnologia de acesso a dados do Entity Framework (MILLER, 2016). É um aprimoramento do ADO (Activex Data Objects) .Net, que fornece um mecanismo automatizado para acesso e armazenamento no banco de dados.

De acordo com Miller (2016), o acesso a dados é executado usando um modelo composto por classes de entidade e um objeto de contexto que representa uma sessão com o banco de dados.

2.3. NHibernate

ORM de código aberto desenvolvido para o Microsoft .Net Framework baseado no Hibernate, que é um ORM bastante popular para a plataforma JAVA (SIMAS, 2013). Entre as definições de classe e o esquema do banco de dados, há os chamados metadados de mapeamento que informam ao NHibernate como referenciar as classes em C# para o banco de dados.

2.4. Dapper

Um ORM de código aberto que fornece suporte para associação de objetos usando transações, procedimentos armazenados ou inserções de dados em massa (KANJILAL, 2017). A ideia é oferecer métodos de extensão para a interface IDBConnection do ADO .Net (WILLIAN, 2012).

Trabalha com qualquer provedor de banco de dados através de três processos (Dapper, s.d.):

- Criação de um objeto do tipo “IDBConnection”;
- Escrita do código para inserção, leitura, atualização e exclusão de registros;
- Passagem do código escrito anteriormente como parâmetro para o método “Execute”.

3. MATERIAL E MÉTODOS

O teste de performance dos frameworks propostos consistiu em gravações e leituras de registros de clientes em um banco de dados por projetos no Microsoft Visual Studio Community 2017. Para cada framework foi desenvolvido um projeto para evitar interferências entre as bibliotecas. Foram três projetos do tipo ‘aplicação de console’ utilizando a linguagem de programação C#.

O sistema gerenciador de banco de dados utilizado foi o Microsoft SQL Server versão 2014. No banco de dados, foi criada uma tabela chamada “Cliente” com os atributos: nome, data de nascimento, telefone, endereço, bairro, cidade, estado e CEP.

Em cada projeto foi criada a classe “Cliente” de modo que o framework fizesse o mapeamento entre a classe e a tabela no banco de dados.

Foi registrado o tempo decorrido na gravação e na leitura dos dados no banco e, para tanto, foi utilizada a classe do Microsoft .Net Framework denominada “Stopwatch”.

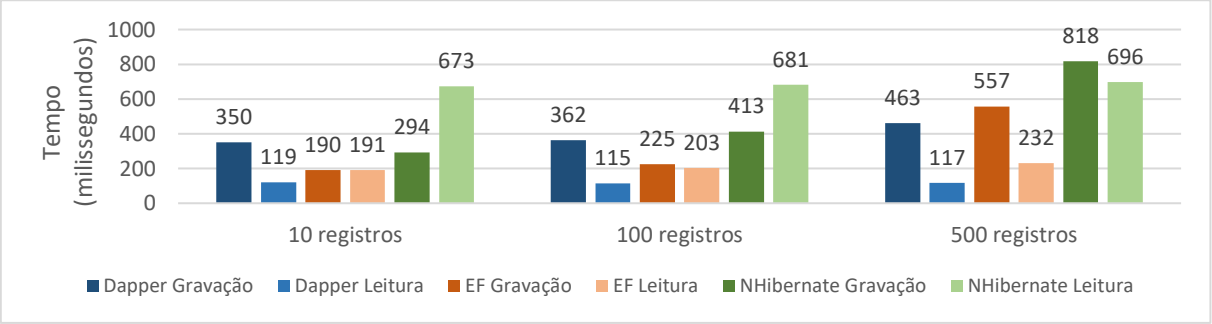
Foram utilizadas diferentes quantidades de registros para o teste, desde pequenas quantidades, como 10 registros, até grandes volumes, como 50 mil registros, com informações diferentes geradas aleatoriamente para cada cliente através da classe “Random”. Para cada quantidade de registro e para

cada framework foi calculada a média de tempo entre dez operações de inserção e consulta de registros. Foram realizados no total 360 testes envolvendo todos os frameworks.

4. RESULTADOS E DISCUSSÕES

O Gráfico 1 mostra o resultado obtido na gravação e na leitura de pequenas quantidades de registros no banco de dados. O Entity Framework Core apresentou performance melhor nas gravações de poucos registros, porém o Dapper apresentou melhor performance na leitura.

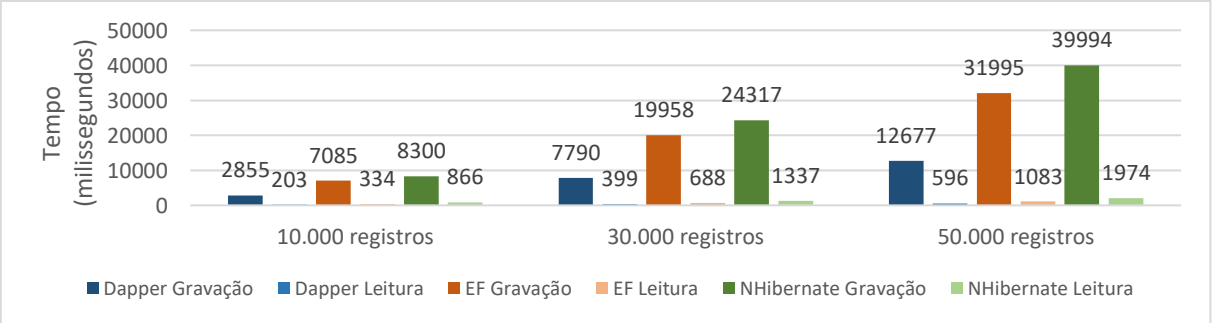
Gráfico 1. Resultado obtido na gravação e leitura de pequenas quantidades de registros



Fonte: do autor

No Gráfico 2 é mostrado o resultado obtido na gravação e leitura de grandes quantidades de registros no banco de dados. O framework Dapper mostrou performance bastante superior.

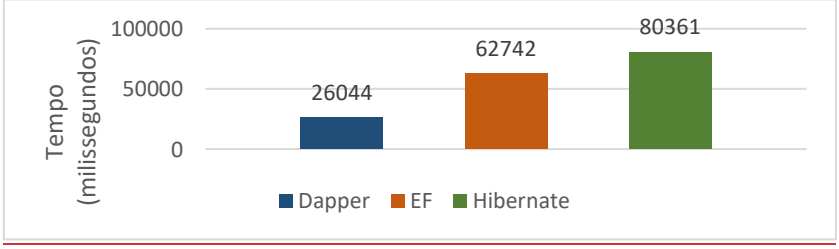
Gráfico 2. Resultado obtido na gravação e leitura de grandes quantidades de registros



Fonte: do autor

O Gráfico 3 exibe o somatório de todos os tempos, incluindo todas as quantidades de registros e considerando a gravação e a leitura dos dados. O Dapper obteve uma performance bem melhor em relação aos outros frameworks levando em consideração os testes realizados, ainda que o gráfico 1 mostre que perde para o Entity Framework Core na gravação de um número pequeno de registros.

Gráfico 3. Resultado obtido na leitura de pequenas quantidades de registros



Fonte: do autor

5. CONCLUSÕES

Com os testes efetuados entende-se que o framework Dapper só é menos performático frente ao Entity Framework Core na gravação de pequenas quantidades de dados, (desempenho inferior com dez e cem registros). Pôde-se constatar, todavia, que o Dapper consegue ser mais rápido nas operações de gravação a partir de quinhentos registros e, na gravação de grandes quantidades de registros, mostra-se bastante superior aos outros frameworks.

Considerando o resultado obtido na leitura de dados, chama atenção o fato de que o NHibernate leva um tempo consideravelmente superior para concluir o processo. Em todas as quantidades de registros testadas o Dapper tem um desempenho superior.

Conforme Brito (2018), o Dapper foi criado para resolver problemas de performance do sítio eletrônico "Stack Overflow", um dos endereços mais utilizados por desenvolvedores de software na Internet. Desde o seu surgimento, portanto, este framework teve como um dos seus pilares principais ser performático. É um framework mais enxuto, pois não possui ferramentas como, por exemplo, geradores de modelo de classes e geradores de consultas, entretanto tem menor facilidade de uso, pois há necessidade de se escrever determinadas instruções em SQL (*Structured Query Language*).

O mapeamento objeto-relacional é uma ferramenta bastante valiosa e o Dapper mostrou-se uma excelente escolha como ferramenta de mapeamento objeto-relacional nos projetos de desenvolvimento de software.

REFERÊNCIAS

BAUER, C.; KING, G. *Java Persistence with Hibernate.* NY. Manning Publications. 2007.

Dapper Tutorial. (s.d.). *Dapper*. Disponível em https://dapper-tutorial.net/dapper. Acesso 05 de Abril de 2019.

BRITO, Bruno. *Dapper em Detalhes*. MVP Bruno Brito. 19 de Julho de 2018. Disponível em: https://www.brunobrito.net.br/dapper-em-detalhes Acesso em 08 de Outubro de 2020.

KANJILAL, Joydip. *How to use Dapper ORM in C#.* Infoworld. 2017. Disponível em https://www.infoworld.com/article/3025784/how-to-work-with-dapper-in-c.html. Acesso em 05 de Abril de 2019.

MILLER, Rowan. *Entity Framework Core*. Microsoft. 2016. Disponível em: EF Core - Visão Geral: https://docs.microsoft.com/pt-br/ef/core/ Acesso em 06 de Abril de 2019.

SATO, Priscila Mayumi. *ORM - Object Relational Mapping - Revista Easy .Net Magazine 28.* Devmedia. 2013. Disponível em: https://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158. Acesso em 03 de Abril de 2019.

SIMAS, Gabriel. *NHibernate Tutorial*. Devmedia. 2013. Disponível em https://www.devmedia.com.br/introducao-ao-nhibernate-framework-para-mapeamento-objeto-relacional/28671. Acesso em 20 de Abril de 2019

WILLIAN. *Acesso a dados com Dapper Micro ORM*. Using .Net. 2012. Disponível em https://usingdotnet.wordpress.com/2012/07/14/acesso-a-dados-com-dapper-micro-orm/ Acesso em 05 de Abril de 2019.