|  |  |
| --- | --- |
|  | *Ministério da Educação* **Universidade Tecnológica Federal do Paraná**  *Pró-Reitoria de Pesquisa e Pós-Graduação* |

**Relatório Final de Atividades**

**Comparação do tempo de trabalho entre modelo relacional e modelo não relacional**

**vinculado ao projeto**

**<Título do Projeto>**

**Jean Carlos da Silva**

**Bolsista UTFPR**

**Ciência da computação**

**Israel Rodrigues**

**Voluntário**

**Ciência da computação**

**Data de ingresso no programa: 08/2019**

**Prof(ª). Dr(ª). Kelyn Schenatto**

Área do Conhecimento: (de acordo com a tabela de áreas de conhecimento do CNPq)

*CAMPUS SANTA HELENA*, 2019

**JEAN CARLOS DA SILVA**

**ISRAEL RODRIGUES**

**KELYN SCHENATTO**

**GIUVANI CONTI**

**COMPARAÇÃO DE TEMPO DE TRABALHO DO MODELO RELACIONAL E NÃO RELACIONAL**

Relatório do Programa de Iniciação Científica da Universidade Tecnológica Federal do Paraná.

*Universidade Tecnológica Federal do Paraná, Campus-Santa Helena*, 2019

**SUMÁRIO**

|  |  |
| --- | --- |
| **INTRODUÇÃO** | **1** |
| **MATERIAIS E MÉTODOS** | **2** |
| **RESULTADOS E DISCUSSÕES** | **X** |
| **CONCLUSÕES**  **REFERÊNCIAS** | **X** |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**1. INTRODUÇÃO**

Com o constante crescimento da informação no mundo, o aumento das utilizações dos bancos de dados vem tornando-se a cada dia mais importante, tanto relacional quanto o modelo não-relacional, que por este estudo vem fazer um levantamento de informações de quais as vantagens que cada modelo oferece e assim comparar ambos os modelos nas questões de inserção, atualização de registros e a remoção. Sendo levantado duvidas pelos profissionais da área tecnológica de qual modelo seria mais eficiente para as aplicações a serem desenvolvidas.

O modelo não-relacional contem boas experiências comparado ao modelo relacional em diversas situações nos quais outros trabalhos já desenvolvidos resultaram em seu desempenho aprimorado em relação ao modelo relacional. Contudo o modelo não-relacional requer um pouco mais de atenção em relação ao modelo relacional, pois requer um grau de conhecimento mais avançado pelo fato de que trabalha de forma diferente, pois traz consigo um paradigma novo no quesito de armazenamento de dados. Como exemplo o armazenamento orientado a documentos.

# 2. METODOLOGIA

**Tecnologia.**

A tecnologia usada para o desenvolvimento do trabalho foi um notebook com o sistema operacional Windows 10 Home de 64 bits, processador Intel Core i5-7300HQ 2.50 GHz, 8 GB de RAM, 1T de HD.

O Software usado para a utilização dos sistemas gerenciadores de bancos de dados foi a versão do PostgreSQL 4, MySQL e o MongoDB.

**Programação.**

A programação foi desenvolvida utilizando a linguagem Java, onde todos as funções exercidas pelo programa para o modelo relacional, foram feitas com o auxílio do framework JPA (Java Persistence API) usado para persistir objetos dentro do banco de dados. Sendo assim não apenas um framework de mapeamento objeto-relacional, mas também tem uma gama de funcionalidades que facilita qualquer projeto.

Foram criadas duas classes para simular as atividades de inserção, atualização e a parte de deletar os dados dentro do banco de dados do modelo relacional, já para o modelo não relacional foi usado de forma direta através do java sem a criação de classe até pelo fato de que o NoSQL trabalha de forma diferente sendo orientado a documentos, colunas, que diferente do relacional que trabalha com junções entre tabelas que foram ou são criadas dentro do banco de dados.

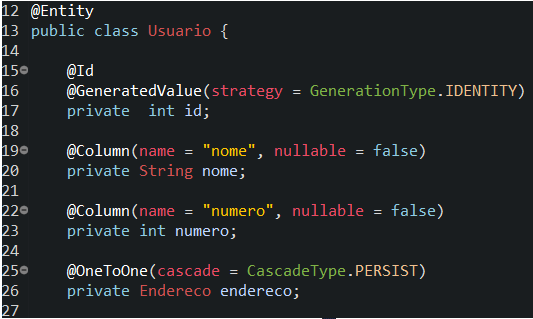


Figura 1. Demonstração da classe usuário.

Na figura 1 vemos a classe usuário com seus respectivos atributos e seus objetos para realizar o mapeamento através do JPA. Essas variáveis são criadas dentro da linguagem java e com o JPA são inseridas de forma automática dentro do banco MySQL quanto do PostgreSQL para assim não ter a necessidade de ir dentro do banco de dados e efetuar a criação da tabela manualmente.

A variável Id usada para identificar o usuário do qual pertence o nome que será informado pelo usuário, o número gerado aleatoriamente para cada usuário dentro de uma range de 0 a 1 bilhão e a variável endereço sendo do tipo Endereço da qual desta forma irá instanciar a classe endereço utilizando o Id do mesmo para ter o referenciamento. Na figura a seguir poderá ser observado a classe endereço.

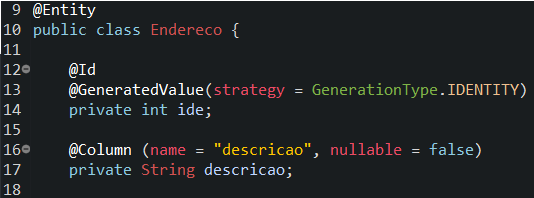


Figura 2. Demonstração da classe endereço.

A classe endereço tem como suas variáveis um ide que será gerado de forma também aleatória e a variável descrição onde será informado pelo usuário a localidade onde reside.

A anotação *@Entity* é usada para informar que uma classe é uma entidade que será relacionada a uma tabela no banco de dados através do mapeamento objeto relacional. Dentro de varias anotações utilizadas no projeto o *@Id* é utilizado para informar ao JPA que a variável que recebe esta anotação será uma chave primaria da respectiva tabela em que a classe está relacionada.

Logo após a anotação *@Id* deverá ter *@GeneratedValue* do qual utiliza-se para realizar a geração do valor único da entidade que será persistido pelo próprio provedor. Sem a utilização desta anotação implicará que o valor será gerado e gerenciado pelo próprio software.

**Conexão com banco de dados.**

Para realizar a conexão aos bancos de dados do modelo-relacional foi utilizado o arquivo chamado persistence.xml no qual foram feitas as configurações de acesso aos bancos MySQL e PostgreSQL.



Figura 3. Arquivo persistence xml

A elaboração da configuração do arquivo *persistence.xml* inicia-se atribuindo dentro do arquivo as classes que foram criadas com o efeito de criar as tabelas dentro do bando de dados. Após a esta etapa, as propriedades da conexão são configuradas com o driver do banco que no caso deste trabalho foi usado o MySQLe PostgreSQL.

Em seguida foi inserido o endereço do banco, sendo o endereço do localhost ou IP dependendo da aplicação e o nome da base onde foram criadas as tabelas. Usuário e senha são informados para que haja uma conexão com o banco de dados.

Após a conexão inicia-se a parte do hibernate sendo que uma das etapas mais importante encontra-se na última linha, aonde têm a opção *create* que tem como função gerar o banco e as tabelas. Entre esse comando também existe outros no quais fazem suas respectivas funções. No trabalho foi usado o comando comum *create*(apenas gera o banco) , *create-drop* (gera o banco e logo em seguida o apaga), *update*( utilizado para a atualização do banco) e *none* que não muda e não interfere no banco de dados sem ser pela aplicação, mas usado no projeto para apagar os dados inseridos dentro do banco de dados.

No NoSQL a conexão com o banco de dados do MongoDB foi realizada de forma direta, sem a utilização do *persistence.xml*. Pois trata-se de ser um método mais simples e eficaz para a conexão.

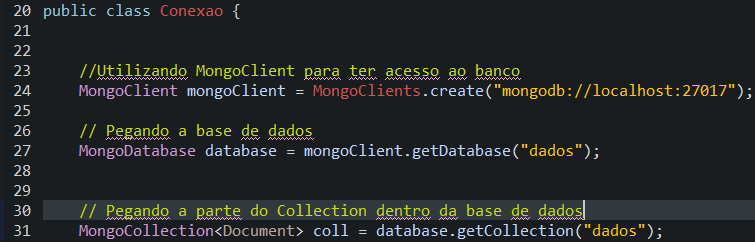


Figura 4**.** Classe conexão com o MongoDB

A classe conexão foi desenvolvida para realizar dentro dela a conexão direta com o MongoDB. A primeira parte da linha no qual começa com MongoClient que faz parte da biblioteca interna do mongo onde está relacionado ao objeto mongoClient do qual é atribuído o endereço de acesso ao servidor do banco que neste caso é em local host e a respectiva porta de acesso. Logo após no database é atribuído a base de dados que está dentro do mongo para ser adicionado todas as informações dentro dessa base de dados. Ao fim o MongoCollection que tem como um array na forma de documento para ser salvo dentro da base de dados do mongo. Collection é responsável pelas coleções pertencentes a base de dados e é uma função interna do Mongo que é instanciado e relacionado ao objeto coll.

**Função inserir (modelo relacional).**

Para o desenvolvimento do projeto foram dividas três funções como inserir, atualizar e deletar os dados para todos os bancos trabalhados.

Na função inserir iniciamos declarando as variáveis *i* e *x* que serão utilizadas no processo da construção da função. A variável *i* funcionará de forma que um id interno seja adicionado através da função **FOR** para obter o controle e garantir que o Id seja fornecido em forma crescente e organizada e que o Id não seja de forma aleatória.

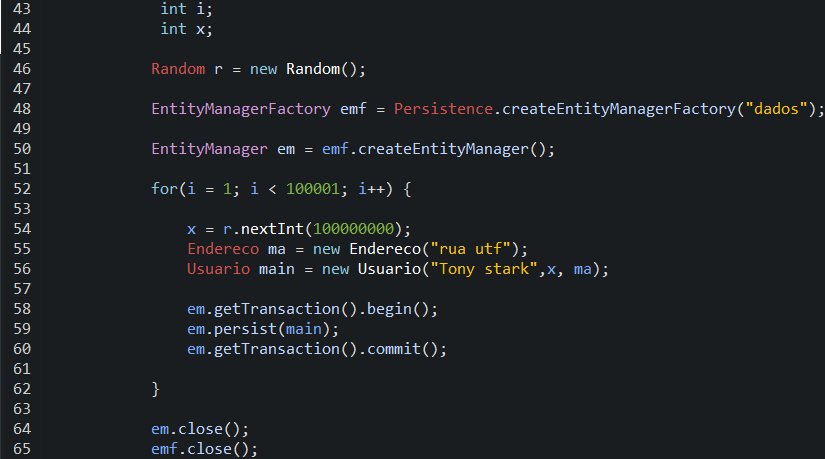


Figura 5. Função inserir.

A variável *x* assumirá o valor da variável *r* que será gerado de forma aleatória através de uma função randômica**,** que produz números aleatórios de um certo range estipulado pelo programa de zero a um bilhão. Dentro dos parâmetros da função para ocorrer a inserção nota-se de que o valor da variável *i* inicia-se com o número 1 e que a segunda parte do valor de *i* seria menor que o número de inserções.

As classes endereço e usuário são instanciadas e colocados seus respectivos parâmetros como na classe endereço a descrição do nome da rua e na classe usuário com o nome do indivíduo, o valor aleatório atribuído pela variável *x* e dentro como parâmetro da classe usuário foi utilizado uma variável endereço que tem como objetivo referenciar a classe endereço como parâmetro no usuário. Em seguida temos as funções do JPA sendo utilizadas para iniciar a persistência dentro do banco de dados do modelo relacional e após a função **FOR** ser totalmente executada é usada a função close para terminar a conexão com o banco de dados.

**Função atualizar (modelo relacional).**

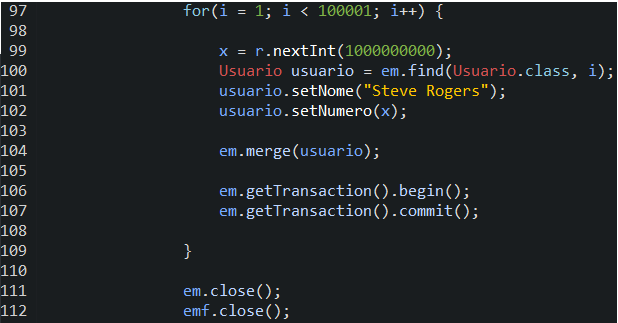


Figura 6. Função atualizar.

Nesta parte foi utilizado a função **FOR** novamente para realizar a passagem em todos os usuários cadastrados no banco de dados e utilizando o comando **find** através da classe usuário identificar o id de cada e realizar a atualização do nome e alterar o número aleatório gerado anteriormente na função inserir.

O comando merge do JPA irá criar um novo documento que no caso será o usuário. Dessa forma o novo documento era mudar o nome e o número do usuário cadastrado com o respectivo id que foi encontrado pelo método **find**. Mas continuará a manter as outras informações persistidas no documento que já foi preenchido. Ao grosso modo ele atualizará mesclando os arquivos mantando a informação, mas atualizando com informações nova.

**Função deletar (modelo relacional).**

# RESULTADOS E DISCUSSÕES

As figuras, tabelas, etc., devidamente referenciadas no texto, podem ser colocadas da maneira mais conveniente para o autor em uma ou duas colunas, desde que o texto permaneça em apenas uma coluna. Antes e após os elementos não textuais e suas respectivas legendas, deve-se deixar uma linha de espaçamento. Os autores não devem se esquecer da colocação de legendas nas figuras, tabelas e outros elementos gráficos. As figuras devem ser numeradas sequencialmente com algarismos arábicos conforme o exemplo da figura 1.



Figura 1. Centralizada na coluna e com legenda abaixo da figura.

Todas as equações deverão ser tabuladas a 1 cm da margem esquerda e numeradas sequencialmente, com os números entre parênteses, conforme o exemplo abaixo:

 (1)

As equações devem ser referenciadas no texto da seguinte forma: "Substituindo a equação (1) na equação (3), obtém-se ..."

Todas as tabelas deverão ser numeradas sequencialmente com algarismos arábicos, conforme o exemplo abaixo:

Tabela 1. Legenda acima da tabela, centralizada.

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| X | 1 | 2 | 3 |
| Y | 4 | 5 | 6 |
| W | 7 | 8 | 9 |
| Z | 8 | 7 | 6 |

Ressalta-se que as legendas das figuras devem ser colocadas abaixo, enquanto que as legendas das tabelas devem ser colocadas acima das mesmas.

Pede-se, ainda, que todos os autores revisem cuidadosamente a versão final do trabalho para evitar erros de digitação e formatação.

# CONCLUSÕES

Os bolsistas e demais alunos participantes dos programas de iniciação científica e tecnológica da UTFPR, deverão enviar os relatórios pelo meio recomendado até a data requisitada, para cumprir os requisitos do programa e ter direito a certificado.

# REFERÊNCIAS

[1] VAN EKENSTEIN, G.O.R.A.; DEURING, H.; TEN BRINKE, G.; ELLIS, T.S. Blends of Caprolactam/Caprolactone Copolymers and Chlorinated Polymers. *Polymer*, v. 38, p.3025-3034, 1997.

[2] COLEMAN, M.M.; GRAF J.F.; PAINTER, P.C. Specific Interactions and the Miscibility of Polymer Blends. Technomic, Lancaster, 1991.

[3] NRIAGU, J.O. Historical Perspectives. In: NRIAGU, J.O. e NIEBOER, E.(Eds.). Chromium in the Natural and Human Environments, ADVANCES IN ENVIRONMENTAL SCIENCE AND TECHNOLOGY. New York : John Wiley, v.20, 1988, p. 1-19.

[4] WIZENTIER, S.E., BOSCHI, A.O., VIEIRA, J.M. Fabricação de Membranas

Cerâmicas para Microfiltração. ANAIS DO 10o CONGRESSO BRASILEIRO DE

ENGENHARIA E CIÊNCIA DOS MATERIAIS, Caxambu, 2010, p 34-38.

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Nome Orientador e Assinatura | Nome Aluno e Assinatura |