Universidade Federal de Uberlândia
Curso de Bacharelado em Sistemas de Informação

Estudo Comparativo de Formas de Armazenamento de Arquivos Binários: BLOB em Sistemas de Gerenciamento de Bancos de Dados Relacionais ou Sistemas de Arquivos

Luan Borges Nunes

Uberlândia-MG, Brasil

Maio 2020

Luan Borges Nunes

Estudo Comparativo de Formas de Armazenamento de Arquivos Binários: BLOB em Sistemas de Gerenciamento de Bancos de Dados Relacionais ou Sistemas de Arquivos

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Orientador: Bruno Augusto Nassif Travençolo

Uberlândia-MG, Brasil

Maio 2020

Luan Borges Nunes

Estudo Comparativo de Formas de Armazenamento de Arquivos Binários: BLOB em Sistemas de Gerenciamento de Bancos de Dados Relacionais ou Sistemas de Arquivos

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Bruno Augusto Nassif Travençolo
Orientador

Prof. Dr. Humberto Luiz Razente

Prof. Dr. Wendel Alexandre Xavier de Melo

Uberlândia-MG, Brasil
Maio 2020

Agradecimentos

Agradeço ao meu orientador Bruno, por todos os conselhos e pela paciência de ajudar nesse período. Agradeço a minha família pelo apoio dado e por entender que não podia estar juntos em todos os momentos, e também a todos os professores que de alguma forma me transmitiram conhecimento durante esse tempo de estudo.

Resumo

Desde início da computação houve a necessidade de armazenar dados pra consultas futuras sendo esta uma tarefa que pode ser feita de diversas formas. Neste trabalho, é apresentado um estudo com duas destas formas: (i) uso do sistema de arquivos do sistema operacional – *filesystem*; e (ii) armazenamento interno de dados binários em sistemas gerenciadores de banco de dados, por meio do tipo de dado BLOB. O objetivo é saber, dentro de um contexto de um sistema de informação, qual das duas formas é mais eficiente para armazenamento de arquivos de tamanho relativamente pequenos (100KB até 20 MB). O estudo é feito levando em consideração as manipulações mais comuns para esses arquivos, como a leitura e armazenamento. Os testes consistiram em contabilizar o tempo gasto em: incluir os arquivos nas duas formas de armazenamentos citadas no estudo, carregar esses arquivos do meio de armazenamento para a memória principal do computador e a exclusão dos arquivos do sistema. Com esses testes, e levando em consideração que uma aplicação inicial dos resultados obtidos será o uso em um sistema existente em que a operação mais comum é a visualização de documentos, o meio de melhor rendimento foi o armazenamento em BLOB em um Banco de Dados.

PALAVRAS CHAVE: desempenho, sistemas de arquivos, armazenamento, BLOB.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MB	Megabyte
KB	Kilobyte
BD	Banco de Dados
FS	Sistema de Arquivos, do inglês *Filesystem*
RPM	Rotação por minuto
PDF	Documento de formato portátil, do inglês *portable document format*
BLOB	Objeto grande binário, do inglês *binary large object*
SCAD	Sistema de Cadastro de Atividades Docente
SGBD	Sistema Gerenciador de Banco de Dados
SO	Sistema Operacional
ACID	Atomomicidade, Consistência, Isolamento e Durabilidade, do inglês *Atomicity, Consistency, Isolation, Durability*

Sumário

1 Introdução

Uma enorme quantidade de arquivos é gerada por diferentes softwares (arquivos de fotos, textos formatados, PDF, planilhas, apresentações de slides, entre outros). O gerenciamento do armazenamento desses arquivos é feito pelo Sistema Operacional (SO), por meio dos sistemas de arquivos (FS – do inglês *file system*). Por outro lado, os Sistemas de Informação (SI) fazem uso de Sistemas Gerenciadores de Banco de Dados (SGBD) para armazenamento de informações relacionadas a esses sistemas. No entanto, em diversas situações, os SIs também necessitam armazenar arquivos gerados externamente por outros softwares. Eles podem fazer essa tarefa de duas formas: utilizando SGBDs (relacionais ou não), ou diretamente no sistema de arquivos do SO. Na maioria das vezes o que é feito no software se baseia no que o profissional tem mais conhecimento, não traduzindo, às vezes, a opção mais adequada para o problema.

Tendo em vista a necessidade de um estudo mais detalhado sobre o assunto, este trabalho fará os testes necessários para demonstrar qual a maneira mais eficiente de armazenamento do sistema, usando para os testes o armazenamento em SGBD utilizando BLOB e gravação diretamente no SO. Para fins de padronização, será utilizado o termo armazenamento em FS para referir ao armazenamento feito em arquivos utilizando o sistema de arquivos do SO. Já o termo armazenamento em BD será usado para referir ao armazenando em um SGDB relacional, utilizando tipo de dado BLOB.

1.1 Motivação

Na Faculdade de Computação da Universidade Federal de Uberlândia existe um sistema chamado SCAD (Sistema de Cadastro de Atividades Docente) [Silva 2019]. Esse software responsável por armazenar todas as atividades desenvolvidas pelo docente para fins de progressão/promoção funcional, além de emitir relatórios com a produção de cada docente. O SCAD necessita de armazenamento de longo prazo para diversos documentos. O projeto foi desenvolvido para que o armazenamento seja feito no banco de dados, porém, foi levantada a possibilidade de fazer o armazenamento diretamente no SO. No entanto, não se sabia de fato qual era a melhor forma de realizar o armazenamento, sendo essa dúvida a motivação inicial para elaborar este trabalho. Embora essa foi a motivação inicial, os resultados obtidos poderão ser usados em outros sistemas que possuam contexto semelhante ao SCAD.

1.2 Objetivos

O objetivo deste trabalho é demonstrar qual a melhor maneira de gerenciar documentos armazenados em um Sistema de Informação em um determinado contexto. Serão analisadas aqui duas formas de armazenamento, uma utilizando um Sistema Gerenciador de Banco de Dados relacional e outra armazenando diretamente no Sistema Operacional.

1.3 Trabalhos Correlatos

O trabalho *To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?* [Russell Sears , Catharine van Ingen , Jim Gray (Abril,2006)], faz diversas análises de tempo de acesso em arquivos, simulando vários níveis de fragmentação e vários tamanhos de arquivos em recuperações diretamente do sistema de arquivos do servidor e do banco de dados do tipo de dados *BLOB*.

Os autores fazem uma análise de tempo de transferência de arquivo, porém, não deixam claro ao que corresponde esse tempo, se é o tempo gasto para cópia de um servidor para o computador local, ou o tempo que se leva para copiar o arquivo do servidor até o mesmo abrir na tela do usuário. Eles trazem como ponto principal a fragmentação de disco, simulando algumas fragmentações em discos de 40GB e 400GB.

No trabalho eles descrevem cinco operações: incluir, ler, atualizar, escrever e excluir. Os resultados obtidos mostraram que, caso os objetos fossem maiores que 1 MB, em média, o armazenamento no SO tem uma clara vantagem, e quando abaixo de 256 KB o armazenamento em BD é a melhor escolha. No entanto, tudo dependia de quanto o armazenamento estava fragmentado.

1.4 Organização do Trabalho

O trabalho foi dividido em cinco capítulos. O Capítulo 1 traz a introdução do trabalho , qual a principal motivação e a descrição de um trabalho relacionado. O Capítulo 2, descreve como foram feitos o preparativos para os testes, como foram feitos os testes em todo o estudo e o hardware que foi utilizado no experimento. O Capítulo 3 traz todos os resultados obtidos separados por tamanho de arquivo utilizado, descrevendo para alguns tamanhos o tempo médio e em outros o tempo absoluto gasto por arquivo. O Capítulo 4 conta com uma pequena discussão sobre a necessidade de armazenamento em softwares, e uma breve explicação sobre a escolha do meios de armazenamento do experimento. E no último capítulo é feita a conclusão de todo o experimento trazendo ao leitor qual o meio de armazenamento se mostrou melhor em relação a todo o experimento.

2 O experimento

O experimento foi feito em duas etapas. A primeira parte consistiu na criação do aplicativo que faria os testes de tempo. Após o software estar pronto, foram criados os arquivos de diversos tamanhos para testes, e os experimentos foram realizados utilizando o software desenvolvido e os arquivos criados.

2.1 Condução do experimento

Foi criado um aplicativo chamado Temporizador (parte do código fonte é apresentada no Apêndice) para comparar os tempos das diversas movimentações dos arquivos. Esse programa foi construído em C# utilizando o Visual Studio 2019, pois os testes tinham como um dos pontos principais manter o mesmo ambiente e linguagem utilizado pelo SCAD.

Na criação da base de documentos, foi construído um aplicativo chamado Criador, em Visual Basic do Excel. Esse programa permite a criação de arquivos no formato PDF em diversos tamanhos (50 arquivos de 500KB no formato PDF, 50 arquivos de 1MB no formato PDF, 20 arquivos do formato PDF de 5MB, 10 arquivos de 20MB e mais 10 arquivos de 100MB). Após a finalização da criação os arquivos foram conferidos via sistema operacional, para se verificar se o tamanho correspondia ao esperado.

No experimento, foi utilizado o SO Windows 10 Pro 64 bits e como SGBD foi utilizado o PostgreeSql 12, versão 12.2.2 64bits [PostgreSQL 12.3 Documentation 2020], o SGBD também foi escolhido para manter o mesmo utilizado pelo SCAD. Não foi utilizado nenhum tipo de melhoria tanto do SGBD quanto no SO para melhora de desempenho. O SGBD utilizado estava com sua configuração padrão, em que o PostgreeSql utiliza o nível de isolamento *Read Committed*, que garante que apenas dados com a gravação ou alteração confirmada possa ser retornada em consultas [Ramakrishnan e Gehrke 2002].

Todo o trabalho descrito a seguir foi feito pelo software Temporizador. Durante a criação do Temporizador, foram realizados testes de consulta no SGBD, neste testes a consulta precisou ser reescrita pois quando retornava todas as colunas da tabela onde o BLOB estava armazenada, esta consulta demorava mais em comparação de quando apenas a coluna de ID do BLOB era retornada. Foram feitas 3 movimentações em cada arquivo em cada tipo de armazenamento do experimento (FS e BD), totalizando 6 movimentações diferentes em cada um dos arquivos, são elas: *inclusão, exibição* e *exclusão*. E, após cada uma, foi gravado em um arquivo de texto estruturado as seguintes informações: o tempo gasto, o tamanho do arquivo, sua descrição, o tipo de movimentação e qual o meio de

Tabela 1 – Funções implementadas.

Função	Local
Estrutura de conexão com o BD	Apêndice A
Estrutura básica do Temporizador	Apêndice B
Funções de teste de tempo de inclusão dos arquivos	Apêndice C
Funções de teste de tempo de abertura dos arquivo	Apêndice D
Funções de teste de tempo para deletar os arquivos	Apêndice E

armazenamento que estava sendo utilizado.

Na *inclusão* de arquivos no BD foi contabilizado o tempo que foi gasto desde a indicação do arquivo, até sua total cópia para a tabela correspondente do banco de dados e confirmação da gravação no banco, no fonte foi utilizado o *commit* implícito. Após isso, as informações sobre a transação foram gravadas no arquivo de texto estruturado. Para a operação de *exibição* do arquivo que estava gravado no BD, foi contabilizado o tempo, após identificação do nome do arquivo, passando pela busca do arquivo no BD até sua cópia total para a memória principal do computador. E, por último, o teste de *exclusão* foi contabilizado o tempo, entre a escolha de qual arquivo seria excluído até o retorno positivo do BD da execução correta da operação de exclusão, no fonte foi utilizado o *commit* implícito.

Para as operações feitas no teste do armazenamento em FS, o processo de *inclusão* teve seu tempo contabilizado desde a escolha do arquivo, na origem, passado pela gravação do caminho de destino na tabela correspondente no BD até se findar a cópia do arquivo no local de destino final. Para o processo de *exibição* o tempo considerado foi após a escolha do arquivo, passando pelo comando *SELECT* no banco de dados, que retorna a localização do arquivo no computador, até sua total cópia, do local de armazenamento para a memória principal do servidor. Para a *exclusão*, o tempo considerado foi a partir do momento em que se escolheu o arquivo, com o tempo de execução do comando de exclusão no banco de dados, até a finalização da exclusão do arquivo no FS.

Após executar as fases acima, foram gerados diversos gráficos levando em consideração o tamanho do arquivo e operação realizada, que serão mostrados no capítulo de Resultados. A Tabela 1 mostra quais funções foram implementadas e em qual apêndice se encontram parte do código fonte.

2.2 Hardware Utilizado

No experimentos foi utilizado um notebook Dell, modelo Inspiron 15 3576, com as configurações de hardware descrita a seguir. Todo o experimento foi feito em máquina física, sendo possível utilizar todo o potencial disponível, e conectado diretamente a uma fonte de energia, não utilizando a bateria do equipamento.

Processador: Intel® Core™ I5-8250U CPU @ 1,60Ghz

Memória RAM: 8 GB DDR4, *Single Channel*

Sistema Operacional: Windows 10 Pro 64 Bits, utilizando sistema de arquivo NTFS.

Unidade de Armazenamento: Kingston SA400S37480G, SATA 3 até 6 Gbits/s, 480 GB, unidade de estado sólido

3 Resultados

Neste capítulo serão apresentados os resultados encontrados durante a execução do experimento, divididos por tamanho de arquivo, para cada um dos cinco tamanhos: 500KB, 1MB, 5MB, 20MB e 100MB.

3.1 Arquivos de 500KB

Nos arquivos de 500KB, a Figura 1 exibe o tempo médio gasto em um total de 50 arquivos. Foram feitos os três testes citados anteriormente e nas duas formas de armazenamento, FS e BD, e os tempos médios foram calculados. Nesse tamanho de arquivo, o BD se mostrou o meio de armazenamento 500% mais rápido em relação ao FS para *exibir* os arquivos. Para as outras operações o BD também se mostrou o mais eficiente, gerando um economia de 40% no tempo gasto pra *incluir*, e gastando menos da metade do tempo para *excluir* o arquivo se comparado ao FS.

Considerando que o sistema para o qual o estudo está sendo feito a operação que mais vai ocorrer será a de exibição, o meio de armazenamento melhor para arquivos de 500KB é o BD.

Figura 1 – Tempo gasto para as operações em arquivos de até 500KB.

3.2 Arquivos de 1MB

A Figura 2 exibe o tempo médio gasto em um total de 50 arquivos com tamanho de 1 MB. Para esses arquivos o meio de armazenamento que se mostrou mais eficiente foi o BD, que levou um pouco menos de 33% do tempo para *exibir* o arquivo se comparado

ao FS. Para *inserir* e *deletar* o meio de armazenamento que se demonstrou melhor desempenho foi o FS, porém como já dito anteriormente por se tratar de um software de avaliação de conteúdo a operação de *exibir* será a que terá maior impacto na produção.

Com esse resultado, baseado nos testes para este tamanho de arquivo, também será escolhido o BD como melhor forma de armazenamento.

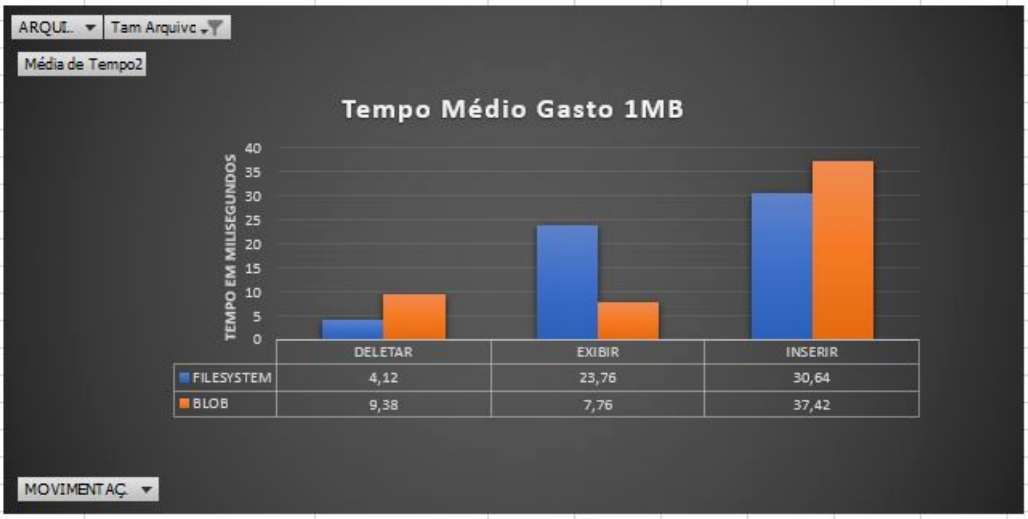

Figura 2 – Tempo médio gasto para as três operações em arquivos de até 1MB.

3.3 Arquivos de 5MB

No trabalho foram feitos testes em 20 arquivos de 5MB. A Figura 3 mostra o tempo médio gasto para *inserir*, *exibir* e *excluir* esses arquivos. O meio de armazenamento que mostrou melhor resultado para *exibir* os arquivos foi o BD, que obteve um desempenho de 30% superior ao armazenamento do FS, sendo esse o padrão que decidimos adotar. Para *inserir* os arquivos, o BD se mostrou melhor, e para *excluir* os arquivos o FS demonstrou desempenho superior ao BD. Portanto para arquivos de 5MB, o meio recomendado para armazenamento é o BD.

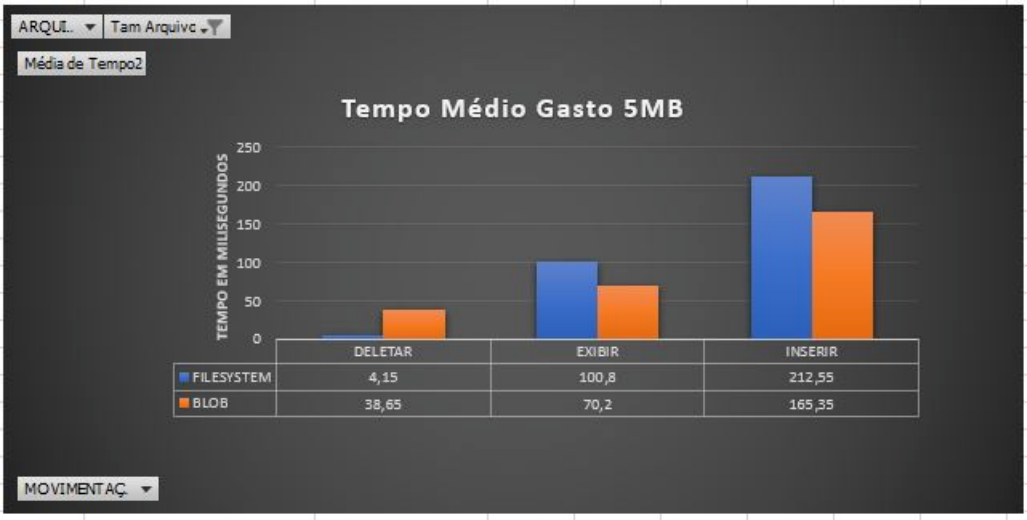

Figura 3 – Tempo médio gasto para executar as três operações em arquivos de 5MB.

3.4 Arquivos de 20MB

Para o experimento com arquivos de 20MB foram criados 10 arquivos. Por se tratar de arquivos de tamanho superior aos que serão usados no SCAD, foi utilizado um conjunto para testes menor, e repetidos os testes de inclusão, exibição e exclusão. Na Figura 4 estão apresentados os tempos médios. Neste tamanho de arquivo, o meio de armazenamento FS se mostrou 18% mais eficiente se comparado, ao tempo gasto para exibição se comparado ao armazenamento BD. Em todas as outras operações, o FS se mostrou mais eficiente, contudo o experimento está mais preocupado com o tempo gasto na exibição, dito isso, o meio de armazenamento recomendado para trabalhar com arquivos de 20MB é o FS.

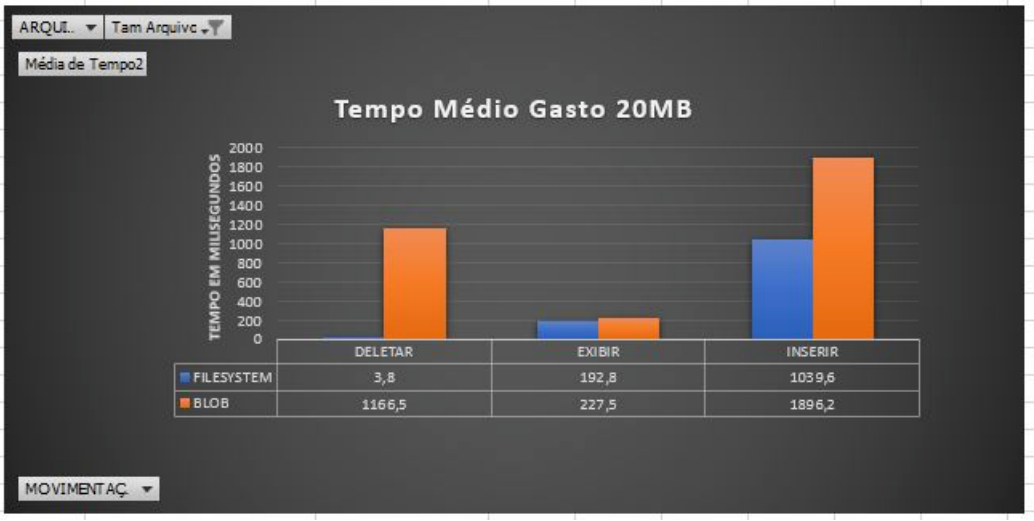

Figura 4 – Tempo médio gasto para executar as três operações em arquivos de 20MB.

3.5 Arquivos de 100MB

Para o experimento com arquivos de 100 MB, foram criados 10 arquivos. Por se tratar de arquivos de tamanho superior aos que serão usados no SCAD foi utilizada um conjunto para testes menor, e repetidos os testes de de *inserção*, *exibição* e *exclusão*. Na Figura 5 estão apresentados os tempos médios. Neste tamanho de arquivo, o meio de armazenamento FS teve o melhor tempo se comparado ao BD para as operações de *inserção*, *exibição* e *exclusão*. Portanto, o meio de armazenamento escolhido neste tamanho de arquivo é o FS.

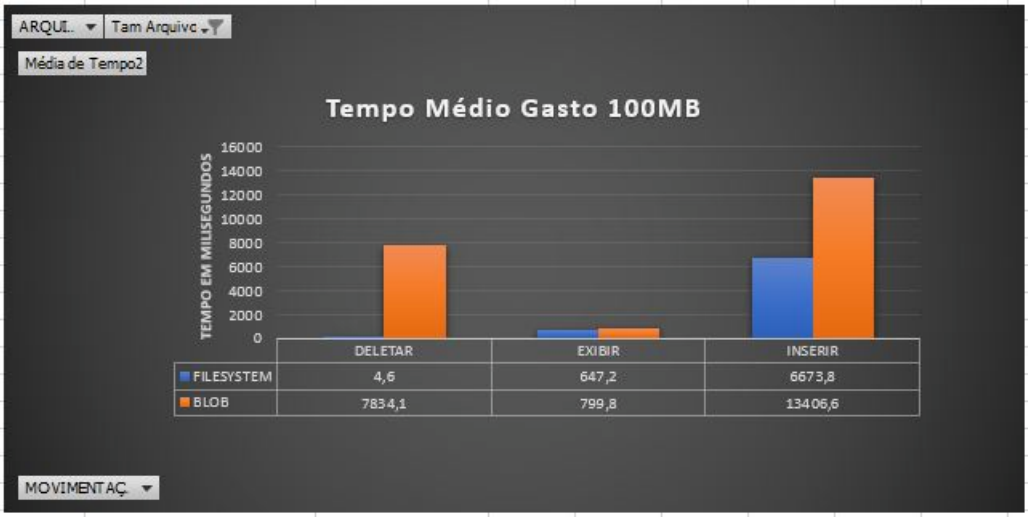

Figura 5 – Tempo médio gasto para executar as três operações em arquivos de 100MB.

3.6 Tempos Médios

A Figura 6 representa o tempo médio gasto pela operação de *inserir* para todos os arquivos utilizados no teste. Em arquivos de 5MB o armazenamento em BD e FS mantém seus tempos bem próximos, e após isso 20MB o armazenamento em FS se mostra a melhor opção.

A Figura 7 representa o tempo médio gasto pela operação de *exibir* para todos os arquivos utilizados no teste. O tipo de armazenamento FS apresentou melhor resultado em arquivos maiores de 20MB. Por outro lado o BD mostrou melhor desempenho em todos os outros tamanhos de arquivos

A Figura 8 representa o tempo médio gasto pela operação de *excluir* para todos os arquivos utilizados no teste. O tipo de armazenamento FS se mostrou mais eficiente em arquivos acima de 1MB, e o BD se mostrou melhor em arquivos de 500KB.

Um ponto a se observar e que o tempo gasto para *excluir* os arquivos no FS é praticamente constante pois, a exclusão independentemente do tamanho do arquivo é

feita da mesma forma, excluindo a referência inicial do endereço do arquivo no disco rígido [Tanenbaum 2016], assim não se alterando o tempo gasto para executar tal operação.

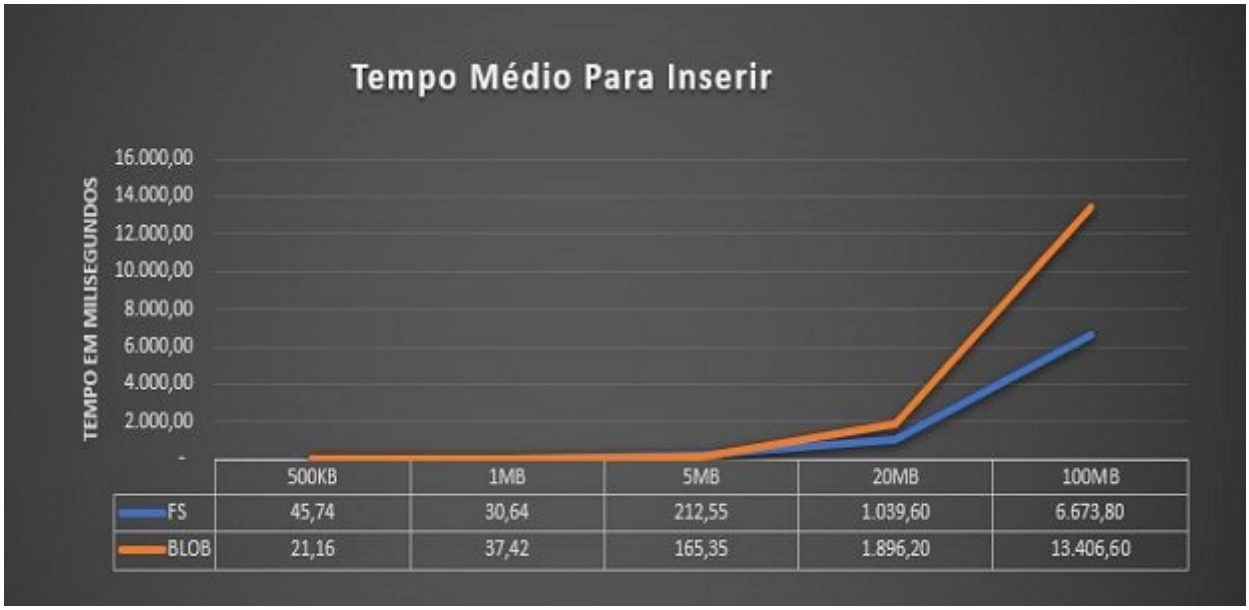

Figura 6 – Tempo médio gasto para inserir em todos os tamanhos de arquivos.

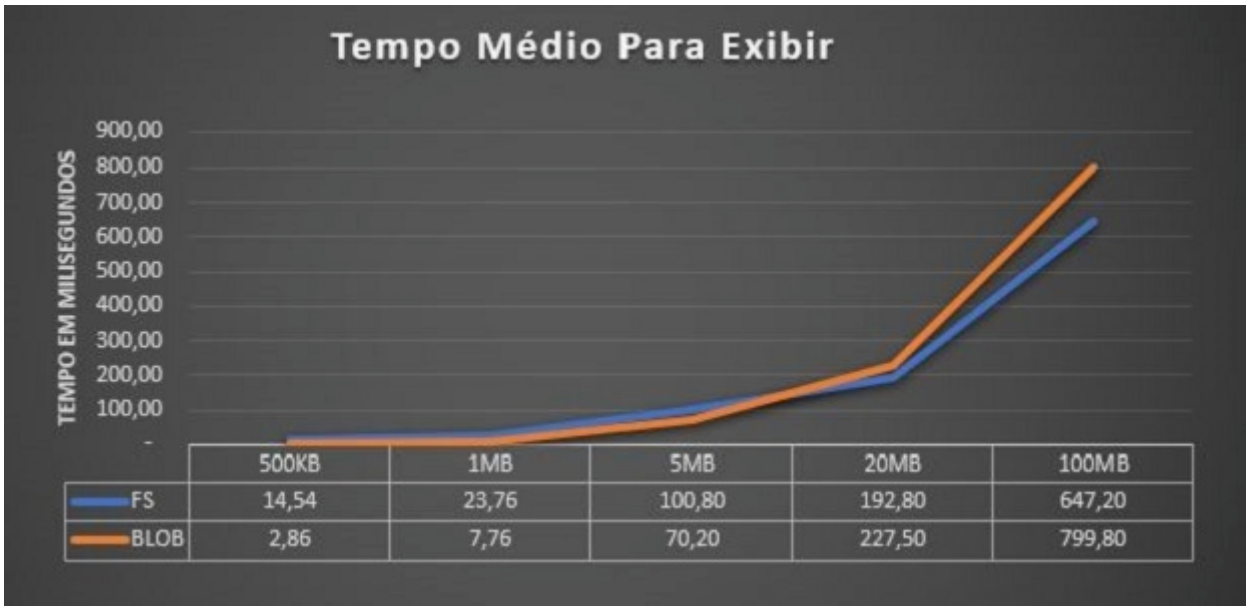

Figura 7 – Tempo médio gasto para exibir em todos os tamanhos de arquivos.

Figura 8 – Tempo médio gasto para excluir em todos os tamanhos de arquivos.

4 Discussão

A maioria dos sistemas que estão em funcionamento atualmente tem a necessidade de ter um meio de armazenamento de dados de longo prazo. Esses dados podem ser armazenados de maneiras estruturadas ou não.

Um dos modos de armazenamento que foi usado para testes foi o armazenamento diretamente no FS do sistema operacional. No estudo desenvolvido foi usado o banco de dados para armazenar o caminho dos arquivos do FS, para que a busca por determinado arquivo fosse feita de maneira mais eficiente, com isso o nível de conhecimento do programador ou equipe deve ser maior, pois além de salvar o arquivo no FS, deve-se também se conhecer previamente o banco de dados (estrutura, tipos de arquivos, linguagem, comunicação com a linguagem de programação). Algumas vulnerabilidades durante o estudo foram percebidas, como caso a pasta que contenha os arquivo tivesse seu conteúdo movido, toda a informação estruturada no banco seria desnecessária, pois estaria se referenciando a um arquivo que já não existe, onde a consistência desses arquivos não poderia ser garantida ; um acesso não autorizado que apagasse um arquivo de maneira injustificada, fazendo o sistema perder essa informação. E foi notada que a única segurança no armazenamento é a do próprio sistema operacional e que qualquer segurança extra deveria ser implementada diretamente no código fonte. Por exemplo, o uso de criptografia para a inclusão do arquivo e o processo de descriptografia para se conseguir visualizar o arquivo, deveria ser implementado diretamente no aplicativo criado.

O outro método de armazenamento que foi utilizada para os testes foi o armazenamento em BD do tipo de dado BLOB. Essa forma de armazenamento traz alguns benefícios, dentre eles estão todas as vantagens de um banco de dados (linguagem estruturada de busca, apenas usuários autorizados a uso no banco, maior facilidade na busca de arquivos). Do ponto de vista de programação do aplicativo que recebe os arquivos o banco de dados pode ser um pouco mais trabalhoso, pois o programador precisa conhecer ou ter alguém em sua equipe que conheça sua estrutura, os tipos de dados que ele reconhece, e como ele se comunica com a linguagem de programação usada em seu software.O BD foi escolhido para o experimento por possuir as propriedades ACID [Ramakrishnan e Gehrke 2002], que garante que todo documento inserido no SCAD possa ser exibido a qualquer tempo.

5 Conclusão

Ao se tirar conclusões sobre qual será o melhor meio de armazenamento de um software, foram levados em consideração dois fatores: segurança e velocidade.

O meio de armazenamento considerado mais seguro aqui descrito foi o BD. Além de tudo já citado anteriormente, ele já possui várias implementações de segurança prontas, sem precisar que o programador da aplicação se preocupe demais neste aspecto. O BD também se destaca por todas suas transações terem as propriedades ACID. Essas propriedades garantem que todo dado armazenado possa gerar uma informação segura quando necessário. Elas são essenciais pela natureza do aplicativo base dos testes, e portanto o tempo gasto na execução é o preço a se pagar pelas características benéficas adicionadas ao projeto. Outro ponto de destaque do BD, ele é utilizado por diversas pessoas, o que resulta em variadas formas de uso diferentes, que pode ser considerado como forma de testes do software.

Do ponto de vista de velocidade, e levando em consideração o principal forma uso do SCAD [Silva 2019], o BD foi considerado como a melhor opção de armazenamento, pois na maioria dos testes se mostrou mais veloz em relação ao armazenamento no FS. O principal teste utilizado para indicar o uso do BD, foi a função *exibir*, pois considerando que no geral, cada arquivo do SCAD, será incluído uma única vez, e raramente será excluído. Logo, a função *exibir* é a que terá maior impacto na velocidade do SCAD [Silva 2019].

Após todo o estudo, o BD foi considerada a melhor maneira para se fazer o armazenamento para Sistemas de Informação em que operações de consultas são bem mais frequentes que operações de inserção e remoção, pois traz mais segurança e velocidade.

Para trabalhos futuros, fica como sugestão a inclusão de outros SGBDs e outras implementações de sistemas de arquivos de diferentes sistemas operacionais como base de comparação. Outra sugestão, é a utilização de tabelas no SGBD com tamanho pré-alocados para armazenamento dos documentos. Utilização de melhorias nos arquivos de log do SGBD, para que a gravação do arquivos BLOB seja feita diretamente no destino, causando uma possível melhoria de desempenho. Esses trabalhos futuros podem incluir também informações de tempo mínimo, máximo e mediana nas operações executadas, a fim de revelar possíveis discrepâncias de tempo e também demonstrar qual o nível de utilização do dispositivo de armazenamento no momento dos testes. Outra sugestão de trabalho futuro é realizar testes simulando a concorrência de acessos. Outra possibilidade é simular diferentes níveis de fragmentação de disco.

Referências

POSTGRESQL 12.3 Documentation. 2020. Disponível em: <https://www.postgresql.org/files/documentation/pdf/12/postgresql-12-A4.pdf/>. Acesso em: 09 de abril 2020. Citado na página 11.

RAMAKRISHNAN, R.; GEHRKE, J. *Database Management Systems*. 3. ed. USA: McGraw-Hill, Inc., 2002. ISBN 0072465638. Citado 2 vezes nas páginas 11 e 20.

Russell Sears , Catharine van Ingen , Jim Gray. *To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?* (Abril,2006). Disponível em: <https://arxiv.org/abs/cs/0701168>. Acesso em: 16 de junho 2020. Citado na página 10.

SILVA, W. S. *Desenvolvimento do Sistema de Cadastro de Atividades Docente*. 2019. Disponível em : https://repositorio.ufu.br/handle/123456789/28962. Acesso em: 10/06/2020. Citado 2 vezes nas páginas 9 e 21.

TANENBAUM, H. B. A. S. *Sistemas Operacionais Modernos*. 4ª. ed. [S.l.]: Pearson, 2016. ISBN 978-85-4301-818-8. Citado na página 18.

Appendices

A Estrutura de conexão com o banco de dados

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data;
7  using Npgsql;
8  namespace PostGRES
9  {
10  class DAL
11  {
12   static string serverName = "127.0.0.1";  //ip do servidor de BD
13   static string port = "2703";   //porta configurada no servidor
        de BD para reconhecer a conexão externa
14   static string userName = "postgres"; //nome do usuário com
        permissão de escrita/leitura/deleção do BD
15   static string password = "*******"; //senha do usuário com
        permissão de escrita/leitura/deleção do BD
16   static string databaseName = "postgres";  //nome do BD
17   NpgsqlConnection pgsqlConnection = null;  //Cria a conexão do
        BD
18   string connString = null; //variavel da string de conexão do BD
19   public string getStringCon()
20   {  return connString;    }
21
22   public DAL()   //instancia um novo objeto criando a string de
        conexão ao BD
23   {   connString = String.Format("Server={0};Port={1};User Id
        ={2};Password={3};Database={4};Timeout=1024;CommandTimeout
        =1024;", serverName, port, userName, password, databaseName);
24   }
25   public DataTable GetTodosRegistros()  //retorna todos os
        registros da tabela criada para o trabalho, tabela dos FS.
26   {
27    DataTable dt = new DataTable(); //cria o objeto para receber a
         tabela do BD
```

```
28    using (pgsqlConnection = new NpgsqlConnection(connString))
          //conecta ao BD usando a string de conexão
29    {
30     pgsqlConnection.Open(); // abre a conexão com o PgSQL e
          define a instrução SQL
31     string cmdSeleciona = "Select Id from documento2 ORDER BY 1";
32     using (NpgsqlDataAdapter Adpt = new NpgsqlDataAdapter(
          cmdSeleciona, pgsqlConnection)) //grava toda a consulta no
          objeto dt criado acima
33     {
34      Adpt.Fill(dt);
35     }
36    }
37    pgsqlConnection.Close(); //fecha a conexão com o BD.
38    return dt;
39   }
40   //RETORNA A TABELA COM OS BLOBs
41   public DataTable GetTodosByte()
42   {
43    DataTable dt = new DataTable();  //cria o objeto para receber
          a tabela do BD
44    using (pgsqlConnection = new NpgsqlConnection(connString))  //
          conecta ao BD usando a string de conexão
45    {
46     pgsqlConnection.Open();  // abre a conexão com o PgSQL e
          define a instrução SQL
47     string cmdSeleciona = "Select id , tipo from documento ORDER
          BY 1";
48     using (NpgsqlDataAdapter Adpt = new NpgsqlDataAdapter(
          cmdSeleciona, pgsqlConnection)) //grava toda a consulta no
          objeto dt criado acima
49     {
50      Adpt.Fill(dt);
51     }
52    }
53    pgsqlConnection.Close();
54    return dt;
55   }
56  }
57 }
```

B Estrutura do software Temporizador

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Diagnostics;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Odbc;
using Npgsql;
using System.Data.Common;
using System.Data.SqlClient;

namespace PostGRES
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }



        int codigo;
        DAL acesso = new DAL();

        private void Form1_Load(object sender, EventArgs e)
        {   }


  // FUNÇÕES DE INCLUIR, VISUALIZAR, DELETAR ARQUIVOS

    } }
```

C Funções de teste de tempo de inclusão dos arquivos

```
1  private void BtnInsFile_Click(object sender, EventArgs e)
2    {
3     int[] iQtdDoc = new int[5]; //quantidade de tamanhos que
         desejo fazer o teste no filesystem
4      iQtdDoc[0] = 50; //quantidade de arquivos que serão inseridos
          no primeiro tamanho
5      iQtdDoc[1] = 50; //quantidade de arquivos que serão inseridos
          no segundo tamanho
6      iQtdDoc[2] = 20; //quantidade de arquivos que serão inseridos
          no terceiro tamanho
7      iQtdDoc[3] = 10; //quantidade de arquivos que serão inseridos
          no segundo tamanho
8      iQtdDoc[4] = 10; //quantidade de arquivos que serão inseridos
          no terceiro tamanho
9
10    StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
         append: true); //caminho do arquivo de tempos
11    string sCaminhoOrigem = "C:\\TCC1\\TCC2 - ORIGEM\\"; //origem
         dos arquivos que serão inseridos no FILESYSTEM
12    string sCaminhoDestino = "C:\\TCC1\\TCC2\\";//Pasta de destino
          dos arquivos filesystem
13    String[] sDescDoc = new string[5]; // nome dos arquivos que
         serão inseridos
14    sDescDoc[0] = "Relatório_até_500KB_";
15    sDescDoc[1] = "Relatório_até_1mb_";
16    sDescDoc[2] = "Relatório_até_5mb_";
17    sDescDoc[3] = "Relatório_entre_5_e_20_mb_";
18    sDescDoc[4] = "Relatório_entre_20_e_100 _mb_";
19    salvar.WriteLine("TP_DOC;MOVIMENTAÇÃO;ARQUIVO;TEMPO"); //
         arquivo de tempos estruturado
20    for (int i = 0; i < 5; i++)
21    {
22     for (int j = 1; j <= iQtdDoc[i]; j++)
23     {
24      Stopwatch sw2 = new Stopwatch(); // cria o objeto para
         contabilizar o tempo
```

```
25      sw2.Start(); // inicializao cronômetro de tempo de inclusão
           no filesystem
26      NpgsqlConnection conn = new NpgsqlConnection(acesso.
           getStringCon()); // cria a conexão com o BD
27      conn.Open(); //Abre a conexão com o BD
28      NpgsqlCommand Insertcmd = new NpgsqlCommand("INSERT INTO
           documento2 VALUES(COALESCE((select max(id) from documento2
           ),0)+1,'"
29          + sDescDoc[i] + j.ToString() + ".pdf" + "')", conn); //
              inclue no BD o caminho do arquivo, que será inserido no
               FILESYSTEM
30      Insertcmd.ExecuteReader(); //Executa no bd o comando
           anterior
31      FileStream reader = File.Open(sCaminhoOrigem + sDescDoc[i] +
            j.ToString() + ".pdf", FileMode.Open); // lê o arquivo de
            origem
32      FileStream reader1 = new FileStream(sCaminhoDestino +
           sDescDoc[i] + j.ToString() + ".pdf", FileMode.Create,
           FileAccess.ReadWrite); // cria o arquivo de destino
33      reader.CopyTo(reader1); // copia o arquivo da origem para o
           arquivo de destino
34      reader.Close(); // fecha o arquivo de origem
35      reader1.Close(); // fecha o arquivo de destino
36      sw2.Stop(); // para o crônometro
37      conn.Close(); // fecha a conexão com o BD
38      tbMostrar.Text = tbMostrar.Text + "INSERINDO FILESYSTEM : "
           + sCaminhoDestino +
39          sDescDoc[i] + j.ToString() + ".pdf \r\n"; // exibir a
              inclusão no filesystem no textbox da aplicação
40      long tempo2 = sw2.ElapsedMilliseconds; // grava o cronômetro
            em millisegundos
41      salvar.WriteLine("FILESYSTEM;INSERIR;" + sDescDoc[i] + j.
           ToString() + ".pdf ; " + tempo2.ToString() + ""); //
           Contabiliza o tempo gasto, para estudo posterior
42     }
43    }
44    salvar.Close();
45   }
46
47 private void BtnInsBloob_Click(object sender, EventArgs e)
48   {
49    NpgsqlConnection conn = new NpgsqlConnection(acesso.
```

```
        getStringCon());
50      int[] iQtdDoc = new int[5]; //quantidade de tamanhos que
          desejo fazer o teste no filesystem
51      iQtdDoc[0] = 50; //quantidade de arquivos que serão inseridos
          no primeiro tamanho
52      iQtdDoc[1] = 50; //quantidade de arquivos que serão inseridos
          no segundo tamanho
53      iQtdDoc[2] = 20; //quantidade de arquivos que serão inseridos
          no terceiro tamanho
54      iQtdDoc[3] = 10; //quantidade de arquivos que serão inseridos
          no segundo tamanho
55      iQtdDoc[4] = 10;
56      StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
          append: true);
57      string sCaminho = "C:\\TCC1\\TCC2 - ORIGEM\\";
58      String[] sDescDoc = new string[5]; // nome dos arquivos que
          serão inseridos
59      sDescDoc[0] = "Relatório_até_500KB_";
60      sDescDoc[1] = "Relatório_até_1mb_";
61      sDescDoc[2] = "Relatório_até_5mb_";
62      sDescDoc[3] = "Relatório_entre_5_e_20_mb_";
63      sDescDoc[4] = "Relatório_entre_20_e_100 _mb_";
64      for (int i = 0; i < 5; i++)
65      {
66       for (int j = 1; j <= iQtdDoc[i]; j++)
67       {
68        Stopwatch sw2 = new Stopwatch();
69        sw2.Start(); // inicia o relógio
70        FileStream pgFileStream = new FileStream(sCaminho + sDescDoc
            [i] + j.ToString() + ".pdf", FileMode.Open, FileAccess.
            Read); //cria um stream do arquivo origem
71        BinaryReader pgReader = new BinaryReader(new BufferedStream(
            pgFileStream));//cria um objeto para leitura do arquivo de
             origem
72        byte[] PdfBytea = pgReader.ReadBytes(Convert.ToInt32(
            pgFileStream.Length)); // cria o objeto que será copiado
            para o bd, com o tamanho da origem, e faz a copia os dados
             da origem
73        string sQL = "INSERT INTO documento VALUES( COALESCE((select
             max(id) from documento),0)+1, @PDF, '" + sDescDoc[i] + j.
            ToString() + ".pdf" + "')";//cria o comando de insert
74        using (var command = new NpgsqlCommand(sQL, conn)) //cria a
```

```
        variavel de gravação do BD, copiando o arquivo
75      {
76       NpgsqlParameter param = command.CreateParameter();
77       param.ParameterName = "@PDF"; // cria o parametro que será
            usado para salvar o arquivo no BD
78       param.NpgsqlDbType = NpgsqlTypes.NpgsqlDbType.Bytea;
79       param.Value = PdfBytea;
80       command.Parameters.Add(param);
81       conn.Open();
82       command.ExecuteNonQuery(); // INSERI O ARQUIVO PDF, NO
            BANCO DE DADOS PARA UM TIPO BLOB
83      }
84      sw2.Stop();
85      conn.Close();
86      long tempo2 = sw2.ElapsedMilliseconds;
87      tbMostrar.Text = tbMostrar.Text + "INSERINDO BLOOB : " +
           sCaminho + sDescDoc[i] + j.ToString() + ".pdf \r\n";
88      salvar.WriteLine("BLOOB;INSERIR;" + sDescDoc[i] + j.ToString
           () + ".pdf ; " + tempo2.ToString() + "");
89     }
90    }
91    salvar.Close();
92   }
```

D Funções de teste de tempo de abertura dos arquivo

```
1  private void BtnExbFile_Click(object sender, EventArgs e)
2    {
3     StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
         append: true);
4     string sCaminhoDestino = "C:\\TCC1\\TCC2\\"; //Caminho base de
          onde estão os arquivos
5     DataTable table = acesso.GetTodosRegistros(); // lê toda a
         tabela que armazena o local dos arquivos do filesystem
6     System.Diagnostics.ProcessStartInfo startInfo = new System.
         Diagnostics.ProcessStartInfo();
7     tbMostrar.Text = "";
8     salvar.WriteLine("tp_doc;movimentação;arquivo;tempo_gasto");/
9     foreach (DataRow row in table.Rows) // Lê todas a linhas da
         tabela onde estão os dados dos FS
10    {
11     try
12     {
13      Stopwatch sw2 = new Stopwatch();
14      sw2.Start();
15      NpgsqlConnection conn = new NpgsqlConnection(acesso.
           getStringCon());
16      conn.Open();
17      NpgsqlCommand arquivo = new NpgsqlCommand("SELECT
           link_arquivo FROM documento2  WHERE id=" + row["id"].
           ToString(), conn);
18      NpgsqlDataReader dr = arquivo.ExecuteReader();
19      dr.Read();
20      if (dr.HasRows)
21      {
22         FileStream reader = File.Open(sCaminhoDestino+dr[0].
              ToString(), FileMode.Open);
23         MemoryStream reader1 = new MemoryStream();
24         reader.CopyTo(reader1);
25         reader.Close();
26         reader1.Close();
27       sw2.Stop();
```

```
28        long tempo2 = sw2.ElapsedMilliseconds;
29        salvar.WriteLine( "FILESYSTEM;EXIBIR;" + dr[0].ToString() +
              ";" + tempo2.ToString());
30          tbMostrar.Text = tbMostrar.Text + "EXIBINDO FILESYSTEM :
                " + dr[0].ToString() + "\r\n";
31          reader1.Dispose();
32          reader.Dispose();
33       }
34        conn.Close();
35      }
36      catch(Exception erro)
37      {
38       tbMostrar.Text = tbMostrar.Text + " Tamanho: 0 bytes \r\n";
39      }
40    }
41    salvar.Close();
42  }
43
44  private void BtnExbBloob_Click(object sender, EventArgs e)
45  {
46    StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
        append: true);
47    salvar.WriteLine("tp_doc;movimentação;arquivo;tempo_gasto");
48    string sPASTA = "C:\\TCC1\\TCC2_BLOOB\\";  // lê o byteb
49    DataTable table1 = acesso.GetTodosByte(); // lê todas as
        linhas da tabela onde estão os blob e retorna o id.
50    System.Diagnostics.ProcessStartInfo startInfo1 = new System.
        Diagnostics.ProcessStartInfo();
51    foreach (DataRow row in table1.Rows) // Loop over the rows.
52    {
53      NpgsqlConnection conn = new NpgsqlConnection(acesso.
          getStringCon());
54      conn.Open();
55      Stopwatch sw = new Stopwatch();
56      sw.Start();
57      NpgsqlCommand arquivo = new NpgsqlCommand("SELECT arquivo
          FROM documento  WHERE id=" + row["id"].ToString(), conn);
58      NpgsqlDataReader dr = arquivo.ExecuteReader();
59      dr.Read();
60      if (dr.HasRows)
61      {
62       Byte[] result = new Byte[Convert.ToInt32((dr.GetBytes(0, 0,
```

```
        null, 0, Int32.MaxValue)))];
63      dr.GetBytes(0, 0, result, 0, result.Length);
64      sw.Stop();
65      long tempo = sw.ElapsedMilliseconds;
66      FileStream fs = new FileStream(sPASTA + row["tipo"].ToString
           (), FileMode.Create, FileAccess.ReadWrite);
67      for (int i = 0; i < result.Length; i++)
68       fs.WriteByte(result[i]);
69      fs.Close();
70      System.IO.File.Delete(sPASTA + row["tipo"].ToString());
71      fs.Dispose();
72      salvar.WriteLine("BLOOB;EXIBIR;" + row["tipo"].ToString() +
           " ; " + tempo.ToString() + "");
73      tbMostrar.Text = tbMostrar.Text + "EXIBIR BLOOB :  " + row["
           tipo"].ToString() + " \r\n";
74     }
75     conn.Close();
76    }
77    salvar.Close();
78   }
```

E Funções de teste de tempo para deletar os arquivos

```
1  private void BtnDelFile_Click(object sender, EventArgs e)
2   {
3    StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
        append: true);
4    string sCaminhoDestino = "C:\\TCC1\\TCC2\\"; //lê  os arquivo
        do filesystem
5    DataTable table = acesso.GetTodosRegistros(); // Get the data
        table.
6    System.Diagnostics.ProcessStartInfo startInfo = new System.
        Diagnostics.ProcessStartInfo();
7    salvar.WriteLine("tp_doc;movimentação;arquivo;tempo_gasto");
8    //traz os dados do filesystem
9    foreach (DataRow row in table.Rows) // Loop over the rows.
10   {
11     NpgsqlConnection conn1 = new NpgsqlConnection(acesso.
          getStringCon());
12     conn1.Open();
13     Stopwatch sw2 = new Stopwatch();
14     sw2.Start();
15     NpgsqlCommand arquivo = new NpgsqlCommand("SELECT
          link_arquivo FROM documento2  WHERE id=" + row["ID"].
          ToString(), conn1);
16     NpgsqlDataReader dr = arquivo.ExecuteReader();
17     dr.Read();
18    if (dr.HasRows)
19    {
20     try
21     {
22      NpgsqlConnection conn = new NpgsqlConnection(acesso.
           getStringCon());
23      conn.Open();
24      System.IO.File.Delete(sCaminhoDestino+dr[0].ToString());
25      NpgsqlCommand Insertcmd = new NpgsqlCommand("DELETE FROM
           documento2 WHERE ID =  " + @row["ID"].ToString(), conn);
26      Insertcmd.ExecuteReader();
27      sw2.Stop();
```

```
28        conn.Close();
29        long tempo2 = sw2.ElapsedMilliseconds;
30        salvar.WriteLine("FILESYSTEM;DELETAR;" + dr[0].ToString() +
              ";" + tempo2.ToString());
31        tbMostrar.Text = tbMostrar.Text + "DELETANDO FILESYSTEM :
             " + dr[0].ToString() + "\r\n";
32       }
33       catch (Exception erro)
34       {
35        tbMostrar.Text = "ID NÃO ENCONTRADO....";
36       }
37       conn1.Close();
38      }
39     }
40     salvar.Close();
41    }
42
43
44    private void BtnDelBloob_Click(object sender, EventArgs e)
45    {
46     StreamWriter salvar = new StreamWriter("C:\\TCC1\\DADOS.csv",
          append: true);
47     salvar.WriteLine("tp_doc;movimentação;arquivo;tempo_gasto");
           // lê o byteb
48     DataTable table1 = acesso.GetTodosByte(); // Get the data
          table.
49     System.Diagnostics.ProcessStartInfo startInfo1 = new System.
          Diagnostics.ProcessStartInfo();
50     foreach (DataRow row in table1.Rows) // Loop over the rows.
51     {
52      try
53      {
54       NpgsqlConnection conn1 = new NpgsqlConnection(acesso.
            getStringCon());
55       conn1.Open();
56       NpgsqlCommand arquivo = new NpgsqlCommand("SELECT tipo FROM
            documento  WHERE id=" + row["id"].ToString(), conn1);
57       NpgsqlDataReader dr = arquivo.ExecuteReader();
58       dr.Read();
59       Stopwatch sw2 = new Stopwatch();
60       sw2.Start();
61       NpgsqlConnection conn = new NpgsqlConnection(acesso.
```

```
        getStringCon());
62      conn.Open();
63      NpgsqlCommand Insertcmd = new NpgsqlCommand("DELETE FROM
           documento WHERE ID =  " + @row["id"].ToString(), conn);
64      Insertcmd.ExecuteReader();
65      sw2.Stop();
66      conn.Close();
67      long tempo = sw2.ElapsedMilliseconds;
68      salvar.WriteLine("BLOOB;DELETAR;" + dr[0].ToString() + " ; "
            + tempo.ToString() + "");
69      tbMostrar.Text = tbMostrar.Text + "DELETAR BLOOB :  " + dr
           [0].ToString() + " \r\n";
70      conn1.Close();
71     }
72     catch (Exception erro)
73     {
74      tbMostrar.Text = "ID NAO ENCONTRADO" ;
75     }
76        }
77   salvar.Close();  }
```