Utilização da Classe de Banco de Dados NOSQL como Solução para Manipulação de Diversas Estruturas de Dados

Ricardo Cardoso de Almeida[1], Parcilene Fernandes de Brito[1]

[1]Curso de Sistemas de Informação – Centro Universitário Luterano de Palmas (CEULP/ULBRA)
Caixa Postal 160 – 77.054-970 – Palmas – TO - Brasil

ricardoalmeida.br@gmail.com,pfb@ulbra-to.br

Resumo. *Ferramentas que tem como função atender a usuários que necessitam tratar dados advindos de diversas fontes muitas vezes apresentam como gargalo o fato de ter que pré-definir as possíveis estruturas de dados que a ferramenta suporta, principalmente quando utiliza modelo de banco de dados relacional. Com propósito de estender o escopo deste tipo de ferramentas e possibilitar que o usuário vincule diversas estruturas de dados, o presente trabalho tem por objetivo apresentar a classe de banco de dados NOSQL como solução para viabilizar flexibilidade para uma ferramenta de geração de relatórios, tornando possível a utilização de diversas estruturas de dados, com a finalidade de oferecer alta escalabilidade e disponibilidade, que são características importantes considerando o fato de que não se tem informação sobre o volume de dados que o usuário da ferramenta importará.*

Palavras-Chave: NOSQL, Escalabilidade, MONGODB

1. Introdução

Geralmente quando se trata de relatórios gerados a partir de bases de dados operacionais, o gestor necessita de mão-de-obra especializada que entenda suas necessidades e gere relatórios utilizando recursos como linguagem de programação ou consultas diretas a base de dados (através de *scripts*, por exemplo). O desenvolvimento e entendimento da necessidade do gestor por parte do especialista demandam tempo e custo. Muitas vezes, o gestor necessita de uma análise simples e funcional, que seria rapidamente sanada se ele tivesse uma forma de manipular os dados da base sem a necessidade de um conhecimento técnico específico.

Uma Ferramenta para atender este cenário ainda apresentaria limitações, caso utilizasse modelo de banco de dados relacional, pois este tipo de modelo exige que a estrutura dos dados seja pré-definida, criando assim, uma pequena margem de possibilidades. Para estender o espoco da ferramenta e permitir que diversas estruturas de dados possam ser manipuladas, pode-se utilizar a classe de banco de dados denominada NOSQL, que atende "requisitos de alta escalabilidade necessários para gerenciar grandes quantidades de dados, bem como para garantir a alta disponibilidade dos mesmos" (LÓSCIO et al., 2011, p. 3).

Assim, o presente trabalho tem por objetivo apresentar a proposta de aplicação da classe de banco de dados NOSQL como solução de armazenamento de possíveis dados que possam ser utilizados em ferramentas com esta demanda. Desta forma, tais ferramentas poderão oferecer suporte a estruturas de dados variadas e que não tenha

necessariamente uma estrutura pré-definida, além de solucionar problemas como escalabilidade da ferramenta (por poder trabalhar com grandes cargas de dados sem que afete o desempenho da ferramenta).

2. Not Only SQL (NOSQL)

Existem diversos SGBDs (Sistemas de Gerenciamento de Banco de Dados) disponíveis no mercado. Porém, tanto na literatura quanto em soluções comerciais, o modelo relacional de banco de dados é o mais difundido. Isto porque, segundo Lóscio et al. (2011, p. 3), "os conceitos básicos do modelo relacional são: relação (tabela), atributo (coluna) e tupla (linha)", os quais são fáceis de compreender.

Os bancos de dados relacionais surgiram no início dos anos 70, "os quais se firmaram como solução comercial para armazenamento e gerenciamento de dados convencionais, ou seja, dados que possuem uma estrutura fixa, bem definida e com tipos de dados simples" (LÓSCIO et al., 2011, p. 3). Porém, com advento da Web 2.0 as necessidades em armazenamento de dados foram modificadas, pois são grandes volumes de dados produzidos continuamente, principalmente através de serviços de busca como *Google Search*, *Bing*, entre outros e redes sociais como *Twitter*, *Facebook*, *MySpace* etc. Em geral, estes dados não possuem estrutura fixa, como as páginas *web* (em que as estruturas descritas nos documentos HTML não indicam muito da semântica do conteúdo do documento) e materiais multimídia (como textos, vídeos, áudios etc.). Para suprir essas necessidades, surgiu uma nova categoria de banco de dados denominada NOSQL (abreviação para "*Not Only* SQL"), que veio como "uma solução para a questão da escalabilidade no armazenamento e processamento de grandes volumes de dados na *Web* 2.0" (DIANA & GEROSA, 2010, p. 2).

Tabela 6 – **Análise Comparativa Modelo Relacional x NOSQL (BRITO, 2010, p. 5)**

	Relacional	**NOSQL**
Escalonamento	Possível, mas complexo. Devido à natureza estruturada do modelo, a adição de forma dinâmica e transparente de novos nós no *grid* não é realizada de modo natural.	Uma das principais vantagens desse modelo. Por não possuir nenhum tipo de esquema pré-definido, o modelo possui maior flexibilidade o que favorece a inclusão transparente de outros elementos.
Consistência	Ponto mais forte do modelo relacional. As regras de consistência presentes propiciam um maior grau de rigor quanto à consistência das informações.	Realizada de modo eventual no modelo: só garante que, se nenhuma atualização for realizada sobre o item de dados, todos os acessos a esse item devolverão o último valor atualizado.
Disponibilidade	Dada a dificuldade de se conseguir trabalhar de forma eficiente com a distribuição dos dados, esse modelo pode não suportar a demanda muito grande de informações do banco.	Outro fator fundamental do sucesso desse modelo. O alto grau de distribuição dos dados propicia que um maior número de solicitações aos dados seja atendida por parte do sistema e que o sistema fique menos tempo não disponível.

A Tabela 1 permite uma análise comparativa entre as classes de banco de dados, a partir do teorema CAP (*Consistency, Availability e Partition Tolerance)*, que em português, significa: consistência, disponibilidade e tolerância ao particionamento (na Tabela 1 representada pelo escalonamento). Essas três características são esperadas em um sistema computacional distribuído, porém segundo o teorema, no mundo real, um sistema computacional distribuído só pode garantir apenas duas dessas características simultaneamente (Leite, 2010). Essa abordagem pode ser observada em bases NOSQL,

pois, apesar das bases de dados NOSQL não oferecerem a propriedade de consistência como em bases relacionais (que garantem a integridade dos dados), apresenta como pontos fortes o escalonamento e a disponibilidade (conforme visto na Tabela 1). Pode-se concluir que a classe de bando de dados NOSQL não tem por objetivo substituir o modelo relacional, mas atender uma nova demanda do mercado, em situações em que seja necessário de forma flexível trabalhar com cargas de dados semiestruturadas ou não estruturadas e também em sistemas que apresentem grandes volumes de dados e necessitem de disponibilidade para seus usuários.

Segundo Diana & Gerosa (2010), são "os tipos mais comuns de bancos de dados NOSQL: bancos de dados orientados a documentos, armazéns de chave-valor, bancos de dados de famílias de colunas e bancos de dados de grafos". As seções a seguir detalham cada um desses tipos de banco de dados NOSQL.

2.1 Chave-valor

Este modelo tem sua composição de maneira simples, "trata-se de uma abordagem parecida com uma tabela *hash*" (TOTH, 2011, p. 3). Segundo Lóscio et al. (2010, p. 6), este "banco de dados é composto por um conjunto de chaves, as quais estão associadas um único valor, que pode ser uma *string* ou um binário".

Segundo Strauch (2012, p. 52, tradução nossa), o modelo chave-valor "favorece alta escalabilidade ao invés da consistência e, portanto, a maioria deles também omite recursos ricos para consultas e ferramentas analíticas (especialmente operações de *joins* e agregação são postas de lado)". Segundo Toth (2011), o modelo chave-valor na maioria dos casos apresenta uma interface composta principalmente pelos seguintes métodos:

- Put(chave, valor) – representa o método que insere um registro, garante maior velocidade a operação;
- Get(chave) – responsável por recuperar um registro, não oferece opções para buscas mais complexas.

2.2 Orientado a Colunas

Este modelo se contrapõe ao paradigma do modelo relacional de armazenamento em tuplas (instâncias dos atributos em formato de linha), pois no modelo baseado em família de colunas "os dados são indexados por uma tripla (linha, coluna e *timestamp*), onde linhas e colunas são identificadas por chaves e o *timestamp* permite diferenciar múltiplas versões de um mesmo dado" (LÓSCIO et al., 2011, p. 6).

Para melhor entendimento do modelo baseado em colunas, Silva (2012) utiliza um banco de dados denominado "Cassandra" e separa a estrutura da seguinte forma:

- Colunas: compostas por um nome que identifica a coluna, além de um valor e um *timestamp* (que são fornecidos pela aplicação cliente no momento da inserção).
- Família de colunas: é análogo a uma tabela do modelo relacional, ao contrário das colunas as famílias de colunas não são dinâmicas, sendo necessário declará-las anteriormente em um arquivo de configuração.
- Super Colunas: são compostas por outras colunas.
- Super Famílias de Colunas: são compostas somente por Super Colunas.

Segundo Abadi (2008), se o objetivo do trabalho é acessar dados sobre a granularidade de uma entidade (por exemplo, pesquisar um curso, inserir um novo curso, excluir um curso) então o modelo tradicional baseado em linhas é o mais indicado, considerando que todas as informações desejáveis são armazenadas em conjunto. Já em situações em que é necessário ler apenas alguns atributos de vários registros, então o modelo de colunas é preferível (por exemplo, uma consulta que verifica a faixa etária mais comum no curso de Sistemas de Informação).

Diana & Gerosa destacam que o custo de escrita do modelo baseado em família de colunas é bem maior que em um banco de dados relacional, em que "os bancos tradicionais são mais adequados a processamento de transações online (OLTP) enquanto os bancos de dados de famílias de colunas são mais interessantes para processamento analítico online (OLAP)" (DIANA & GEROSA, 2010, p. 6).

2.3 Orientado a Grafos

O modelo de grafo é muito difundido em outras áreas da computação e está relacionado a soluções matemáticas. A estrutura do modelo de grafos é composta por: "nós (são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós e relacionamentos" (LÓSCIO et al., 2011, p. 8). Toth (2012, p.4) ainda destaca que "pode-se armazenar qualquer tipo de dados dentro do conteúdo".

A seguir são listadas algumas das vantagens na utilização do modelo orientado a grafo:

- O fato de "não existir tanta replicação de dados como nos outros modelos, fato que acontece por se aproveitar do relacionamento entre os registros" (TOTH, 2012, p. 4). Em modelos como o orientado a documentos, por exemplo, cada documento armazena todas suas relações de forma atômica, desta forma, se houver dois documentos, um representando a pessoa "A" e outra representando a pessoa "B" e ambos conhecem a pessoa "C", tanto o documento "A" quanto "B" teriam que guardar uma instância da pessoa "C". No caso dos grafos, isso é resolvido apenas utilizando ligações com arestas que armazenam um valor que indica que a pessoa conhece "C";
- "A vantagem de utilização do modelo baseado em grafos fica bastante clara quando consultas complexas são exigidas pelo usuário. Comparado ao modelo relacional, que para estas situações pode ser muito custoso" (LÓSCIO et al., 2010, p. 8). Isso ocorre por existirem algoritmos heurísticos que podem ser aplicados para otimizar a busca por valores ao se percorrer um grafo, definindo o melhor caminho diretamente do nó inicial até o nó destino, sem que seja necessário verificar outros nós. No modelo relacional essas consultas exigiriam a construção de *joins*, que dependendo da complexidade podem afetar no desempenho de execução da consulta;
- "Esse modelo também dá suporte ao uso de restrições sobre os dados, como restrições de identidade e de integridade referencial, por exemplo" (Diana & Gerosa, 2010, p. 6). Ao contrário da maioria dos modelos NOSQL que são mais flexíveis;
- "Esse modelo pode tornar consultas rápidas, devido à possibilidade da utilização de propriedades de grafos, medidas de centralidade (*pagerank*) e algoritmos de menor caminho" (TOTH, 2012, p. 3).

Logo, para situações de contexto complexo em que se conhece a semântica entre os objetos armazenados, o modelo orientado a grafo pode apresentar-se como uma escolha vantajosa.

2.4 Orientado a Documentos

Esse modelo de banco de dados armazena coleções de documentos, em que um documento representa "um objeto com um identificador único e um conjunto de campos, que podem ser *strings*, listas ou documentos aninhados" (LÓSCIO, 2011, p. 7). Diana & Gerosa (2010, p. 5) destacam que as bases de dados orientadas a documentos "não possuem esquema, ou seja, os documentos armazenados não precisam possuir estrutura em comum", isto permite que novos campos sejam adicionados a um documento sem que isso cause algum problema na base de dados.

O fato dos documentos integrarem suas relações diretamente dentro do próprio documento é o fator que gera duplicidade dos dados, porém torna a consulta direta. Desta forma, por exemplo, tendo como contexto uma estrutura que mantenha a relação de Pessoa com Cargo e fosse criada uma consulta para recuperar o cargo da pessoa "A", no modelo orientado a documentos bastaria recuperar o documento da "A", que conteria o documento de seu cargo, já no modelo relacional seria necessário realizar um *join* entre a tabela Pessoa e Cargo, o que tem um custo de desempenho para execução, melhor percebido em consultas mais complexas.

Outra vantagem da duplicação de dados é que "facilita a distribuição do sistema, já que a quantidade de nós a serem consultados em uma busca envolvendo várias entidades relacionadas é menor caso elas estejam próximas" (DIANA & GEROSA, 2010, p.5). Porém, dependendo do contexto, segundo Diana & Gerosa (2010, p. 5), "pode criar problemas de consistência no banco de dados, causados por anomalias de atualização e deleção", isso ocorre, por exemplo, caso um documento duplicado seja atualizado em apenas um documento que o armazena.

Segundo Strauch (2012), as bases de dados *Apache CouchDB* e *MongoDB* são os principais representantes dessa classe de banco de dados baseada em documento. A seção a seguir aborda o banco de dados *MongoDB* que foi utilizado na proposta de solução apresentada no presente trabalho.

3. MongoDB

Segundo Matteussi (2010, p. 39), "*MongoDB* é um banco de dados orientado a documentos de alta performance, *open source* e de esquema livre, escrito em C++. Formado por uma mistura entre os repositórios escaláveis e a tradicional riqueza de funcionalidades dos bancos de dados relacionais". Os documentos contemplados pela base de dados *MongoDB* é persistido "por um formato chamado BSON que é muito semelhante ao JSON mas em uma representação binária, por razões de eficiência e por causa de tipos de dados adicionais em comparação com JSON" (STRAUCH, 2012, p. 77). O modelo JSON é muito difundido no desenvolvimento *Web*, pois é bastante utilizado em soluções Ajax em aplicações que dependem de comunicações assíncronas.

Camara & Garcia (2011) sugere que os seguintes recursos do MongoDB sejam considerados:

- Binários disponíveis para Linux, Sun Solaris, Apple MacOS e Microsoft Windows;

- Shell online, com possibilidade de testes sem a necessidade de instalação;
- Documentação abrangente e detalhada – diminuindo assim a curva de aprendizagem;
- *Drivers* oficiais para C, C++, C#, Haskell, Java, JavaScript, Perl, PHP, Python, Ryby e Scala, além de *drivers* para diversas outras linguagens suportados pela Comunidade – O que demonstra a estabilidade e procura do banco de dados MongoDB como solução de armazenamento;
- Suporte a expressões regulares em consultas – Desta forma permite a realização de consultas mais complexas;
- Escala horizontal com auto-sharding – Permite que *clusters* sejam contados de forma dinâmica através de recursos oferecidos pela API do banco de dados MongoDB;
- Opções de filtragem, agregação e classificação, tais como limit(), skip(), sort(), count(), distinct() e group() – Recursos comuns no modelo relacional de banco de dados;
- Armazenamento de grandes arquivos com uso de GridFS - É "um mecanismo para armazenar grandes objetos eficientemente, visto que um objeto no formato BSON é limitado a 4MB e dividido em vários documentos" (SANTOS, 2011, p. 28);
- Suporte a indexação com índices semelhantes aos do modelo relacional de banco de dados – "a utilização de BSON habilita a indexação de qualquer tipo de dados, proporcionando o melhoramento do desempenho relativo às consultas" (SANTOS, 2011, p. 27);
- Replicação Master/Slave – "A leitura torna-se mais rápida, porém a capacidade de escrita torna-se um gargalo nesta abordagem" (LÓSCIO, 2011, p. 4).

Segundo os desenvolvedores do banco de dados *MongoDB* o seu principal objetivo é fechar a lacuna entre o rápido e altamente escalável modelo chave-valor e os bancos de dados relacionais que são ricos em funcionalidades tradicionais (STRAUCH, 2012, p.76). Deve-se destacar que esse banco de dados não substitui a utilização dos bancos de dados relacionais, muitas das aplicações necessitam, por exemplo, de uma estrutura rígida e/ou alta consistência. Porém, quando é preciso aplicar uma solução que utilize banco de dados orientado a documentos, o *MongoDB* pode ser uma boa opção, pois além de apresentar uma documentação abrangente e detalhada, também oferece recursos avançados de consulta (como filtragem, agregação e classificação), o que torna a curva de aprendizagem menor para desenvolvedores que têm experiência na utilização do modelo relacional.

5. Utilização do NOSQL como Solução

O fato de uma ferramenta poder aceitar qualquer estrutura de dados descarta a utilização do modelo relacional de banco de dados, pois estes apresentam estrutura rígida. Uma possível solução demandaria a criação de um modelo abstrato, que poderia ter muito custo para aplicação. Além disso, aceitar qualquer estrutura de dados deixa a aplicação sem a certeza do volume de dados que o usuário irá importar, o que pode trazer problemas de escalabilidade, em que o aumento do volume de dados afeta diretamente

na performance da ferramenta. Para solucionar tanto a flexibilidade de aceitar qualquer estrutura de dados quanto para promover alta escalabilidade, a classe de banco de dados NOSQL foi utilizada como proposta de solução.

Dentre os principais tipos de banco de dados NOSQL, para a solução proposta neste trabalho, foi escolhido o banco de dados orientado a documentos, pois não apresenta uma estrutura pré-definida, podendo ser alterada a qualquer momento, além de oferecer alta escalabilidade e disponibilidade. O banco de dados orientado a documentos escolhido foi o *MongoDB*, que oferece vários dos recursos avançados de consulta (como filtragem, agregação e classificação), necessários para a construção dos filtros dinâmicos previstos para ferramenta. A Figura 1 apresenta como a solução NOSQL funcionará no protótipo.

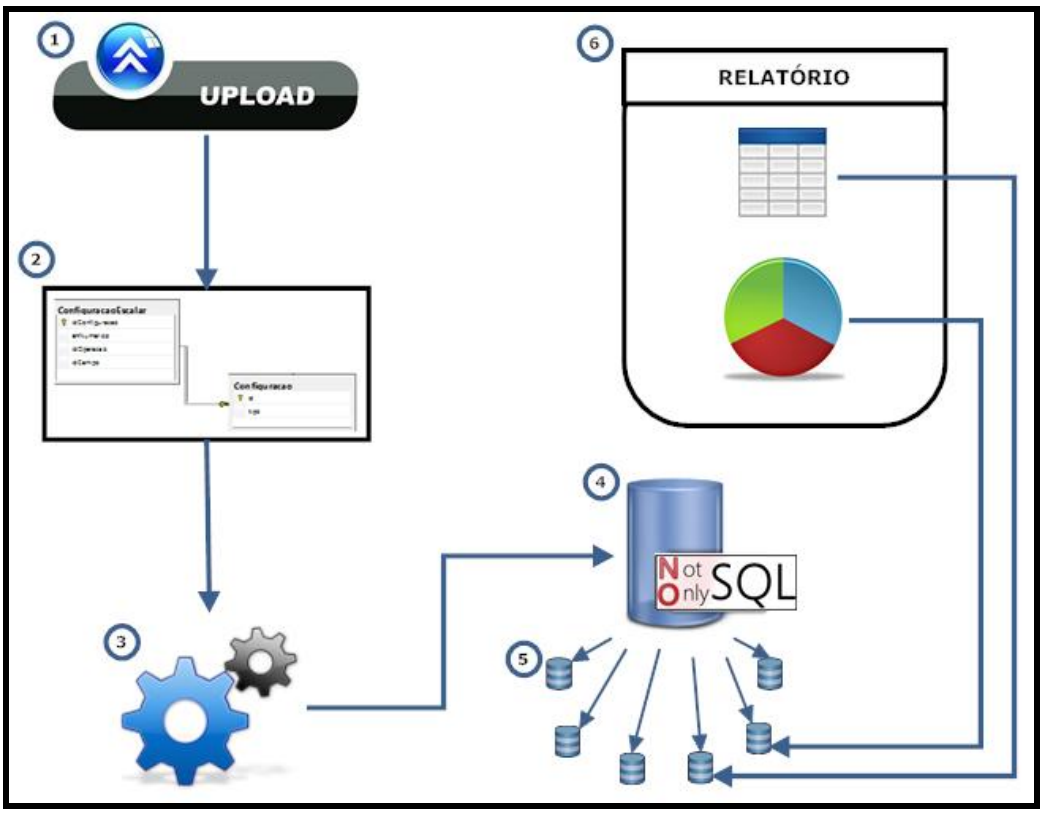

Figura 1 - Papel do NOSQL no protótipo

A partir do esquema da Figura 1, podem-se observar os seguintes passos na utilização da classe de banco de dados NOSQL:

- **Item 1** – representa o momento em que o usuário indica a base de dados que deseja importar;
- **Item 2** – a ferramenta se conecta a base de dados indicada pelo usuário;
- **Item 3** – através de um mecanismo escrito na linguagem PHP, a estrutura dos dados importados pelo usuário é convertida do modelo relacional para o modelo NOSQL. Por fim, após criar a estrutura de documento os dados são migrados para base NOSQL;

- **Item 4** – indica o banco de dados que guarda todas as fontes de dados importadas pelo usuário;
- **Item 5** – representam os *datasources* gerados a partir das importações. Para que a ferramenta diferencie cada importação criada pelo usuário, é inserido a cada documento o nome do *datasource* a que ele pertence, este dado é posto em cada consulta montada dinamicamente na ferramenta;
- **Item 6 –** é um exemplo de relatório final, que contem dois objetos, um gráfico e uma tabela, em que cada objeto foi gerado a partir de um datasource diferente. Por exemplo, o gráfico pode estar utilizando um *datasource* referente a dados de uma instituição de ensino de São Paulo, enquanto a tabela está sendo preenchida por dados advindos de um *datasource* com informações de uma instituição de ensino do Tocantins.

A seguir (Figura 2) é apresentada a relação de operadores de filtragem do modelo relacional com operadores baseados no *MongoDB*.

1

id	nome	descricao
1	Igual a	= $valor
2	Diferente de	<> $valor
3	Maior do que	> $valor
4	Maior ou igual a	>= $valor
5	Menor do que	< $valor
6	Menor ou igual a	<= $valor
7	Começa com	like('$valor%')
8	Não começa com	not like ('$valor%')
9	Termina com	like('%$valor')
10	Não termina com	not like('%$valor')
11	Contêm	like('%$valor%')
12	Não Contêm	not like('%$valo...

2

id	nome	descricao
1	Igual a	=
2	Diferente de	$ne
3	Maior do que	$gt
4	Maior ou igual a	$gte
5	Menor do que	$lt
6	Menor ou igual a	$lte
7	Começa com	/^@valor/i
8	Não começa com	$not,/^@valor/i
9	Termina com	/@valor^/i
10	Não termina com	$not,/@valor^/i
11	Contêm	/@valor/i
12	Não Contêm	$not,/@valor/i

Figura 2 – Equivalência dos Operadores entre a base relacional e o MongoDB

Pode-se observar, a partir da Figura 2, que todos os tipos de filtragem oferecidos pelo protótipo, tanto em sua versão anterior (Item 1) quanto na versão apresentada no presente trabalho (Item 2), tem as mesmas opções de pesquisa. Na coluna "descricao" do Item 1 é possível notar que as instruções condizem com o padrão SQL e, para cada uma dessas instruções, foi inserido seu equivalente no formato aceito pela API do *MongoDB*. Tornando possível assim importar dados de outras fontes de dados e executar operações de consulta.

6. Conclusões

O presente trabalho teve por objetivo apresentar a proposta de aplicação da classe de banco de dados NOSQL como solução de armazenamento para ferramentas que manipulem diversas fontes de dados.

A classe de banco de dados NOSQL se apresentou como uma boa solução para resolução de possíveis problemas de escalabilidade (por não se ter a informação do volume de dados que o usuário pode importar) e flexibilidade no que tange um dos objetivos específicos do trabalho, que é permitir que o usuário importe qualquer tipo de estrutura de dados. Dentre os principais tipos de banco de dados NOSQL, para o protótipo proposto neste trabalho, foi escolhido o banco de dados orientado a documentos, pois não apresenta uma estrutura pré-definida, podendo ser alterada a

qualquer momento, além de oferecer alta escalabilidade e disponibilidade. Sabe-se que o banco de dados orientado a documentos não garante a consistência dos dados, assim, esta solução é melhor empregada em casos em que a ferramenta não permite que os dados sejam alterados depois de importados (processo comum quando aplicado *datawarehouse* por exemplo).

O banco de dados orientado a documentos escolhido foi o *MongoDB* que oferece vários dos recursos avançados de consulta (como filtragem, agregação e classificação), que foram necessários para a construção dos filtros dinâmicos previstos para ferramenta. Ainda o *MongoDB* já oferece soluções de *sharing* de forma nativa possibilitando que se crie *clusters* do banco de dados, garantindo alta escalabilidade com a inserção de novos nós a *grid*.

7. Referências Bibliográficas

ABADI, Daniel J. **Query Execution in Column-Oriented Database Systems.** 2008. 148 p. Thesis (Doctor of Philosophy in Computer Science and Engineering) – Massachusetts Institute of Technology, Massachusetts, USA.

BRITO, Ricardo W. Bancos de Dados **NoSQL x SGBDs Relacionais:** Análise Comparativa. In: InfoBrasil, 3, 2010, Fortaleza. ***Anais...*** Fortaleza, 2010.

CAMARA, Evandro Augusto Muchinski; GARCIA, Renato Joukoski. **Avaliação de Sistemas de Reconhecimento de Fala Para Indexação Automática De Vídeos.** 2011. 39 p. Monografia (Bacharel em Ciências da Computação) – Universidade Federal do Paraná, Curitiba, Paraná.

DIANA, Mauricio De; GEROSA, Marco Aurélio. **NOSQL na Web 2.0:** Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0. In: Workshop de Teses e Dissertações em Banco de Dados, 9, 2010, Belo Horizonte. ***Ainais...*** Belo Horizonte, 2010.

LEITE, Gleidson Sobreira. **Análise Comparativa do Teorema CAP Entre Bancos de Dados NoSQL e Bancos de Dados Relacionais.** 2010. 49 p. Monografia (Bacharel em Ciências da Computação) – Faculdade Farias Brito, Fortaleza.

LÓSCIO, Bernadette Farias; OLIVEIRA, Hélio Rodrigues de; PONTES, Jonas César de Sousa. NoSQL no desenvolvimento de aplicações Web colaborativas. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, 8, 2011, Paraty. ***Anais...*** Paraty: SBC, 2007.

MATTEUSSI, Kassiano José. Protótipo **De Interface Web Com Php Para Gerenciamento de Banco de Dados CouchDB.** 2010. 81 p. Monografia (Bacharel em Ciências da Computação) – Universidade Comunitária Da Região De Chapecó, Chapecó, Santa Catarina.

SANTOS, Nuno Miguel Queirós Arantes dos. **Bases de Dados alternativas para Websites.** 2011. 130 p. Dissertação (Mestrado em Engenharia Informática e Computação) – Faculdade De Engenharia Da Universidade Do Porto, Porto, Portugal.

SILVA, Tiago Pasqualini da. **Cassandra –** Uma sistema de armazenamento NoSQL altamente escalável. Disponível em:

http://www2.sor.ufscar.br/verdi/topicosCloud/Cassandra.pdf. Acesso em: 22 de maio de 2012.

STRAUCH, Christof. **NoSQL Databases.** Disponível em: http://www.christof-strauch.de/nosqldbs.pdf. Acesso em: 25 de abril de 2012.

TOTH, Renato Molina. **Abordagem NoSQL –** uma real alternativa. Disponível em: http://www2.sor.ufscar.br/~verdi/topicosCloud/nosql_artigo.pdf. Acesso em: 16 de maio de 2012.