Estudo Comparativo de Banco de Dados Chave-Valor com Armazenamento em Memória

Dinei A. Rockenbach[1], **Nadine Anderle**[1], **Dalvan Griebler**[1,2], **Samuel Souza**[1]

[1] Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC)
Faculdade Três de Maio (SETREM) – Três de Maio – RS – Brasil

[2]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS/PPGCC)
Porto Alegre – RS – Brasil

{dineiar,nadianderle}@gmail.com,dalvan.griebler@acad.pucrs.br,
samuel@samuelsouza.com

Abstract. *Key-value databases emerge to address relational databases' limitations and with the increasing capacity of RAM memory it is possible to offer greater performance and versatility in data storage and processing. The objective is to perform a comparative study of key-value databases with memory storage Redis, Memcached, Voldemort, Aerospike, Hazelcast and Riak KV. Thus, the work contributed to an analysis of different databases and with results that qualitatively demonstrated the characteristics and pointed out the main advantages.*

Resumo. *Bancos de Dados (BD) chave-valor surgem para suprir limitações de BDs relacionais e com o aumento da capacidade das memórias RAM é possível oferecer maior desempenho e versatilidade no armazenamento e processamento dos dados. O objetivo é realizar um estudo comparativo dos BDs chave-valor com armazenamento em memória Redis, Memcached, Voldemort, Aerospike, Hazelcast e Riak KV. Assim, o trabalho contribuiu para uma análise de diferentes BDs e com resultados que demonstraram qualitativamente as características e apontaram as principais vantagens.*

1. Introdução

As necessidades de alta disponibilidade e velocidade, bem como o aumento e consumo de dados nos sistemas e aplicações atuais, influenciaram no desenvolvimento de novas formas para o armazenamento de dados. Estas vão além dos bancos de dados relacionais, impulsionando o movimento NoSQL (*Not only SQL*) [Fowler 2015]. Não obstante, com a popularidade em alta em um mercado sem um líder estabelecido, há uma consequente explosão no número de sistemas de armazenamento disponíveis, o que dificulta a tomada de decisão quanto à opção que melhor supre as necessidades organizacionais.

Com o objetivo de solucionar problemas específicos, os bancos NoSQL foram categorizados de acordo com suas características e otimizações. Por exemplo, a Amazon utiliza seu sistema chave-valor (*key-value*) Dynamo [DeCandia et al. 2007] para gerenciar as listas de mais vendidos, carrinhos de compras, preferências do consumidor, gerenciamento de produtos, entre outras aplicações. Exitem também sistemas de família de colunas (*column family* ou *columnar*) que foram influenciados pelo Bigtable da Google [Chang et al. 2008]. Enquanto isso, os assim chamados de sistemas de documentos (*document*) resultaram, por exemplo, no MongoDB[1]. Por fim, do chamado banco triplo

[1]https://www.mongodb.com

(*graph database* ou *triple*), tem-se como exemplo o Neo4j[2]. Cada uma destas categorias traz sistemas que cobrem diferentes limitações dos bancos relacionais tradicionais.

O foco deste trabalho está nos bancos de dados chave-valor com armazenamento em memória, a fim de realizar um estudo comparativo e qualitativo de banco de dados ainda pouco estudados na literatura (Seção 2). Este tipo de banco de dados NoSQL é o representante com as estruturas mais simples dentre os existentes [Pokorny 2013]. No entanto, cabe ressaltar que eles possuem um amplo espectro de casos de uso, sendo largamente adotados tanto como armazenamento primário quanto para auxiliar outros sistemas de armazenamento [Carlson 2013]. Ainda, o aumento na capacidade das memórias RAM fez com estes sistemas se adaptassem para efetuar o armazenamento e processamento de dados em memória, com a finalidade de melhorar o desempenho [Zhang et al. 2015]. A realidade corrobora a afirmação de Jim Gray de que memória é o novo disco e o disco é a nova fita [Robbins 2008], o que justifica o aumento da popularidade destes sistemas.

Este artigo está organizado em 5 seções, incluindo esta seção introdutória. Na Seção 2 estão os trabalhos relacionados. A Seção 3 traz o embasamento sobre os bancos de dados pesquisados. Na Seção 4 está detalhado o estudo comparativo destes sistemas. Por fim, na Seção 5 estão as conclusões e propostas para trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção é apresentada uma discussão sobre os trabalhos relacionados publicados recentemente na literatura. De forma semelhante, todos os trabalhos escolhidos buscam caracterizar e comparar bancos de dados NoSQL. No entanto, não voltam os estudos para um determinado tipo de banco de dados, o que é importante para avaliar características específicas. Tanto [Hecht and Jablonski 2011] quanto [Han et al. 2011] se propõem a fazer um estudo e avaliação dos bancos de dados NoSQL, com o mesmo objetivo principal: prover informações para auxiliar na escolha do banco NoSQL que melhor atende às necessidades. Por não delimitarem um tipo específico de banco NoSQL, ambos possuem um escopo mais abrangente do que o presente trabalho. No ponto de intersecção entre este trabalho e os citados, [Hecht and Jablonski 2011] inclui em seu trabalho os bancos Project Voldemort, Redis e Membase, enquanto que [Han et al. 2011] avalia Redis, Tokyo Cabinet-Tokyo Tyrant e Flare.

Em [Deka 2014] é apresentada uma visão geral de vários sistemas NoSQL e os representantes chave-valor incluídos na avaliação são Hypertable, Voldemort, Dynomite, Redis e Dynamo. Nota-se a falta, porém, de uma visão comparativa mais clara sobre aspectos de garantias de durabilidade, disponibilidade, protocolos suportados, e outras informações que podem vir a ter uma influência significativa na escolha de um banco de dados chave-valor. Já [Zhang et al. 2015] traz uma visão bem estruturada dos objetivos que nortearam o projeto de cada um dos sistemas descritos no trabalho, o qual foca em sistemas com gerenciamento e processamento de dados em memória. Dentre os sistemas estudados, os representantes dos bancos chave-valor são MemepiC, RAMCloud, Redis, Memcached, MemC3 e TxCache. Dos sistemas, o trabalho descreve as cargas de dados mais adequadas ao sistema, a estratégia para construção de índices, o controle de concorrência, tolerância a falhas, tratamentos para conjuntos de dados maiores do que a memória disponível e o suporte a consultas personalizadas em baixo nível (como *stored procedures* e *scripts* em linguagem nativa, por exemplo), porém com pouca abordagem de alto nível que auxilie na escolha de um sistema em favor de outro.

[2]https://neo4j.com

Ainda que o tema NoSQL tenha sido bastante explorado na academia, e a bibliografia focada no armazenamento de dados em memória tenha crescido muito nos últimos anos, nota-se a falta de um estudo comparativo entre os sistemas chave-valor em memória Redis, Memcached, Voldemort, Aerospike, Hazelcast e Riak KV.

3. Banco de Dados com Armazenamento em Memória

O crescimento dos bancos de dados com armazenamento de dados em memória (IMDB ou *in-memory databases*) segue uma tendência que teve seu início no hardware, com a capacidade da memória dobrando em média a cada três anos e seu preço caindo uma casa decimal a cada cinco anos [Zhang et al. 2015]. As memórias não-voláteis (NVM ou *Non-Volatile Memory*) como o SSD (*Solid State Disk*), também têm evoluído, mas seu custo [Kasavajhala 2011], durabilidade e confiabilidade [Schroeder et al. 2016] continuam sendo impeditivos para a maioria das aplicações.

A vantagem em manter os dados na memória ao invés do disco está relacionada à latência de acesso a estes dados, pois remove-se a necessidade de acessar a camada mais lenta da hierarquia de memória, conforme demonstrado pela Figura 1 (adaptada de [Zhang et al. 2015]), que detalha as camadas de armazenamento, bem como uma estimativa de sua capacidade atual e da latência de acesso aos dados nela armazenados.

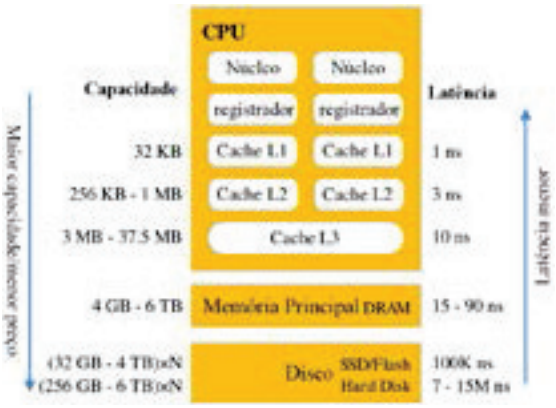


Figura 1. Hierarquia de memória.

Para que os dados possam ser processados pela CPU é necessário que estes estejam nos registradores e para tal é preciso que estes dados passem por todas as camadas da hierarquia de memória até chegarem aos registradores [Zhang et al. 2015]. Como pode ser visto na Figura 1, o disco é a camada mais distante e mais lenta, porém com a maior capacidade de armazenamento. Com o aumento da capacidade da camada de memória principal (composta pela memória RAM) os IMDB buscam trazer os dados para esta camada e evitar o nível mais lento da hierarquia de memória.

Dentre a miríade de bancos de dados com armazenamento em memória, os bancos chave-valor podem ser considerados os representantes mais versáteis, simples e com melhor desempenho, advindo principalmente da sua simplicidade [Pokorny 2013]. Portanto, muitos sistemas desta categoria sacrificam garantias de consistência em favor do desempenho [DeCandia et al. 2007] [Fowler 2015]. Nestes bancos, cada valor armazenado está vinculado a uma chave que identifica unicamente um valor [Han et al. 2011], sendo que este valor pode ser tanto um conteúdo binário quando uma estrutura de dados complexa, conforme as funcionalidades oferecidas pelo banco [Pokorny 2013]. Nas pró-

ximas seções são apresentados e discutidos os sistemas de armazenamento chave-valor em memória Redis, Memcached, Voldemort, Aerospike, Hazelcast e Riak KV.

3.1. Redis

Redis (*REmote DIctionary Server*) [Redis 2009] é um sistema de armazenamento de dados estruturados em memória que pode ser utilizado como banco de dados, *cache* e *message broker* [Cao et al. 2016]. Ele opera em um modelo cliente-servidor através de conexões TCP utilizando um protocolo próprio chamado RESP (REdis Serialization Protocol).

O modelo de dados do Redis é composto por cinco estruturas de dados diferentes para os valores (*string*, *list*, *set*, *sorted set* e *hash*), a persistência dos dados da memória em disco através de dois métodos (*snapshopts* chamados RDB e *append-only file* ou AOF) [Zhang et al. 2015]. A possibilidade de escalabilidade horizontal através do Redis Cluster foi adicionada apenas em 2015, na versão 3.0 do sistema. Segundo os autores de [Sanfilippo 2010], um sistema deve ser eficiente em um único nó quando for escalado.

3.2. Memcached

O Memcached [Memcached 2003] caracteriza-se como um sistema genérico de *cache* em memória. Ele foi construído pensando na melhoria de desempenho de aplicações *web* através da redução na demanda de requisições ao banco de dados em disco. Brad Fitzpatrick desenvolveu ele para melhorar o desempenho do site Livejournal.com através de uma solução melhor de *cache* [Galbraith 2009]. A sua implementação é na linguagem Perl e posteriormente reescrito em C. O Memcached utiliza uma arquitetura *multi-thread* e o controle de concorrência interno é feito através de uma *hash-table* estática de *locks* [Zhang et al. 2015].

A classificação do Memcached como banco de dados é discutível, uma vez que o mesmo não implementa persistência, failover [Galbraith 2009] nem escalabilidade horizontal, pois a distribuição dos dados entre múltiplas instâncias do sistema deve ser feita pelo cliente [Zhang et al. 2015]. O funcionamento do Memcached segue o modelo cliente-servidor e a comunicação ocorre através de conexões TCP ou UDP utilizando um protocolo próprio que suporta textos puros em ASCII ou dados binários [Soliman 2013].

3.3. Voldemort

O Voldemort [Voldemort 2009] foi desenvolvido pelo LinkedIn em linguagem Java com o objetivo de gerenciar funcionalidades dependentes de associações entre dados da rede social, tais como a recomendação de relacionamentos através da análise dos relacionamentos atuais [Sumbaly et al. 2012].

O Voldemort é inspirado no Dynamo, da Amazon [DeCandia et al. 2007], oferece comandos simples (put, get e delete) [Sumbaly et al. 2013] e uma arquitetura completamente distribuída, onde cada nó é independente e não existe um servidor principal de coordenação [Deka 2014]. O sistema é completamente modularizado e tanto a serialização dos dados quanto a persistência são oferecidas através de módulos plugáveis. Segundo [Sumbaly et al. 2012], a grande vantagem do Voldemort em relação ao Dynamo, é um mecanismo próprio de armazenamento desenhado para o pré-carregamento de grandes volumes de dados, em que o Voldemort passa a funcionar em modo somente leitura.

3.4. Aerospike

O Aerospike [Aerospike 2012] tem uma arquitetura modelada com foco em velocidade na análise de dados, escalabilidade e confiabilidade para aplicações *web*. Esse banco de

dados se apresenta como uma solução para a combinação de diferentes tipos de dados e também acessos por milhares de usuários. Pensando nisso, as suas operações são focadas em chave-valor e otimizadas para o uso da memória RAM em conjunto com memórias *flash* (NVM) [Aerospike 2012].

Diferente de vários de seus concorrentes, o próprio Aerospike disponibiliza bibliotecas de integração aos clientes, a fim de melhorar o desempenho na sua utilização. Quanto ao *cluster*, todos os nós são iguais, em uma arquitetura conhecida como *shared nothing*. A respeito do *server* é possível utilizar índices secundários e definir funções para otimizar a utilização dos dados. E por fim, a camada de armazenamento incorpora a utilização da memória RAM e de sistemas de armazenamento permanente.

3.5. Hazelcast

O Hazelcast [Hazelcast 2009] é uma ferramenta distribuída sob licença *open source* e comercial desenvolvida em Java. Possui seu foco em computação distribuída e escalabilidade horizontal, se destacando dos concorrentes por oferecer as garantias ACID (Atomicidade, Consistência, Isolamento e Durabilidade) dos bancos de dados relacionais tradicionais.

O *cluster* funciona em uma arquitetura *shared nothing*, onde não existe um ponto único de falha. Além de oferecer clientes para as linguagens comuns como Java, C, C++ e C#, o Hazelcast oferece uma API REST, está preparado para trabalhar com o protocolo de comunicação do Memcache e pode ser utilizado através do Hibernate [Hazelcast 2009].

3.6. Riak KV

O Riak KV [Basho 2009] possui como principal objetivo oferecer disponibilidade máxima, com escalabilidade horizontal em forma de *cluster*, sendo considerado um banco de dados de simples operação e fácil escalabilidade. Em sua versão comercial há suporte a *multi-cluster replication*, ou seja, é possível realizar a replicação de dados através de diferentes *clusters*, geograficamente distantes, a fim de reduzir a latência de acesso uniformemente para clientes através do globo.

Nota-se claramente a influência do Dynamo [DeCandia et al. 2007] no Riak KV, desde suas funcionalidades para execução distribuída até nas configurações do fator de replicação e arquitetura *shared nothing*.

4. Estudo Comparativo

Esta seção apresenta uma comparação entre os sistemas apresentados anteriormente. Para isso, as características foram classificadas em três grupos: (I) características mercadológicas, onde são explorados aspectos sem relação direta com funcionalidades ou o funcionamento do banco, tais como ano de lançamento, licenciamento e linguagem de desenvolvimento; (II) características do projeto, onde são descritas definições decididas no projeto do sistema, tais como escalabilidade, disponibilidade e consistência; e (III) características de manutenção, onde são explanadas os aspectos que tem relação direta com a manutenção e suporte ao sistema, tais como ferramentas internas para monitoramento e interface de gerenciamento.

Na Tabela 1 é possível avaliar: o ano em que a primeira versão do sistema foi lançada, os licenciamentos sob os quais o *software* é distribuído, a linguagem na qual o sistema foi desenvolvido, os sistemas operacionais suportados, as linguagens nas quais são oferecidos clientes para comunicação e os protocolos de comunicação suportados. A

partir dos dados disponibilizados, os interessados podem avaliar a maturidade do sistema, se a licença está alinhada com as necessidades empresariais e se a infraestrutura disponível e o esforço de implementação estão dentro do esperado.

Vale ressaltar que o Redis não suporta oficialmente Windows, mas uma versão para Windows x64 é mantida pela equipe da MS Open Tech (Microsoft Open Technologies). Quanto ao Voldemort e ao Hazelcast, como os mesmos rodam na JVM (Java Virtual Machine), podem-se considerar os sistemas operacionais suportados por esta tecnologia. Tanto Aerospike quanto Riak KV oferecem pacotes para sistemas baseados nas distribuições Linux Red Hat, Debian e Ubuntu. O Aerospike ainda oferece sua execução no OS X e Windows através de máquinas virtuais.

Tabela 1. Características mercadológicas

	Redis	Memcached	Voldemort	Aerospike	Hazelcast	Riak KV
Lançamento	2009	2003	2009	2012	2009	2009
Licenciamento	BSD-3 e comercial	BSD-3	Apache 2	AGPL e comercial	Apache 2 e comercial	Apache 2 e comercial
Desenvolvido	C	C	Java	C	Java	Erlang
SO Suporte	Linux, BSD, OSX e Windows	Debian/Ubuntu e Windows	JVM	Linux, OS X e Windows	JVM	Linux
Clientes	48 linguagens	Não existe listagem oficial	4 linguagens	12 linguagens	6 linguagens	21 linguagens
Protocolos	Próprio (RESP)	Próprio	HTTP, Socket, NIO	Próprio e JDBC	Próprio e Memcached	API HTTP e próprio

Quanto ao item linguagens com cliente, é importante notar que foram considerados apenas as linguagens e clientes listados no site oficial de cada sistema e que o site do Memcached não oferece uma listagem oficial das linguagens suportadas. Nota-se também que as linguagens C++, Java e Python são as únicas para as quais todos os sistemas possuem clientes. Os protocolos de comunicação são o meio de comunicação entre o cliente e o servidor, porém, na maioria dos casos a comunicação é feita através de um dos clientes já construídos e a empresa não precisa se preocupar com o protocolo utilizado pelo cliente.

Na Tabela 2 é possível avaliar: as opções para escalabilidade horizontal (ou *clusterização*), a classificação do sistema segundo o teorema CAP [Brewer 2000], como é feito o controle de concorrência, o suporte à transações ACID, as opções de persistência dos dados em disco, o suporte a dados complexos e as opções para autenticação do cliente. Analisando a Tabela 2 é possível avaliar se as funcionalidades e características do banco de dados atendem às demandas e características referentes aos dados que se pretende armazenar nos mesmos.

Quanto à escalabilidade, todos os bancos exceto o Memcached incluem suporte a *sharding* e replicação, sendo que a maioria (a exemplo do Dynamo [DeCandia et al. 2007]) oferecem fator de replicação configurável para leituras e escritas. O Redis utiliza o modelo master/slave, comumente utilizado nas bases de dados relacionais tradicionais.

O teorema CAP (*Consistency*, *Availability* e *Partition tolerance*) foi proposto por Eric Brewer em [Brewer 2000] e verificado em [Gilbert and Lynch 2002], desde então passou a ser largamente aceito pela academia. O teorema afirma que na existência de uma falha de comunicação (*partition*) cada nó de um sistema distribuído deve escolher entre responder requisições, mantendo a disponibilidade (*availability*) e assumindo o risco de não retornar os dados mais atuais, ou rejeitar requisições para garantir a consistência dos dados (*consistency*). Sistemas classificados como AP priorizam a disponibilidade, en-

quanto que sistemas classificados como CP priorizam a consistência. O teorema tem sido alvo de muitas críticas e Brewer explora algumas de suas limitações em [Brewer 2012], enquanto Abadi propõe o teorema PACELC como alternativa em [Abadi 2012].

Quanto à classificação do Redis, é importante mencionar que o ele não atende todos os requisitos de um sistema CP de [Brewer 2000], por usar replicação assíncrona entre os nós do *cluster*. O teorema CAP também não se aplica ao Memcached pelo fato de ele não suportar a criação de *clusters* e, portanto, não haver comunicação entre nós. O Riak KV possui uma configuração onde é possível definir qual dos atributos devem ser preservados (disponibilidade ou consistência).

Tabela 2. Características do projeto

	Redis	Memcached	Voldemort	Aerospike	Hazelcast	Riak KV
Escalabilidade horizontal	Mestre - Escravo	Não	Fator de Replicação	Fator de Replicação	Fator de Replicação	Fator de Replicação
Teorema CAP	CP	N/A	AP	AP	AP	Config.
Controle Concorrência	Single-thread	Mutex lock	MVCC	Test-and-set	Multi-single-thread	MVCC
Transações	Parcial	Não	Não	Parcial	Sim	Não
Persistência em disco	RDB e AOF	Não	Config.	Assínc.	Banco auxiliar	Banco auxiliar
Suporte a dados complexos	Sim	Não	Sim	Sim	Sim	Sim
Autenticação	Simples	SASL	Kerberos	Somente comercial	Simples, SSL, Kerberos,IP	Sim, e autorização

O controle de concorrência é implementado de diferentes maneiras. No Redis a execução é *single-thread* e as requisições são processadas de forma assíncrona internamente [Zhang et al. 2015], sendo que apenas um *master* responde por uma determinada chave, portanto não há concorrência. O Memcached utiliza *mutex* (*mutual exclusive*) *lock*. Tanto Voldemort quanto Riak KV seguem a implementação do Dynamo [DeCandia et al. 2007] e utilizam *vector clocks*, uma implementação do versionamento baseado em locking otimista conhecida por MVCC (Multi Version Concurrency Control). O Aerospike utiliza o método conhecido como *test-and-set* ou *check-and-set* (CAS), uma operação atômica implementada a baixo nível que escreve em um local de memória e retorna o valor antigo. No Hazelcast, é criada uma *thread* para atender cada uma das partições internas de dados, portanto ainda que ele seja *multi-thread*, uma chave específica está num contexto *single-thread* e, portanto, não há concorrência.

Quanto as transações, há um nível bastante variado de suporte oferecido pelos sistemas. Ainda que o Redis tenha suporte básico a transações, estas não possuem opção de rollback e a durabilidade da mesma depende da persistência em disco. Memcached, Voldemort e Riak KV declaram não suportarem transações, enquanto que o Aerospike suporta transações que envolvam uma única chave ou que sejam somente leitura, no caso de envolverem múltiplas chaves. O Hazelcast se destaca sendo o único a oferecer transações ACID completas.

A persistência em disco é oferecida por todos os bancos, exceto o memcached. No Redis são oferecidas duas formas complementares: *snapshot* (RDB), onde todos os dados na memória são gravados em disco, e *append-only file* (AOF), onde cada operação é gravada em um arquivo de log e o arquivo é reescrito quando chega em um tamanho pré-determinado. O Voldemort oferece opções para configurar a persistência como síncrona (*write through*, onde a operação é persistida antes do retorno ao cliente) ou assíncrona (*write behind*, onde o cliente recebe a confirmação e posteriormente a operação é persistida), enquanto que o Aerospike faz a persistência de forma assíncrona. Vale mencionar

que o Aerospike tem melhorias focadas no uso de SSD como dispositivo de armazenamento permanente. Tanto Hazelcast como Riak KV oferecem persistência através do acoplamento de um banco de dados auxiliar e a assincronicidade é configurável.

Referente ao suporte a dados complexos, vale notar que todos os sistemas, exceto o Memcached, suportam chaves do tipo lista e hashtable (ainda que com nomes diferentes). Hazelcast e Voldemort se baseiam fortemente nas classes do Java, Redis e Riak KV ainda oferecem suporte ao tipo HyperLogLogs, enquanto Redis e Aerospike oferecem suporte a tipagem ou comandos relativos a georreferenciamento.

Finalizando, enquanto que Redis oferece autenticação simples através de credenciais pré-configuradas, o Memcached oferece autenticação através do protocolo SASL (*Simple Authentication and Security Layer*). O Voldemort é integrado ao protocolo Kerberos. O Aerospike oferece autenticação apenas em sua versão com licenciamento comercial. O Hazelcast oferece todos os protocolos supracitados e o SSL. Por último, o Riak KV oferece um sistema próprio de usuários e grupos, com autenticação e autorização baseada em diversos mecanismos, incluindo senhas e certificados digitais.

Na Tabela 3 está descrita a existência de interfaces de gerenciamento, ferramentas de monitoramento e benchmarks para os sistemas estudados. É importante notar que foram avaliadas apenas as ferramentas oficiais dos desenvolvedores dos sistemas. Portanto, muitas destas ferramentas, ainda que não estejam declaradas aqui, já foram desenvolvidas pela comunidade e estão disponíveis. Estas informações são particularmente interessantes para avaliar o esforço de manutenção que será despendido após a implantação do sistema.

Tabela 3. Características de manutenção

	Redis	Memcached	Voldemort	Aerospike	Hazelcast	Riak KV
Interface de Gerenciamento	Não	Não	Básica	À parte	Comercial	Sim
Ferramentas de Monitoramento	'INFO'	'stats'	JMX	'asadm'	JMX	'stats'
Benchmark embutido	Sim	Não	Sim	Sim	Não	Sim

Quanto às interfaces de gerenciamento oferecidas pelos sistemas avaliados, o Redis e o Memcached são os únicos que não as oferecem nativamente (ainda que existam opções na comunidade) e o Voldemort oferece uma interface básica à parte escrita em Ruby, que está sem manutenção. O Aerospike oferece uma interface que deve ser instalada à parte. No Hazelcast, esta funcionalidade está disponível apenas na versão comercial. O Riak KV é o único onde a interface de gerenciamento já está integrada ao código principal do programa, não requerendo nenhuma instalação extra.

A respeito de ferramentas de monitoramento, o Redis oferece comandos como *INFO*, *MEMORY* e *LATENCY*, enquanto que o Memcached oferece o comando *stats* e o Aerospike oferece o comando *asadm*. O Voldemort possui uma interface completa de monitoramento exposta através de *Java Management Extensions* (JMX). Esta é a mesma estratégia utilizada pelo Hazelcast. O Riak KV oferece os comandos *stat* e *stats* em sua interface de linha de comando (CLI) *riak-admin* e a URL */stats* em sua API HTTP.

No item *benchmark* embutido foi avaliado se são disponibilizados *benchmarks* junto com o sistema, cujo o principal objetivo é avaliar o desempenho do sistema em determinada infraestrutura. Enquanto que Memcached e Hazelcast não oferecem ferramentas próprias para realização de *benchmark*, no Redis existe a ferramenta *redis-benchmark* e o Voldemort oferece a *voldemort-performance-tool*. No Aerospike os *benchmarks* estão nos clientes disponibilizados e no Riak KV o nome dado ao *benchmark* é *Basho Bench*.

Após comparar as características dos bancos de dados chave-valor com armazenamento em memória, Redis, Memcached, Voldemort, Aerospike, Hazelcast e Riak KV, é fácil entender o motivo pelo qual o Memcached não é considerado um banco de dados, pois seu foco destoa bastante de seus semelhantes. É possível perceber também que, mesmo que o Redis tenha oferecido suporte à clusterização em suas versões mais recentes, os outros sistemas ainda estão à frente quando o assunto é funcionalidades para execução distribuída. É possível perceber também como Voldemort e Hazelcast se utilizam do ecossistema Java para prover funcionalidades interessantes e como o paper do Dynamo [DeCandia et al. 2007] influencia principalmente Voldemort e Riak KV.

5. Conclusões

Após estudar os bancos chave-valor com armazenamento em memória, é possível notar que mesmo um subconjunto específico de bancos NoSQL traz muitas variáveis. Neste sentido, destaca-se a grande quantidade de características que devem ser cuidadosamente avaliadas pelo analista para a correta tomada de decisão quanto ao banco mais adequado às necessidades. Dentre estas características, destacam-se o padrão de busca e gravação de dados da aplicação cliente, a importância da durabilidade dos dados, o comportamento desejado frente a partições no *cluster*, o ambiente de infraestrutura onde a solução será implantada, o ambiente de desenvolvimento da aplicação cliente e as perspectivas de crescimento na demanda da aplicação.

Com as características dos sistemas esclarecidas, percebe-se que outro fator importante para a escolha do banco de dados chave-valor com armazenamento em memória a ser adotado é o desempenho, que é justamente o ponto que traz mais interesse a esta categoria de sistemas. Como trabalho futuro, propõe-se uma avaliação do desempenho dos sistemas aqui estudados, comparando o desempenho das variadas características compartilhadas pelos mesmos.

Referências

[Abadi 2012] Abadi, D. (2012). Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. *Computer*, 45(2):37–42.

[Aerospike 2012] Aerospike (2012). Aerospike | High Performance NoSQL Database. Access on <http://www.aerospike.com/>.

[Basho 2009] Basho (2009). Riak KV. Access on <http://basho.com/products/riak-kv/>.

[Brewer 2000] Brewer, E. (2000). Towards Robust Distributed Systems. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 7–, New York, NY, USA. ACM.

[Brewer 2012] Brewer, E. (2012). CAP twelve years later: How the "rules"have changed. *Computer*, 45(2):23–29.

[Cao et al. 2016] Cao, W., Sahin, S., Liu, L., and Bao, X. (2016). Evaluation and Analysis of In-Memory Key-Value Systems. In *2016 IEEE International Congress on Big Data (BigData Congress)*, pages 26–33.

[Carlson 2013] Carlson, J. L. (2013). *Redis in Action*. Manning, Shelter Island, NY, USA.

[Chang et al. 2008] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Trans. on Computer Systems (TOCS)*, 26(2):4.

[DeCandia et al. 2007] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6):205–220.

[Deka 2014] Deka, G. C. (2014). A survey of cloud database systems. *IT Professional*, 16(2):50–57.

[Fowler 2015] Fowler, A. (2015). *NoSQL For Dummies*. John Wiley & Sons, 111 River Street, Hoboken, New Jersey, USA.

[Galbraith 2009] Galbraith, P. (2009). *Developing Web Applications with Apache, MySQL, memcached, and Perl*. John Wiley & Sons.

[Gilbert and Lynch 2002] Gilbert, S. and Lynch, N. (2002). Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. *SIGACT News*, 33(2):51–59.

[Han et al. 2011] Han, J., E, H., Le, G., and Du, J. (2011). Survey on NoSQL database. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 363–366.

[Hazelcast 2009] Hazelcast (2009). Hazelcast the Leading In-Memory Data Grid - Hazelcast.com. Access on <https://hazelcast.com/>.

[Hecht and Jablonski 2011] Hecht, R. and Jablonski, S. (2011). NoSQL evaluation: A use case oriented survey. In *2011 International Conference on Cloud and Service Computing (CSC)*, pages 336–341.

[Kasavajhala 2011] Kasavajhala, V. (2011). Solid State Drive vs. Hard Disk Drive Price and Performance Study. *Proc. Dell Technical White Paper*, pages 8–9.

[Memcached 2003] Memcached (2003). memcached - a distributed memory object caching system. Access on <https://memcached.org/>.

[Pokorny 2013] Pokorny, J. (2013). NoSQL databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1):69–82.

[Redis 2009] Redis (2009). Redis.io. Access on <http://redis.io/>.

[Robbins 2008] Robbins, S. (2008). RAM is the new disk... Access on <https://www.infoq.com/news/2008/06/ram-is-disk>.

[Sanfilippo 2010] Sanfilippo, S. (2010). On Redis, Memcached, Speed, Benchmarks and The Toilet . Access on <http://antirez.com/post/redis-memcached-benchmark.html>.

[Schroeder et al. 2016] Schroeder, B., Lagisetty, R., and Merchant, A. (2016). Flash Reliability in Production: The Expected and the Unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST '16)*, FAST '16, pages 67–80, Santa Clara, CA, USA.

[Soliman 2013] Soliman, A. (2013). *Getting Started with Memcached*. Packt.

[Sumbaly et al. 2012] Sumbaly, R., Kreps, J., Gao, L., Feinberg, A., Soman, C., and Shah, S. (2012). Serving large-scale batch computed data with Project Voldemort. In *Proceedings of the 10th USENIX conference on File and Storage Technologies*, page 18.

[Sumbaly et al. 2013] Sumbaly, R., Kreps, J., and Shah, S. (2013). The Big Data Ecosystem at LinkedIn. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1125–1134, New York, NY, USA. ACM.

[Voldemort 2009] Voldemort (2009). Project Voldemort. Access on <http://www.project-voldemort.com/>.

[Zhang et al. 2015] Zhang, H., Chen, G., Ooi, B. C., Tan, K.-L., and Zhang, M. (2015). In-Memory Big Data Management and Processing: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1920–1948.

Artigos de Aplicações e Experiências

MahoutGUI: Uma Interface Gráfica para Gerar Recomendações com o Apache Mahout Diretamente de Banco de Dados usando Mapeamento Objeto-Relacional

Gláucio R. Vivian[1], **Cristiano R. Cervi**[1]

[1]Instituto de Ciências Exatas e Geociências (ICEG)
Universidade de Passo Fundo (UPF) – Passo Fundo – RS – Brazil

{149293,cervi}@upf.br

Abstract. *The Recommender Systems are responsible by assisting the users in information filtering process. This work reports the development of a graphical user interface to recommender of Apache Mahout. We also propose a java class to access the data stored in relational data base through Object-Relational Mapping using the Hibernate framework. The first contribution of this work is the simplification of the user interaction with system, obtained through the graphical user interface. The second is the flexibility and portability to different relational data base managers systems obtained through Hibernate.*

Resumo. *Os Sistemas de Recomendações são responsáveis por auxiliar os usuários no processo de filtragem de informações. Este trabalho relata o desenvolvimento de uma interface gráfica para o recomendador do Apache Mahout. Também propomos uma classe java de acesso aos dados armazenados em banco de dados relacionais por meio do mapeamento objeto-relacional usando o framework Hibernate. A primeira contribuição deste trabalho é a simplificação da interação do usuário com o sistema, obtida pela interface gráfica. A segunda é a flexibilidade e portabilidade para diferentes gerenciadores de banco de dados obtida por meio do Hibernate.*

1. Introdução

Os Sistemas de Recomendação estão amplamente difundidos, seja no comércio eletrônico, redes sociais, notícias, conteúdo sob demanda e muitas outras aplicações. Um dos mais importantes projetos de Sistema de Recomendação é o Apache Mahout[1]. Segundo [Owen et al. 2012], trata-se de um projeto *open source* com recursos de aprendizado de máquina e mineração de dados para ambientes de execução distribuída baseados no Apache Hadoop[2]. A paralelização é obtida por meio do paradigma de programação MapReduce apresentado por [Dean e Ghemawat 2008]. Os dados são armazenados em um sistema de arquivos distribuídos com tolerância a falhas chamado de *Hadoop Distributed File System* (HDFS).

O projeto conta com diversas e modernas técnicas de Classificação, Recomendação e Agrupamento (Clustering). As técnicas de recomendação disponibilizadas são fatoração de matriz (SVD) e filtragem colaborativa usuário-item e item-item.

[1]http://mahout.apache.org/
[2]http://hadoop.apache.org/

A leitura dos dados é abstraída por meio da classe DataModel, que pode buscá-los diretamente em arquivos no formato CSV ou por meio de conexões JDBC. Os parâmetros de configuração são bastante flexíveis, permitindo utilizar diversas técnicas de similaridades. No caso da técnica usuário-item, podem localizar os usuários por meio do algoritmo do vizinho mais próximo (*Nearest Neighbor*) baseado em limiar (*threshold*) ou n quantidades. Também existem algumas métricas (*Precision, Recall*, RMSE e MAE) para avaliar os resultados, permitindo dessa forma a realização completa de experimentos avaliativos. Uma deficiência no projeto é a inexistência de interface gráfica, sendo tudo realizado por meio do *shell* do Sistema Operacional ou biblioteca Java.

O *framework* Hibernate[3] possibilita um alto nível de abstração no acesso a informações armazenadas em banco de dados relacionais por meio da técnica de mapeamento objeto-relacional. Esta se caracteriza por representar os dados na forma de objetos, atributos e coleções do paradigma de programação orientada a objetos. Para o desenvolvedor, as particularidades do SGBD são todas abstraídas. Existe uma linguagem próxima da tradicional SQL, denominada de HQL que possibilita a execução de consultas.

O objetivo deste trabalho é apresentar uma interface gráfica de usuário que gera recomendações com o Apache Mahout. Além disso, construiu-se uma classe java para abstrair o acesso à diferentes SGBDs por meio do mapeamento objeto-relacional com o *framework* Hibernate.

Este artigo está organizado da seguinte forma: Na seção 2 são analisados alguns trabalhos correlatos. Na seção 3 é apresentado o aplicativo. Na seção 4 são apresentados os experimentos e resultados. Finalmente, na seção 5, são apresentadas as conclusões e trabalhos futuros.

2. Trabalhos Correlatos

[Akbarnejad et al. 2010] apresentam o QueRIE, trata-se de um sistema de recomendação que possibilita aos usuários gerarem consultas personalizadas em linguagem SQL para grandes bancos de dados relacionais. O sistema é constituído por dois diferentes motores de execução. O primeiro, chamado *Tuple-based*, identifica partes com interesse em potencial da base de dados que foram acessados por usuários similares no passado. O segundo, denominado *Fragment-based*, identifica consultas similares que outros usuários postaram para o usuário atual. Os experimentos foram conduzidos em dois diferentes cenários de consultas SQL: simples e complexas. Como resultados, demonstrou-se que a proposta habilita os usuários a gerarem recomendações com o SGBD SkyServer por meio da análise dos seus logs de consultas executadas.

Nos trabalhos de [Sarwat 2012, Sarwat et al. 2013] é apresentado o RecDB. Trata-se de um *fork* do Postgresql com a adição de um *engine* baseado no LensKit[4] do GroupLens[5] com os recursos para gerar recomendações usando a técnica de filtragem colaborativa do tipo item-item e usuário-item. O *engine* é constituído pelos seguintes módulos: i) Rec-Store: módulo responsável pelo armazenamento, manutenção e otimização dos dados. ii) Rec-tree: eficiente estrutura em árvore responsável e pela indexação dos atributos

[3]http://hibernate.org/orm/
[4]http://lenskit.grouplens.org/
[5]http://grouplens.org/

dos usuários/itens. iii) Rec-Query: processador de consultas SQL. Além disso, foi apresentada a adição de recursos ao padrão SQL com o objetivo de definir recomendadores e executar consultas com a solicitação de recomendações. Os autores colocam que a sua proposta apresenta as seguintes características: usabilidade, flexibilidade, facilidade de integração e eficiência. Foram realizados dois exemplos demonstrativos de aplicação, o primeiro com um sistema de recomendações para restaurante e o segundo sobre filmes.

3. A Aplicação Proposta

A aplicação proposta denominada MahoutGUI[6], deve simplificar a interação do usuário com as diversas opções apresentadas de recomendação. Na Figura 1 pode-se visualizar a interface gráfica da aplicação proposta.

(a) (b) (c)

Figura 1. Interface Gráfica da aplicação MahoutGUI.

Em (a) apresenta-se a entrada dos dados, em (b) as configurações e finalmente em (c) as recomendações geradas.

Além da interface gráfica proposta, desenvolveu-se uma classe de acesso a banco de dados utilizando-se o mapeamento objeto-relacional com o *framework* Hibernate. Isto possibilita que a aplicação seja mais flexível entre diversos gerenciadores de banco de dados. Na Figura 2 pode-se visualizar o diagrama de classes da aplicação proposta.

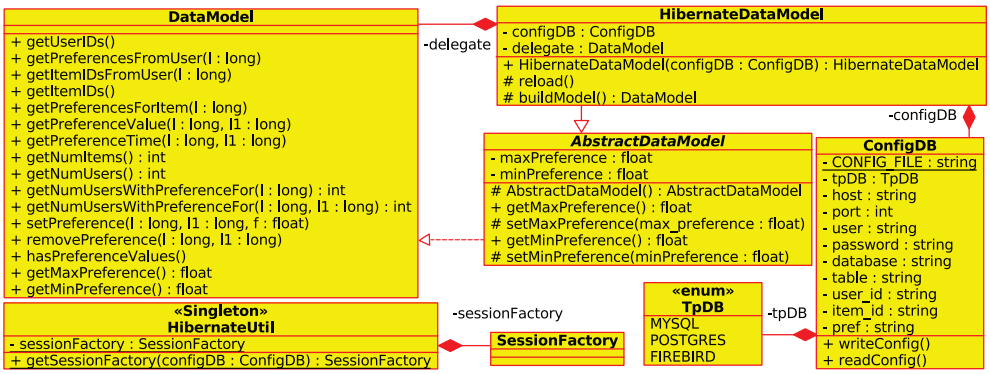


Figura 2. Diagrama de Classes para o mapeamento objeto-relacional.

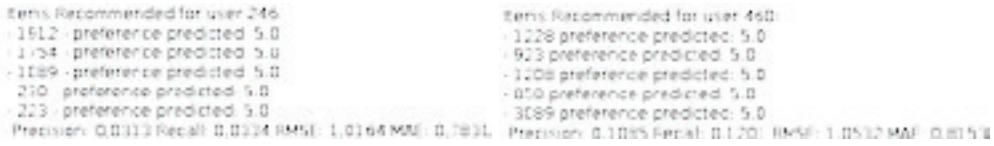
A classe HibernateDataModel é responsável pelo mapeamento objeto-relacional da aplicação. A mesma herda de AbstractDataModel, que por sua vez implementa DataModel. A classe ConfigDB é responsável por armazenar as informações de acesso ao

[6]https://github.com/grvivian/MahoutGUI

banco de dados. A enumeração TpDB define os SGBDs suportados pela aplicação proposta. A classe HibernateUtil implementa o *design pattern* Singleton, responsável por unificar o acesso ao atributo sessionFactory do Hibernate.

4. Experimentos e Resultados

Para testar a aplicação desenvolvida, utilizou-se um banco de dados PostgreSQL 9.5 com dados do projeto MovieLens de [Harper e Konstan 2016]. Ele contém 100.004 avaliações realizadas por 671 usuários sobre 9.125 filmes. Foram realizados dois testes solicitando 5 recomendações para os usuários 460 e 246 (escolhido aleatoriamente). Na Figura 3 pode-se visualizar as recomendações geradas.



(a) (b)

Figura 3. Resultado obtidos.

Em (a) foi utilizada filtragem colaborativa item-item usando a similaridade Log-Likelihood. Em (b) utilizou-se filtragem colaborativa usuário-item, com a mesma similaridade e com os 10 vizinhos mais próximos.

5. Considerações Finais e Trabalhos Futuros

Este trabalho apresentou o desenvolvimento do MahoutGUI, trata-se de uma interface gráfica para o Apache Mahout. A mesma possibilita de forma visual e intuitiva a realização rápida de experimentos com as técnicas de filtragem colaborativa. Além disso, desenvolveu-se uma classe para acesso à informações diretamente em banco de dados relacionais por meio do mapeamento objeto-relacional. Dessa forma, a aplicação proposta simplifica a realização de experimentos com sistemas de recomendação, possibilitando o acesso a usuários não especialistas da área. Como sugestão de trabalhos futuros, pretende-se suportar técnicas de fatoração de matriz e recomendações para usuários anônimos.

Referências

Akbarnejad, J., Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., On, D., Polyzotis, N., e Varman, J. S. V. (2010). Sql querie recommendations. *Proceedings of the VLDB Endowment*, 3(1-2):1597–1600.

Dean, J. e Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Harper, F. M. e Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19.

Owen, S., Anil, R., Dunning, T., e Friedman, E. (2012). *Mahout in action*. Manning Shelter Island, NY.

Sarwat, M. (2012). Recdb: towards dbms support for online recommender systems. In *Proceedings of the on SIGMOD/PODS 2012 PhD Symposium*, páginas 33–38. ACM.

Sarwat, M., Avery, J., e Mokbel, M. F. (2013). Recdb in action: recommendation made easy in relational databases. *Proceedings of the VLDB Endowment*, 6(12):1242–1245.

Desenvolvimento do Sistema de Acompanhamento do Fluxo de Demandas do Plano de Ação do Instituto Federal Farroupilha Campus Alegrete

Lenon Ricardo Machado de Souza[1], Marta Breunig Loose[2]

[1]Instituto Federal Farroupilha -- Campus Alegrete – (IFFar Campus Alegrete)
97.555-00 – Passo Novo – RS – Brasil

[2]Instituto Federal Farroupilha -- Campus Santo Ângelo – (IFFar Campus Santo Ângelo)
98.806-700 – Indúbras – RS – Brasil

lenonrmsouza@gmail.com, marta.breunig@iffarroupilha.edu.br

Abstract. *The Action Plan at the Federal Institute Farroupilha Campus Alegrete consists in a sectors demands list. The demands are goods and services purchase requests, which have a complex processing flow and are not always properly monitored. This paper describes the web system development that will allow the goods requests inclusion and also the demands flow monitoring of the Action Plan. The system uses the graph model, which is a NoSQL database category. The main reason for choosing this model is the characterization of the system stored data, being basically generated documents among several flow steps.*

Resumo. *O Plano de Ação no Instituto Federal Farroupilha Campus Alegrete consiste em uma lista de demandas de um setor. As demandas são pedidos de aquisição de bens e serviços, que possuem um fluxo de tramitação complexo e nem sempre tem o acompanhamento adequado. Este trabalho descreve o desenvolvimento de um sistema web que permitirá a inclusão de solicitações de bens e também o acompanhamento do fluxo das demandas referentes ao Plano de Ação. O sistema utiliza o modelo de grafos, que é uma categoria de banco de dados NoSQL. O principal motivo da escolha desse modelo é a caracterização dos dados armazenados no sistema, sendo basicamente documentos gerados ao longo de fluxo de várias etapas.*

1. Introdução

Anualmente no Instituto Federal Farroupilha Campus Alegrete (IFFar Campus Alegrete) é realizado o planejamento estratégico, que consiste na definição de ações de cada direção junto as suas coordenações através de diversas reuniões de viabilidade. Nessas reuniões são especificadas as prioridades que farão parte do Plano de Ação (PA) do próximo ano.

Atualmente, a solicitação das demandas é realizada manualmente, o que origina alguns problemas ao longo das interações com os setores envolvidos na verificação, validação e execução. Basicamente, esses problemas envolvem a dificuldade do solicitante na localização do documento, problemas de comunicação e falta de informações quanto ao atendimento das demandas.

Observando tais problemas, este trabalho descreve o desenvolvimento de um software que auxilia o acompanhamento da execução das demandas propostas no PA. Nesse sistema, os usuários poderão incluir solicitações de bens, além de visualizar a situação atual, o fluxo e os detalhes do atendimento das demandas. O sistema proporcionará o registro digital das demandas, permitindo também a definição de prazos para o atendimento em cada etapa, o que ajudará a diminuir os problemas citados anteriormente.

2. Banco de Dados Não-Relacional

Aproximadamente no ano de 2009 surgiu o conceito NoSQL, que busca suprir as necessidades do modelo relacional, contando com a alta performance e a rápida replicação de dados. As principais características desse modelo são a escalabilidade e a velocidade nas buscas dos valores armazenados.

Atualmente existem vários modelos de banco de dados não-relacionais disponíveis, sendo que cada um possui conceitos e características próprias, possibilitando sua aplicação de acordo com a necessidade dos desenvolvedores. Dentre os tipos de bancos de dados NoSQL estão os modelos chave-valor, documentos, colunas e grafos.

Especificamente, o modelo de grafos promove diversas soluções inovadoras para o armazenamento e processamento de grandes quantidades de dados. Segundo Vieira et al. (2012), tais soluções foram propostas devido a alguns problemas gerados por aplicações na Web 2.0, às quais necessitavam operar com grande volume de dados e uma arquitetura flexível, capaz de trabalhar com vários nós de processamento sem a necessidade de adicionar mais máquinas físicas.

Em relação à consistência desse modelo de banco de dados, nodos só poderão ser excluídos caso não exista nenhuma relação ligada a ele, garantindo que dados importantes não sejam perdidos acidentalmente. Não existem restrições em relação aos valores que podem ser armazenados no grafo, o que possibilita a criação de um nó com qualquer valor ou propriedade. Segundo Sadalage e Fowler (2013), um dos recursos interessantes dos bancos de dados de grafos é encontrar caminhos entre dois nós e dessa forma determinar suas características.

No sistema descrito nesse trabalho é utilizado o modelo de grafos. Sua escolha é mais adequada pois o fluxo do sistema possui características que remetem a esse modelo, como as etapas e seus respectivos documentos, que podem ser representadas pelos nós de um grafo. Já as ações de cada etapa possuem características semelhantes às arestas de um grafo. Neste trabalho é utilizada a ferramenta Neo4J para a criação dos grafos, em conjunto com a linguagem PHP além de outras tecnologias voltadas para o desenvolvimento web.

3. Estudo de Caso

No IFFar Campus Alegrete, o Plano de Ação consiste em um documento onde são registradas todas as necessidades anuais de um setor ou coordenação, sendo elaborado pelos coordenadores correspondentes a cada eixo tecnológico. O documento é enviado para os setores responsáveis pela avaliação, momento em que as demandas são estudadas a fim de verificar sua viabilidade e desta maneira as aquisições realizam-se ao longo do ano.

Porém todo esse processo é realizado de forma física e manual, ou seja, é necessário se trabalhar com documentos em papel e todo fluxo deve ser realizado e atendido pelas pessoas envolvidas, causando assim certos problemas. A dificuldade de comunicação entre as partes envolvidas, principalmente em relação ao atendimento as demandas, é o maior deles. Dessa forma, o sistema descrito neste trabalho auxiliará no gerenciamento do fluxo de execução do Plano de Ação digitalmente. Assim, o setor responsável por alguma etapa do atendimento das demandas participará do fluxo executando ações, tais como receber, encaminhar ou retornar, além de documentar e acompanhar a situação das mesmas.

4. Resultados e Discussões

Através do levantamento de requisitos foi possível especificar um fluxo para ser aplicado ao software. Conforme mostra a Figura 1, a demanda é iniciada a partir do pedido do usuário e posteriormente são realizadas outras etapas até a finalização daquela solicitação.

Figura 1. Fluxo resumido do sistema

O fluxo apresentado na Figura 1 foi resumido em sete etapas, para possibilitar a identificação do funcionamento do sistema proposto, porém no fluxo completo são realizadas o total de vinte e cinco etapas. Sendo que a duração aproximada de uma demanda no sistema é de um ano e seis meses.

Através da Figura 2 é possível visualizar a solicitação de bens, sendo que o solicitante preenche as informações a partir dos dados das demandas presentes no Plano de Ação do seu setor. Com isso, o documento da solicitação segue para a etapa de Avaliações (conforme Figura 1). Após a verificação da sua viabilidade, um novo documento é gerado a partir da solicitação original, com dados complementares e alterações necessárias para o prosseguimento do fluxo.

Figura 2. Tela de Solicitação da Demanda

5. Conclusões

O modelo de banco de dados de grafos, comparado ao modelo relacional, possui alguns benefícios que são importantes para a elaboração do projeto apresentado. Esses benefícios consistem na escalabilidade, velocidade nas consultas e tolerância a falhas. Tais características são essenciais para o desenvolvimento, além de possibilitar a manipulação de grandes quantidades de dados sem perder o desempenho.

A fase atual do projeto está compreendida no desenvolvimento da integração da linguagem PHP com a ferramenta Neo4J. Com o sistema será possível melhorar a execução das demandas, pois as etapas são realizadas digitalmente, facilitando a geração de documentos, a definição de status e indicação de observações pelos usuários. Uma vantagem importante será a agilidade na busca por informações de determinada demanda, já que através do sistema os usuários poderão verificar essas informações de maneira simplificada. Por fim, é importante ressaltar que a utilização do modelo de grafos para armazenar dados do sistema contribui para o desenvolvimento e a difusão dos bancos de dados NoSQL.

Referências

Lima, Claudio de; Mello, Ronaldo S. "Um Estudo sobre Modelagem Lógica para Bancos de Dados NoSQL."

Vieira, Marcos Rodrigues; et al. "Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de Big Data." *Simpósio Brasileiro de Bancos de Dados* (2012).

Sadalage, Pramod J.; Fowler, Martin. NoSQL Essencial: Um guia conciso para o Mundo emergente da persistência poliglota. Novatec Editora, 2013.

Consulta de Dados Espaciais em um Sistema de Informações de uma Bacia Hidrográfica

Vania Elisabete Schneider[1], Odacir Deonisio Graciolli[2], Helena Graziottin Ribeiro[3], Roberto Canuto Spiandorello[4], Guilherme Vanzin Hoffmann[5], Miguel Ângelo Pontalti Giordani[6]

[1,4,5,6]Instituto de Saneamento Ambiental – Universidade de Caxias do Sul (UCS)
CEP 95070-560 – Caxias do Sul – RS – Brazil

[2,3]Área de conhecimento de Ciências Exatas e Engenharias – Universidade de Caxias do Sul (UCS)
CEP 95070-560 – Caxias do Sul – RS – Brazil

veschnei@ucs.br, odgracio@ucs.br, hgrib@ucs.br, rcspiandorello@ucs.br, gvhoffma@ucs.br, mapgiordani@ucs.br

Abstract. *This article presents the gains obtained in an Information System of a river basin with the application of an extension association for spatial data manipulation in a database management system. It presents some spatial data manipulation processes and their applications, which will ultimately allow a better management decision-making, regarding the monitoring of phenomena inside the limits of the Taquari-Antas basin.*

Resumo. *Este artigo apresenta os ganhos obtidos em um Sistema de Informações de uma bacia hidrográfica com a associação da extensão para manipulação de dados espaciais PostGIS em um sistema gerenciador de banco de dados. Apresenta alguns processos de manipulação de dados espaciais e suas aplicações, que irão em última instância, permitir uma melhor tomada de decisões gerenciais, no que se refere ao acompanhamento dos fenômenos ocorridos dentro dos limites da bacia Taquari-Antas.*

1. Introdução

O Sistema de Informações em questão é uma demanda das pequenas centrais hidrelétricas (PCH) de uma bacia hidrográfica. Esse sistema foi desenvolvido para armazenar dados gerados em monitoramentos de qualidade da água, climatologia e fauna das diferentes instituições instaladas na bacia, visando assegurar a análise semântica e espacial, e também tem a função de gerar relatórios de acordo com as necessidades da gestão ambiental.

Os dados gerados nos monitoramentos de cada sub-bacia são armazenados no PostgreSQL - sistema de gerenciamento de banco de dados relacional, que apresenta recursos orientados a objetos como extensão do modelo relacional e fornece escalabilidade e suporte para tipos de dados complexos (objetos grandes, dados multimídia, dados espaciais, etc.) combinando os modelos relacional e orientado a objetos [1].

O PostgreSQL é habilitado para manipulação de dados espaciais através do PostGIS - um extensor de banco de dados espaciais, compatível com a OGC (*Open Geo Consortium*). É uma ferramenta livre e de código aberto que, segundo [Singh, S. P. & P. Singh 2014], contribui para uma implementação rápida e com pouco ou nenhum custo de software. O PostGIS adiciona funções espaciais para análise de componentes geométricos, manipulação de geometrias e determinação de relações espaciais.

O sistema utiliza o PostgreSQL com o PostGIS, mas algumas consultas exigiram um tratamento específico para manipulação de dados geográficos que não eram contemplados anteriormente. Essas consultas anteriores não estabeleciam relações entre os diferentes objetos espaciais (pontos, linhas, polígonos), resultando na subutilização das colunas para armazenamento de coordenadas geográficas. Este artigo apresenta o processo de inclusão de novas consultas ao sistema de informações para manipulação de dados espaciais, para atender a consultas que precisavam obter a localização de determinados pontos de monitoramento de dados relacionando-os a sua sub-bacia.

2. Banco de dados espacial no contexto do sistema de informação em questão

Os dados espaciais utilizados pelo sistema são pontos, linhas e polígonos, utilizados para representar visualmente os limites da bacia, das sub-bacias, pontos de fauna, pontos de qualidade da água, e pontos de barramento. Eles são armazenados e formatados conforme o código padrão internacional ou *Spatial Reference System Identifier* (SRID)[5].

O PostGIS - banco de dados espaciais, contém operadores que manipulam os objetos geométricos mais diretamente. Esse banco armazena e manipula objetos espaciais como qualquer outro objeto no banco de dados. Pode-se verificar se um ponto intercepta uma linha, se duas linhas se cruzam, se uma linha está completamente contida em um polígono, se um polígono está contido em outro, o que e é bem complicado de fazer se na consulta essas operações são escritas apenas usando pontos: a consulta pode ficar ineficiente.

O PostGIS permite solucionar problemas que abrangem análises baseadas no posicionamento dos objetos (pontos, linhas, polígonos) inseridos nas sub-bacias, estendendo as relações até então obtidas somente através de relacionamentos explícitos, como com a tabela empreendimento. Aplicações para a área ambiental incluem a aplicação de funções de interseção e distância para acompanhamento de áreas desflorestadas e cálculos de distâncias entre corpos de água [3]. Este extensor provê um "índice espacial", capaz de identificar objetos contidos dentro de outro e que estão dispostos em um espaço bidimensional, diferentemente dos índices tradicionais cuja ordenação é feita por strings, números ou datas [4]. O índice espacial, portanto, é utilizado para determinar a relação entre as geometrias selecionadas de maneira rápida: a relação é feita indexando a caixa delimitadora da geometria em vez da própria geometria. Os índices são o que torna possível usar um banco de dados espacial para grandes conjuntos de dados. Sem indexação, qualquer pesquisa de um recurso exigiria uma "varredura sequencial" de cada registro no banco de dados. A indexação acelera a

pesquisa organizando os dados em uma árvore de pesquisa que pode ser rapidamente percorrida para encontrar um registro específico[9].

O banco de dados trata as informações de projeção geográfica de maneira diferente da interface. Conversões devem ser realizadas para a correta exibição dentro dos limites do mapa projetado pelo *OpenLayers* - biblioteca de Javascript puro, gratuita, para exibição de dados em mapas, na maioria dos navegadores web modernos, sem depender do lado servidor. Essas conversões são realizadas durante a inserção e a consulta: a referência espacial no banco de dados é a EPSG:29182, já na consulta é a EPSG:4326. O Conjunto de Dados de Parâmetros Geodésicos EPSG é um conjunto de dados estruturado de Sistemas de Referência de Coordenadas e Transformações de Coordenadas, acessível através deste registro on-line (www.epsg-registry.org) ou, como arquivos zip descarregáveis, através da página inicial do EPSG em www.epsg.org. O EPSG se refere ao mercator esférico, um tipo de projeção que trata a Terra como uma esfera e é usado por grande parte dos principais provedores de APIs comerciais, como o Google Maps[6].

Os dados relativos a representação da geometria também passam por processos de conversão, que são tratados por funções próprias do PostGIS. A função *ST_GeomFromGeoJSON,* por exemplo*,* toma como entrada uma representação geojson de uma geometria e gera um objeto de geometria do PostGIS [7]. Outras funções semelhantes, abrem a possibilidade para a integração com tecnologias de inteligência de negócio geoespaciais [8], o que auxiliaria na ampliação de mecanismos para a tomada de decisões gerenciais pelos usuários do sistema.

3. Resultados e Considerações Finais

As novas consultas obtêm os pontos da sub-bacia relativos a qualidade da água e fauna utilizando o operador topológico *Contains,* que recebe como parâmetro o polígono da sub-bacia e as coordenadas do ponto [Figura 1].

```
SELECT sub_bacia.id_sub_bacia, sub_bacia.nome_bacia,
ponto_agua.id_ponto, ponto_agua.localizacao, ponto_agua.id_ponto_sia
FROM public.sub_bacia,qualidadeagua.ponto_agua
WHERE ST_Contains(sub_bacia.poligono,ponto_agua.coordenadas)
AND sub_bacia.id_sub_bacia = 3
ORDER BY id_sub_bacia, id_ponto_sia
```

Figura 1. Exemplo de consulta: pontos contidos na sub-bacia 3

Após a implementação das estruturas de consulta, conversão e armazenamento de dados espaciais, os resultados já podem ser auferidos pelo usuário do sistema. A figura 2 apresenta as melhorias observadas pelo usuário, que agora pode selecionar a sub-bacia de interesse para uma análise pontual sobre os itens espaciais associados. Observa-se no lado esquerdo as sub-bacias selecionadas e suas respectivas localizações no mapa com a cor de destaque. Verifica-se no banco de dados a evolução em relação aos atributos não espaciais. que descrevem qualitativa e quantitativamente uma entidade geográfica. A relação dos pontos com as sub-bacias foi melhorada, explorando os potenciais dos atributos espaciais, que consideram a localização e a representação do espaço geográfico usando um sistema de coordenadas. As novas consultas implementadas

estabelecem relações considerando os relacionamentos de vizinhança, ou presença dentro dos limites das sub-bacias.

Figura 2. Interface do sistema com acesso aos dados espaciais

4. Referências Bibliográficas

[1] Sabău, G. (2007) "Comparison of RDBMS, OODBMS and ORDBMS", In: Informatica Economica, Vol. XI, pp. 83-85.

[2] Singh, S. P., Singh, P. (2014) "Mapping Spatial Data on the Web Using Free and Open-Source Tools: A Prototype Implementation", In: Journal of Geographic Information System, 2014, 6, 30-39.

[3] Miranda D., Russo D., Alves R. A. L. (2012) "Upgrading to PostGIS 2.0 in the brazilian federal forensics GIS", In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXIX-B4, 2012.

[4] Manual do PostGIS (2016). Disponível em: <http://postgis.net/docs/manual-2.0/>. Acesso em 3 de setembro de 2016.

[5] SRIDs (Spatial Reference Identifiers). Disponível em: <https://msdn.microsoft.com/pt-br/library/bb964707.aspx>. Acesso em 10 de novembro de 2016.

[6] Spherical Mercator (2016). Disponível em: <http://docs.openlayers.org/library/spherical_mercator.html>. Acesso em 12 de novembro de 2016.

[7] ST_GeomFromGeoJSON (2016). Disponível em: <http://www.postgis.org/docs/ST_GeomFromGeoJSON.html>. Acesso em 15 de novembro de 2016.

[8] Enabling Geospatial Business Intelligence (2009). Disponível em: <https://timreview.ca/article/289>. Acesso em 16 de novembro de 2016.

[9] ELMASRI, Ramez; NAVATHE, Shamkant. Sistemas de Bancos de Dados. 4. ed. São Paulo: Pearson, 2005.

Mapeamento de Padrões de Acidentes de Trânsito com Vítimas Fatais a partir de Dados Públicos do Governo do Estado do Rio Grande do Sul

Jorge Alberto F. Flores. Jr[1], Leonardo C. Steffenello[1], Ana T. Winck[1]

[1]Laboratório de Computação Aplicada (LaCa)
Universidade Federal de Santa Maria – Santa Maria – RS – Brazil

{jjunior,lsteffenello,ana}@inf.ufsm.br

Abstract. *Every year, statistics demonstrate an increasingly growth about fatality in Rio Grande do Sul traffic. This paper shows a study mapping traffic fatal accident patterns occurred in Rio Grande do Sul and in Santa Maria from 2007 to February 2016, in order to identify their main factors. To do so, we applied a Knowledge Discovery in Databases process, by means of the execution of Apriori and J48 algorithms in the WEKA tool. We employed a methodology with a extensive preprocessing, algorithm execution and a particular analysis of the generated rules for knowledge mapping about fatal traffic accidents in Rio Grande do Sul.*

Resumo. *A cada ano, estatísticas demonstram um aumento da fatalidade no trânsito do Rio Grande do Sul. Este trabalho apresenta um estudo para mapear os padrões de acidentes de trânsito com vítimas fatais ocorridos no Rio Grande do Sul e em Santa Maria, entre 2007 e fevereiro de 2016, a fim de identificar os principais fatores envolvidos. Para isso, é empregado um processo de Descoberta de Conhecimento em Bases de Dados, por meio da execução dos algoritmos Apriori e J48, na ferramenta de mineração de dados WEKA. É apresentada uma metodologia de extenso pré-processamento dos dados, execuções dos algoritmos e análises das regras geradas para um mapeamento dos conhecimentos adquiridos acerca dos acidentes de trânsito.*

1. Introdução

A Secretaria de Transportes e o Departamento Estadual de Trânsito do Rio Grande do Sul (DETRAN RS), dentre outras várias atribuições, atuam, em conjunto com outros órgãos, estabelecendo a política de transportes e fiscalizando o trânsito de veículos terrestres no Rio Grande do Sul (RS). Com o objetivo de informar e conscientizar a sociedade, o governo do estado do RS, através do Portal de Dados Abertos do RS (2016), disponibilizou uma base de dados, produzida pelo DETRAN RS, da relação de acidentes de trânsito com vítimas fatais no estado. Essa base de dados, denominada de Crimes de Trânsito, é constituída de 16.016 registros e 18 atributos, na qual cada registro representa um acidente de trânsito com vítima fatal ocorrido no estado entre o período de 2007 e fevereiro de 2016, e cada atributo indica uma característica envolvida nestes acidentes.

A W3C (2010) define dados abertos governamentais como os dados produzidos pelo governo e colocados à disposição das pessoas. Entretanto, um dos grandes desafios, em se tratando de dados abertos, é processar, analisar e tirar conclusões desse

imenso volume de dados brutos. Isso evidencia a necessidade de utilizar técnicas que facilitem a tarefa de extração de conhecimento, o que pode ser realizado por meio do processo de Descoberta de Conhecimento em Bases de Dados (KDD). Fayyad et al. (1996) definem o KDD como uma sequência de passos interativos e iterativos de apoio à tomada de decisão, fundamentais quando da existência de grandes volumes de dados e envolvendo as etapas de Seleção, Pré-Processamento, Transformação, Mineração de Dados e Interpretação. Diante da problemática do trânsito e da divulgação pública de seus dados, este trabalho tem como objetivo principal aplicar o processo de KDD a fim de identificar padrões sobre os dados referentes a acidentes de trânsito com vítimas fatais no RS, entre os anos de 2007 e fevereiro de 2016.

2. Metodologia

Para atender aos objetivos deste trabalho, foi aplicada uma sequência de passos fundamentais. Primeiramente, foi realizado o download do conjunto de dados Acidentes de Trânsito com Vítimas Fatais no RS. Os mesmos foram analisados, visando o entendimento dos registros e atributos. Após a seleção dos dados, foi possível aplicar as etapas de pré-processamento e transformação visando a adequação dos dados aos algoritmos a serem executados. Tratada a base de dados, aplicaram-se os algoritmos de mineração a fim de compreender os fatores que contribuem com a acidentalidade fatal no trânsito do RS. Para tanto, optou-se em utilizar regras de associação e árvores de decisão. Estas etapas foram repetidas, até que se chegasse a conhecimentos relevantes.

Neste trabalho, a etapa de Pré-Processamento se preocupou com que os resultados da mineração não fossem afetados pela grande quantidade de dados errôneos, irrelevantes e inconsistentes da base selecionada. Por isso, todos os seus dados e seus atributos tiveram de serem analisados e, quando necessário, aplicados a técnicas de eliminação e limpeza. Como consequência dessa etapa, foram criados novos atributos que possam auxiliar na descoberta de conhecimentos, excluídos atributos não relevantes ou simplesmente modificado a maneira como se apresentavam. Em relação aos dados, quando preenchidos de forma ruidosa, tiveram de serem adequados para se tornarem representativos. O pré-processamento fez com que os dados ruidosos fossem tratados de forma que não afetem os resultados da mineração.

Na execução da etapa de Mineração de Dados optou-se por utilizar as técnicas de regras de associação e árvores de decisão, por meio dos algoritmos Apriori (Agrawal et al., 1996) e J48 (Quinlan, 1993). Com a finalidade de encontrar resultados relevantes ao problema do tráfego, executaram-se testes visando à descoberta de padrões de acidentes em todo o RS e também apenas para Santa Maria. Para isso, foram executados diversos testes com diferentes configurações de parâmetros, através dos dois algoritmos, com diferentes valores de suporte e confiança e com e sem a presença de atributo classe.

Realizadas as execuções dos casos de testes, o último passo do processo de KDD envolveu a interpretação dos resultados gerados da aplicação dos algoritmos, a fim de transformá-los em conhecimentos úteis aos objetivos deste trabalho. A interpretação e avaliação dos resultados foram executadas por meio da análise das regras geradas pelo algoritmo Apriori e das árvores geradas pelo algoritmo J48, destacando os resultados mais relevantes e, principalmente, realizando uma comparação entre os dois algoritmos. Dessa forma, espera-se facilitar o entendimento sobre a acidentalidade fatal no estado.

3. Resultados

Com a execução dos algoritmos Apriori e J48 e pela interpretação de seus resultados, pode-se perceber que a descoberta dos melhores resultados depende muito do pré-processamento aplicado e da qualidade dos dados utilizados e que o número de regras geradas é sempre inversamente proporcional aos valores determinados para o suporte e a confiança. Além disso, devido à má distribuição dos registros entre as categorias, a maior parte das regras geradas continha sempre as mesmas características envolvidas. Entretanto, removendo-se da base de dados os atributos cujos registros foram considerados mal distribuídos, conseguiu-se chegar a resultados mais relevantes e que exploraram melhor os demais atributos. A aplicação da mineração com a presença de atributo classe também auxiliou na geração de regras específicas para os atributos.

Como foram gerados mais de 5000 resultados para regras de associação e árvores de decisão, não foi possível exibir todos os resultados encontrados. Nesse sentido, as regras produzidas pelos algoritmos são destacadas em termos da interpretação daquelas que melhor contribuíram com os objetivos deste trabalho. Dentre as mais relevantes, destacam-se:

a) Tanto para o RS, quanto para Santa Maria, há uma maior propensão à ocorrência de acidentes com vítimas fatais nos finais de semana e no período noturno;

b) No RS, apresenta-se uma maior propensão a acidentes em vias municipais. Caso se leve em comparação apenas as estradas rodoviárias, ocorre com maior frequência acidentes em vias estaduais do que em vias federais. Em Santa Maria, o panorama é diferente: apresenta-se uma maior propensão a acidentes em vias federais;

c) Para ambas regiões de estudo, há uma maior tendência à ocorrência de atropelamentos em vias municipais e de colisões em rodovias;

d) No RS, percebe-se uma maior propensão a acidentes na região metropolitana do estado. Quando analisado o atributo ‘Logradouro’ da base de dados do município de Santa Maria, percebe-se uma maior propensão a acidentes na BR-287;

e) Tanto para o RS quanto para Santa Maria, há uma maior tendência a colisões. Entretanto, se considerar apenas as vias municipais, há um predomínio de atropelamentos;

f) Os atropelamentos durante a madrugada, por ser uma ocorrência onde o pedestre é a principal vítima e onde existe um menor fluxo de pessoas no trânsito, aumentam a probabilidade de resultarem em óbitos;

g) Os acidentes considerados como homicídio doloso durante a madrugada tendem a ocasionar em fuga do local do crime por parte dos motoristas causadores;

h) Motoristas embriagados se envolvem principalmente em ocorrências de capotagens e choques a objetos fixos. Na maioria dos casos, os crimes são considerados como homicídio doloso, quando assume-se o risco de causar mortes; e

i) Motoristas sem CNH, independente do tipo do acidente, tendem a fugir do local do crime, e também causam crimes considerados como homicídio doloso.

Realizando uma análise comparativa entre os algoritmos utilizados, ambos se mostraram eficazes na descoberta de conhecimento. Interpretando as descobertas feitas neste trabalho podemos concluir que eles se complementam, e, apesar apresentarem os

resultados de maneira diferente, ambos os algoritmos podem ser utilizados em conjunto a fim de comprovar os resultados alcançados.

4. Conclusão

Para o desenvolvimento deste trabalho foi aplicado o processo de KDD sobre dados de ocorrências de acidentes de trânsito com vítimas fatais no RS, entre os anos de 2007 e fevereiro de 2016, a fim de analisar os padrões encontrados e realizar um mapeamento dos conhecimentos relevantes. Esse estudo ocorreu por meio da geração de regras de associação e árvores de decisão, utilizando os algoritmos Apriori e J48, na ferramenta de mineração de dados WEKA.

As técnicas e estratégias adotadas, bem como a conclusão sobre cada um dos experimentos realizados foram sendo registrados e documentados. Dessa forma, foi possível apresentar um mapeamento dos conhecimentos alcançados a fim de melhorar o entendimento sobre os resultados. Vale ressaltar a boa iniciativa do governo do estado, em conjunto com o DETRAN RS, de tornar estas informações públicas. Porém, pode-se destacar como ponto negativo, a qualidade de preenchimento dos seus dados, muitas vezes, com erros, faltantes e inconsistentes. Apesar disso, o processo de KDD foi realizado e atingiu os objetivos esperados com a execução deste trabalho.

Em suma, este trabalho visa se apresentar como uma contribuição na demonstração da viabilidade de utilização dos conceitos de mineração de dados a fim de analisar um problema tradicional. Além disso, espera-se que os resultados apresentados neste trabalho possam orientar novos estudos sobre o tema ou assuntos relacionados.

Referências

Agrawal, R. et al. (1993). A Mining Association Rules Between Sets of Items in Large Databases. In: ACM SIGMOD International Conference on Management of Data. 2nd edition

Fayyad, U. M. et al. (1996). Advances in Knowledge Discovery and Data Mining. In: Emerald Group Publishing. 1st edition.

Portal de Dados Abertos do Rio Grande do Sul. “Dados Abertos de seu Interesse”, http://dados.rs.gov.br, 18 abr. 2016.

Quinlan, J.R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

W3C. “Manual dos Dados Abertos: Governo do Rio Grande do Sul”, http://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual_Dados_Abertos_WEB.pdf, 18 abr. 2016.