28TH BRAZILIAN SYMPOSIUM ON DATABASES

10TH DEMOS AND APPLICATIONS SESSION

PROCEEDINGS

September 30th – October 3rd, 2013
Recife, Pernambuco, Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Pernambuco – UFPE

Realization

Centro de Informática (CIn)

Demos and Applications Session Chairs

Damires Souza, IFPB and Daniel Kaster, UEL

Editorial

The Brazilian Symposium on Databases (SBBD) is the official database event of the Brazilian Computer Society (SBC). It is the largest venue in Latin America for presentation and discussion of research results in the database domain. In its 28th edition, the symposium will be held in Recife, in the state of Pernambuco, on September 30 – October 03, 2013. SBBD brings together researchers, students and practitioners, from Brazil and abroad, to discuss problems related to the main topics in modern database technologies.

During SBBD, since 2004, a special session is dedicated to the presentation of database software aggregated with research solutions. This session, named Demonstrations and Applications Session (or Demos Session, for short), aims at revealing practical achievements among researchers, developers and professionals, from both academia and industry, interested in relevant and functional tools or applications that use data/information management technologies to solve non-trivial problems. The demonstrations held in the Demos Session concern relevant problems related to data management technologies, providing interaction between researchers and practitioners.

In 2013, the Demos Session reached its 10th edition. In this anniversary edition, we introduced a new presentation format: the Minute Madness. The Minute Madness is a dynamic event where each author has 1 minute to present his/her research. Authors are challenged to employ creative presentation skills to get the audience attention during this quick and fun moment and to motivate people to visit his/her demonstration/application. In this edition issue, we had 10 demo papers selected from a total of 21 submissions (an acceptance rate of 47%). Each paper was evaluated by 3 reviewers selected from a committee of 33 researchers from different Brazilian institutions. The Minute Madness got together demos and short papers. Then, during the live demonstration moments, the demos were assessed again by the award selection committee, which determined the Best Demo Paper Award. We thank all authors of submitted papers for their interest in the event. We also thank the reviewers and the award selection committee for their prompt and careful evaluations. We would like to express our gratitude for their hard work, which was fundamental for this event to happen.

We also congratulate the SBBD 2013 organizers for the local arrangements that provide the necessary infrastructure for Demos exposition and for the Minute Madness, as well as to SBBD Steering for the continuous support given to SBBD Demos.

We hope the SBBD 2013 Demos Session can awake great insights and many interaction opportunities for the Brazilian Database community.

Damires Souza
SBBD 2013, Demos and Applications Session Chair

Daniel Kaster
SBBD 2013, Demos and Applications Session Chair

28TH BRAZILIAN SYMPOSIUM ON DATABASES

September 30th – October 3rd, 2013

Recife, Pernambuco, Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Pernambuco – UFPE

Realization

Centro de Informática (CIn)

SBBD Steering Committee

Marco A. Casanova, PUC-Rio (Chair)
Ana Carolina Salgado, CIn-UFPE
Cristina Dutra de Aguiar Ciferri, USP
José Palazzo Moreira de Oliveira, UFRGS
Mirella M. Moro, DCC-UFMG

SBBD 2013 Committee

Steering Committee Chair
Marco A. Casanova, PUC-Rio
Local Organization Chair
Bernadette Farias Lóscio, CIn-UFPE
Program Committee Chair
Cristina Dutra de Aguiar Ciferri, USP
Short Papers Chairs
Renato Fileto, UFSC and Marco Cristo, UFAM
Demos and Applications Chairs
Damires Souza, IFPB and Daniel Kaster, UEL
Thesis and Dissertation Workshop Chairs
Karin Becker, UFRGS and André Santanchè, UNICAMP
Tutorials Chair
Mirella M. Moro, DCC-UFMG
Lectures Chair
João Eduardo Ferreira, IME-USP

Local Organization Committee

Bernadette Farias Lóscio, CIn-UFPE (Chair)
Ana Carolina Salgado, CIn-UFPE
Agenor Marinho de Souza Neto, CIn-UFPE

Carlos Eduardo Santos Pires, DSC-UFCG
Danusa Ribeiro Cunha, CIn-UFPE
Priscilla Vieira, CIn-UFPE
Robson Fidalgo, CIn-UFPE

Demos and Applications Program Committee

Altigran Silva, UFAM
Ana Paula Appel, IBM Research Brazil
Angelo Brayner, UNIFOR
Bernadette Lóscio, UFPE
Caetano Traina Jr., ICMC-USP
Carlos Eduardo Pires, UFCG
Carmem Hara, UFPR
Christian Bones, ICMC – USP
Damires Souza, IFPB
Daniel Kaster, UEL
Edleno de Moura, UFAM
Eduardo Speranza, UFSCar
Elaine Sousa, ICMC-USP
Evandro Baccarin, UEL
Fernanda Baião, UNIRIO
Fernando de Souza, UFPE
Fernando Lemos, Université de Versailles
Humberto Razente, UFU
Jonice Oliveira, UFRJ
Jose Maria Monteiro, UFC
Karin Becker, UFRGS
Livio Freire, UFC
Marcela Ribeiro, UFSCar
Marcio Oikawa, UFABC
Marcos Vieira, IBM Research Brazil
Maria Camila Barioni, UFU
Mirella M. Moro, UFMG
Renata Galante, UFRGS
Renato Bueno, UFSCar
Ronaldo Mello, UFSC
Sandra de Amo, UFU
Valeria Times, UFPE
Vânia Vidal, UFC

28TH BRAZILIAN SYMPOSIUM ON DATABASES

DEMOS AND APPLICATIONS SESSION

Table of Contents

NoSQL Data Model Evaluation on App Engine Datastore*

Felipe Ickert[1], Marcos Didonet Del Fabro[1], Eduardo Cunha de Almeida[1], Stefanie Scherzinger[2]

[1]Federal University of Paraná, Brazil (UFPR)
Curitiba – PR – Brazil

[2]Regensburg University of Applied Sciences
Germany

{fvickert,eduardo,didonet}@inf.ufpr.br

stefanie.scherzinger@hs-regensburg.de

Abstract. *Developing web applications with NoSQL databases as storage back-end can be rapidly done with existing application development frameworks. However, designing an unadapted NoSQL schema may cause serious impact on the application scalability. The impacts are often discovered only when the application is running and when the workload increases. In this article, we present a web-based tool with two different database schemas to show how the application scalability is impacted. The tool supports two different settings. First, concurrence-free write tests. Second, a simple benchmark to reproduce the behavior of concurrent writes in a cloud-based architecture. Our approach is presented in a demo app deployed in the cloud with two different NoSQL schemas revealing scalability bottlenecks.*

1. Introduction

With Platform-as-a-Service (PaaS) offering readily available infrasctructure, hosting an application in the cloud has became a commodity. A PaaS framework manages the complex deployment of the application across a large-scale hardware infrastructure, and comes with its own hosted databases, either relational or NoSQL.

NoSQL databases have as key aspect the simplicity of design, with less strict restrictions as we found in traditional DBMS (such as the ACID properties), with the main goal of achieving better scalability and availability. However, writing web applications using these databases is not always synonymous of scalability. Schema design made without care might cause severe performance bottlenecks when the workload increases. The bottlenecks may also affect availability of the application if the problem is discovered when it is running.

In this paper, we present a demo application in the cloud using a NoSQL database in order to emphasize that models done with poor design choices have impact in the performance of an application. We show how a naive schema choice can impact on the application.

This demo shows the difference and importance of the two schema designs of our case study revealing bottlenecks after some concurrency testing. Our demo is based on the

*A screencast of the tool is available at http://sbbd2013.cin.ufpe.br/screencasts.

work presented at [Scherzinger et al. 2013], where more details related to the approach can be found.

The remainder of this paper is structured as follows. In Section 2, we discuss a blogging application with alternative schema designs, as well as their impact on scalability. Our demo overview and implementation is presented in Section 3. Section 4 concludes with a summary and an outlook on future work.

2. Case Study

The core of our demo application consists of a blog application. While it may not be common, some very popular blogs may receive more than 4,500 comments over a single post. We implement two different NoSQL schemas as backend for this blog application. The first version is an article-oriented schema. The second version is na user-oriented schema. These designs use Datastore for persistence, a NoSQL backend provided with Google AppEngine [Google AppEngine 2013].

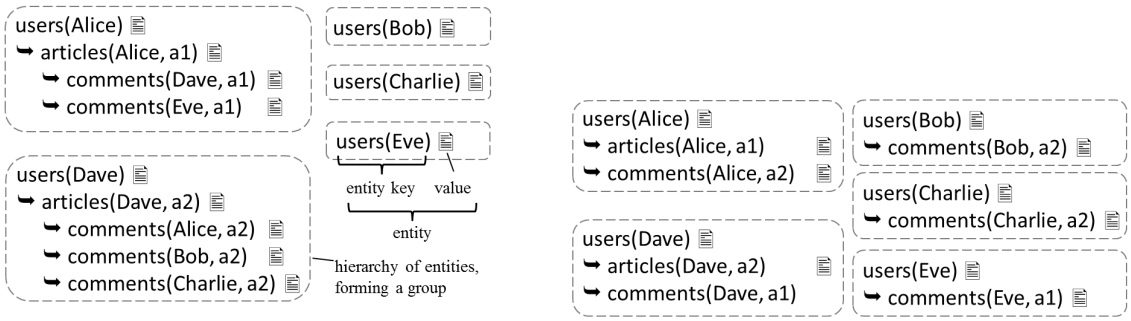
Datastore can be categorized as a key-value store[1] with entities as the basic unit of storage. Entities have a unique *key* and a *value*. Entities are made persistent by calling a simple *put*-operation, and retrieved by key using a *get*-operation. Different from a relational database, the schema is not fixed in advance. Entities need not to be consistent with any pre-defined structure. This makes Datastore a *schema-less* backend. It is nevertheless plausible to assume that developers maintain some loose notion of a schema, otherwise it becomes too complex to maintain an application. When the application logic requires strong consistency in updating several entities, the schemas and the corresponding data may be stored as an *entity group*. The system can then physically co-locate these entities, thus allowing for stricter consistency guarantees. Several systems built upon Megastore-like architectures provide this feature [Agrawal et al. 2012]. We describe the two different design below.

Naive design: The article-oriented schema. Figure 1(a) shows how entities could be grouped for a blogging application. We store an entity for each of the registered users Alice, Bob, Charlie, Dave, and Eve. Then "users(Alice)" is an entity key. The predicate "users" would then be the *entity kind*, a means of categorizing entities for the purpose of querying. As a key-value store, Datastore maps the key "users(Alice)" to the entity value containing data on Alice's user account, such as her password and preferences. In the figure, the value is depicted by a document icon, since we do not care for the exact contents.

Users may publish articles. "articles(Alice, a1)" is the key for an article written by Alice. The article entities may be stored with each user, forming a hierarchy of entities, and thus an entity group. Any comments on this article made by other users, such as a comment by user Dave with key "comments(Dave, a1)", are also stored within this group.

This may seem a natural model, since the conceptual relationships between entities are reflected in the hierarchy. However, by modeling the relationship between articles and comments in this manner, we force Datastore to co-locate all comments on a particular article in a single group. The write throughput for groups is throttled, causing writes

[1]Datastore is actually a *document store*, with structured values. Yet this distinction is of no consequence to our work.



(a) Grouping comments with articles. (b) Grouping postings with their authors.

Figure 1. Different blog schemas.

to fail if too many users are writing against a group simultaneously. With Google Datastore, at least one write can be handled against a group per second, while on average, 5–10 concurrent writes can be managed [Fuller and Wilder 2011]. Due to this limit and an optimistic concurrency control, concurrent writes against the same entity group are likely to fail.

A safer way: The user-oriented schema. The safest bet to avoid write contention is to never group any entities together. Yet storing each entity within its very own entity group comes at the cost of losing transactional safety and restricted query capabilities.

An approach that still allows for strong consistency on updates within a group and additional query options is to group all articles and comments with their authors, as depicted in Figure 1(b). Realistically, a user won't post several articles or comments per second. Hence, we consider it guaranteed that there will not be any concurrent writes against the same entity group. Thus, the alternative schema is safe from scalability bottlenecks.

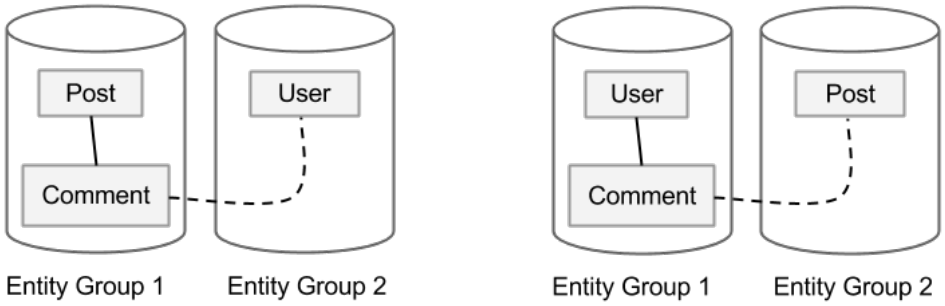
We present in the follwing sections the demo application that implements both design choices. More detailed results about the application performance can be found at [Scherzinger et al. 2013].

3. Blog Demo Overview

In order to build an application to map the two schemas presented in the case study and to show the bottleneck in an unsafe schema, we use three entities for the construction of two application blog schemas (*blog_app1 and blog_app2*). One entity represents the article (Entity Post), other represents the blog user (Entity User), and finally an entity represents a comment (Entity Comment) made by a user on an article.

We define as well the relationships between the entities in both schemas. We created two transaction classes representing the relationships between these three entities. First, one transaction class represents the post-oriented way, which is applied to *blog_app1* schema. Second, the user one is applied to *blog_app2*, considering that each transaction is guaranteed to be atomic and operates on entities in the same entity group or on entities in a maximum of five entity groups (cross-group transaction).

The first transaction class has the Entity Post as the parent of the Entity Comment,



(a) *blog_app1* (b) *blog_app2*

Figure 2. The alternative schema designs implemented.

which contains a reference to the User Entity. In this way we have a Entity Group with Entity Post plus Entity Comment, and another one with only the Entity User. In the second class, the Entity User is the parent of the Entity Comment with a reference to the Entity Post. In the same way, we have a Entity Group containing the Entity User plus Entity Comment and another Entity Group containing only the Entity Post. Each of these transactions classes represents the act of commenting an article. This characterizes the model as being scalability-safe or unsafe.

We see in Figure 2 that the entities are placed in different Entity Groups. This modeling difference is the main aspect that may impact in the application performance, since each entity group can be deployed in a different machine.

3.1. Implementation and Features

Our blog application is implemented in Java based on Google's AppEngine toolkit in the server side. It is publicly accessible at http://vanquish-ufpr-bottleneck.appspot.com and may be executed from any browser. All the requests in the client side are implemented using JQuery (Javascript framework) as well the benchmark part which uses Ajax calls to test the scalability of our different database schemas. Every time a client fires a blog writing request this task is processed by Java Servlets on the server side that communicate with the datastore, which tries to persist the data. If the datastore can not persist the data it raises a exception shown on client side.

The application consists of two main features: Single post or live test, where one can make a single comment on an article simulating a common blog. This feature is presented in two tabs, user-oriented and article-oriented, one per kind of schema, as shown in Figure 3. The other is the multiple post simulation (see Figure 4) where we can simulate a maximum of 999 comments in an article (user-oriented and article-oriented). The results of each transation is presented in real time.

We must first set up a number of users, which are created in the datastore, simulating the user subscriptions. After that the app provides two other inputs, one to include a number of comments on our first model (article-oriented) and another to include a number of comments on our second model (user-oriented). The application will perform

Figure 3. Blog live tests

asynchronous calls, respecting the users' number, to the Web Server, which will try to persist the comment. Every transaction is logged at the top of the page, Figure 5, as well a summary of these logs below the buttons where it is possible to check the bottleneck in the implementation of the unsafe data model.

In the article-oriented scenario, all comments on an article are stored in the same entity group as the article itself. If we run a series of 20, 100, 500, and 1000 write requests against one article, we get between 20% and 35% of write failures [Scherzinger et al. 2013].

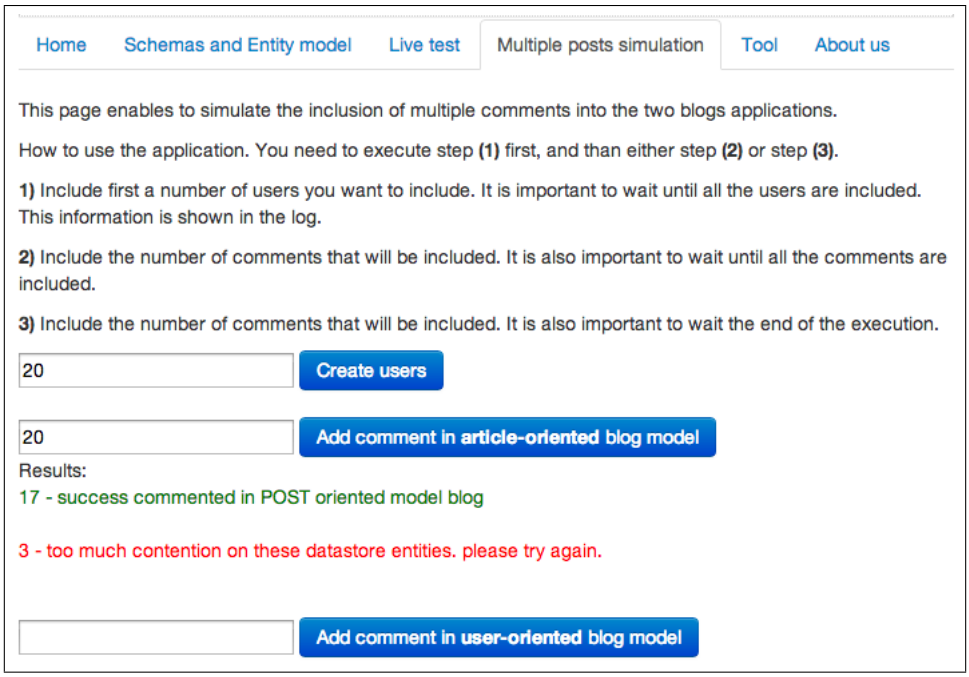


Figure 4. Multiple post simulation

4. Sumary and Outlook

In this paper, we have presented a demo application which implements two distinct blog schemas. We have shown how crucial is the design decision when developing NoSQL schemas for software-as-a-service applications. We have shown that the logical NoSQL schema has a vital impact on the overall scalability of an application. The schema is effectively responsible for physically balancing concurrent write requests. This relationship between schema and application performance is easy to miss for programmers who are new to this domain.

As future work, we plan to build a intelligent way that takes a NoSQL schema as input and that suggest others schemas which are more scalable in the cloud, or a way to help the developer to choose a safe schema before the code is deployed.

Acknowledgments. This work was partially funded by a CNPq grant.

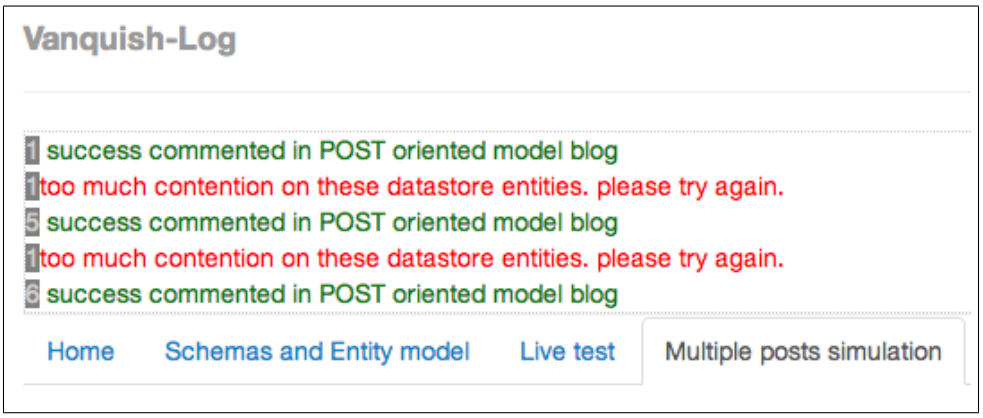


Figure 5. Multiple post simulation LOG

References

Agrawal, D., Das, S., and Abbadi, A. E. (2012). Data management in the cloud: Challenges and opportunities. *Synthesis Lectures on Data Management*, 4(6):1–138.

Fuller, A. and Wilder, M. (2011). "More 9s Please: Under The Covers of the High Replication Datastore". Google I/O Conference, presentation available on: `http://www.google.com/events/io/2011/`.

Google AppEngine (2013). Available on: `https://developers.google.com/appengine/`.

Scherzinger, S., De Almeida, E. C., Ickert, F., and Del Fabro, M. D. (2013). On the necessity of model checking nosql database schemas when building saas applications. In *Proceedings of the 2013 International Workshop on Testing the Cloud*, TTC 2013, pages 1–6, New York, NY, USA. ACM.

CloudSimDB: Um Simulador para o Provisionamento de Máquinas Virtuais para o Processamento de Aplicações Centradas em Banco de Dados*

Humberto Lima, Felipe Aragão, Jonas Lima,

Flávio R.C. Sousa, José Maria Monteiro

Departamento de Computação – Universidade Federal do Ceará (UFC)

Campus do Pici – Bloco 910 – 60.455-760 – Fortaleza – CE – Brazil

{hlima,faragao,jlima,flavio,monteiro}@lia.ufc.br

Abstract.*Cloud computing is a technology aimed at providing on-demand and pay-as-you-go IT services. Currently, many organizations are moving their data and applications to the cloud. In this scenario, given a workload, a SLA and a virtual machines configuration, the data service provider must make sure that it can run this workload respecting the parameters defined in the SLA. In this paper, we present the CloudSimDB, a simulator for resources provisioning towards database centric applications. It allows creating a virtually unlimited amount of simulation scenarios, allowing for each scenario, check the monetary cost to be paid for the use of cloud infrastructure, the expected time for completion of workload, and the SLA violation.*

Resumo. *A computação em nuvem é um paradigma que tem por objetivo prover serviços de TI sob demanda com pagamento baseado no uso. Várias organizações estão movendo seus dados e aplicações para a nuvem. Nesse contexto, dada uma carga de trabalho, um SLA e uma configuração de máquinas virtuais, é necessário verificar se é possível executar esta carga de trabalho no tempo definido o SLA. Neste trabalho, apresentamos o CloudSimDB, um simulador para provisionamento de recursos centrados em bancos de dados. A ferramenta permite criar uma quantidade virtualmente ilimitada de cenários de simulação e, para cada um, verificar o custo monetário da infraestrutura, o tempo esperado para a conclusão da carga de trabalho e a violação do SLA.*

1. Introdução

Computação em nuvem é uma tendência recente da tecnologia cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. A nuvem computacional é um modelo de computação em que dados, arquivos e aplicações residem em servidores físicos ou virtuais, acessíveis por meio de uma rede em qualquer dispositivo compatível (fixo ou móvel), e que podem ser acessados a qualquer instante, de qualquer lugar, sem a necessidade de instalação ou configuração de programas específicos [Buyyaet al. 2010].

* Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts. O código fonte pode ser obtido em https://github.com/jonaslimads/cloudsimdb.

Neste contexto, as aplicações voltadas ao gerenciamento de dados são candidatas potenciais para a implantação em nuvem. Isso ocorre porque, em geral: i) as instalações dos sistemas de banco de dados (SBDs) são complexas, ii) as bases de dados utilizadas envolvem um grande volume de dados e iii) as cargas de trabalho são sazonais, o que ocasiona elevados custos tanto em *hardware* quanto em *software*. Por esses motivos e pelas vantagens oferecidas pela computação em nuvem, como escalabilidade, elasticidade e custos proporcionais, os usuários estão movendo seus dados e aplicações para a nuvem com a finalidade de acessá-los de forma simples, barata, flexível e independente de localização [Sousa et al. 2010].

Nesse cenário, o serviço de gerenciamento de dados assume a responsabilidade pela instalação, configuração e manutenção do sistema de banco de dados (SBD). O provedor do serviço de dados é o responsável por fornecer o serviço, contabilizar a sua utilização, assegurar a sua qualidade e cobrar pelo seu uso. A qualidade do serviço é definida por meio de um acordo de nível de serviço (SLA). O SLA especifica o tempo máximo para execução de uma determinada consulta ou para uma carga de trabalho como um todo, a latência e a vazão esperada, além das penalidades a serem aplicadas ao provedor do serviço caso os níveis de qualidade acordados não sejam atingidos. Neste caso, o provedor deve ajustar a configuração de máquinas virtuais utilizadas com a finalidade de executar a carga de trabalho dentro dos parâmetros definidos no SLA [Sousa et al. 2011].

Utilizar servidores reais para avaliar e dimensionar a configuração de máquinas virtuais a ser utilizada pelo serviço de gerenciamento de dados apresenta inúmeras desvantagens, tais como o custo monetário elevado e a grande complexidade. Neste contexto, apresentamos o CloudSimDB, um simulador para o provisionamento de máquinas virtuais para o processamento de aplicações centradas em banco de dados. O CloudSimDB possibilita que usuários sem experiência de programação possam criar uma quantidade virtualmente ilimitada de cenários de simulação, compostos por uma carga de trabalho W, um SLA e uma configuração de máquinas virtuais, e, para cada um deles, exibe o custo da infraestrutura, o tempo de execução esperado para o processamento de W e a porcentagem de violação do SLA. Desta forma, o CloudSimDB pode ser utilizado para auxiliar os provedores de serviço de dados a maximizar seus lucros, por meio da redução dos recursos utilizados para executar as aplicações de seus clientes sem comprometer a qualidade dos serviços prestados, a qual é especificada no SLA.

2. Trabalhos Relacionados

Em [Calheiros et al. 2011], os autores apresentam o CloudSim, um *framework* de simulação para ambientes de computação em nuvem baseada em uma extensão do GridSim. Um simulador gráfico baseado no CloudSim e denominado CloudAnalyst é apresentado em [Wickremasinghe et al. 2010]. Em [Sá et al. 2011], os autores descrevem a criação de um conjunto de extensões para o *framework* de simulação CloudSim, visando a concepção de um simulador gráfico para ambientes computacionais distribuídos baseados no paradigma de computação em nuvem. Estes trabalhos simulam o ambiente de nuvem, mas estes não permitem simular a quantidade de recursos necessários para garantir a qualidade de um serviço, assim como aspectos de custos.

3. CloudSimDB

O CloudSimDB é uma extensão do *framework*CloudSim [Calheiros et al. 2011], adicionando características para simular a qualidade de serviço e o custo associado ao uso dos recursos. Estas características são essenciais para o ambiente de nuvem, principalmente para SDB, que utilizam uma grande quantidade de recursos. CloudSimDB foi desenvolvido utilizando a linguagem Java e a API JFreeChart[1] para gerar os gráficos que ilustram os resultados das simulações. No CloudSimDB, usuários sem experiência em programação podem configurar e realizar simulações por meio de uma interface gráfica simples e intuitiva. Os cenários de simulação construídos podem ser salvos e armazenados em disco, o que facilita a reprodução dos experimentos realizados e a comparação dos resultados obtidos.

Um cenário de simulação consiste basicamente em uma carga de trabalho W, representada por um conjunto de registros, um SLA e uma configuração de máquinas virtuais.Para um determinado cenário, o CloudSimDB permite simular o custo monetário a ser pago pela utilização da infraestrutura da nuvem, o tempo de resposta esperado para W, além do percentual de violação do SLA. As instâncias de máquinas virtuais disponíveis no CloudSimDB tiveram como parâmetros as características das instâncias oferecidas pela Amazon EC2[2] ("*SmallInstance*", "*LargeInstance*" e "*Extra LargeInstance*"). Assim, esses três tipos de máquinas virtuais podem ser utilizados no CloudSimDB. Esta escolha teve por objetivo aproximar o ambiente simulado dos ambientes reais. Neste trabalho foi utilizado o modelo de custo e de violação de SLA proposto por [Sousa et al. 2011].

4. Exemplos de Simulação

O CloudSimDB permite a realização de quatro tipos distintos de simulações:

- Uma única Máquina Virtual por Tipo
- Várias Máquinas Virtuais de um mesmo Tipo
- Várias Máquinas Virtuais de Tipo Diferentes
- Seleção Automática de uma Configuração de Máquinas Virtuais

4.1 Opção de Simulação 1: Uma Única Máquina Virtual por Tipo

Nesta primeira opção de simulação, o usuário deve fornecer como entrada: i) a carga de trabalho, isto é, a quantidade de registros do banco de dados ii) o SLA, ou seja, o tempo de resposta máximo para o processamento de todos os registros que compõem o banco de dados e iii) os tipos de máquinas virtuais que serão utilizados na simulação (SMALL, LARGE, EXTRA). Vale destacar que neste cenário será utilizada uma única instância de máquina virtual para cada tipo selecionado pelo usuário. Como resultado da simulação é possível observar: i) o custo monetário que o usuário teria que pagar para cada instância de máquina virtual (esse custo depende do tipo de MV e do tempo de utilização da instância, o qual depende do tempo necessário para que esta processe todos os registros do banco de dados) e ii) o percentual de violação do SLA.

[1]http://www.jfree.org/jfreechart/
[2]http://aws.amazon.com/ec2/

A Figura 1 ilustra um estudo de caso referente à primeira opção de simulação suportada pelo CloudSimDB, a qual utiliza uma única instância de máquina virtual por tipo de MV. Observe que o usuário forneceu como entrada cinco quantidade de registros (tamanhos do banco de dados) diferentes (10000000, 20000000, 30000000, 40000000 e 50000000), selecionou os três tipos de máquinas virtuais (neste caso, o cenário de simulação será formado por três instâncias de MV, sendo uma do tipo SMALL, uma do tipo LARGE e uma terceira do tipo EXTRA) e fixou o tempo de resposta máximo (SLA) para o processamento de todos os registros do banco de dados em 5 segundos.

Figura 1. Estudo de Caso 1: Uma única Máquina Virtual por Tipo

Os gráficos presentes na Figura 1 ilustram o custo das instâncias de máquinas virtuais utilizadas e o percentual de violação do SLA. Avaliando os resultados apresentados pelo gráficos de Violação do SLA, podemos observar que nenhuma das MVs viola o SLA para um banco de dados com até 10000000 registros. Portanto, neste caso, a melhor opção seria a utilização de uma máquina virtual do tipo SMALL, uma vez que esta oferece o menor custo por hora (US$ 0,085). Já para um banco de dados contendo 20000000 registros, podemos observar que a utilização de uma única instância de MV do tipo SMALL proporcionaria 50% de violação de SLA, ou seja, somente metade dos registros seriam processados em 2 segundos, e seu custo por hora seria de US$ 0,17. Por outro lado, a utilização de uma instância do tipo LARGE seria suficiente para atender ao SLA. Contudo, a utilização dessa instância geraria um custo por hora de US$ 0,34. Neste caso, o provedor do serviço de dados pode verificar se o custo de usar a instância SMALL mais a penalidade a ser paga pelo não cumprimento do SLA em 50% dos casos é menor que o custo de utilizar a instância do tipo LARGE. Neste caso, seria melhor utilizar a instância SMALL. Caso contrário, seria mais indicado utilizar a instância LARGE.

4.2 Opção de Simulação 2: Várias Máquinas Virtuais de um mesmo Tipo

Na segunda opção de simulação, o usuário deve fornecer como entrada: i) o tamanho do banco de dados, ii) o SLA,iii) o tipo de máquina virtual a ser utilizado na simulação (SMALL, LARGE ou EXTRA), iv) a quantidade inicial de instâncias de máquinas virtuais, v) a quantidade de máquinas virtuais que devem ser adicionadas a cada simulação e vi) a quantidade de simulações. Vale destacar que neste cenário são utilizadas diversas instâncias de máquinas virtuais, todas de um mesmo tipo. Como resultado da simulação é possível observar uma tabela para cada simulação, onde cada linha dessa tabela contém o identificador de uma das MVs utilizadas, o total de registros processados pela MV, a quantidade de registros processados dentro do SLA, a quantidade de registros processados fora do SLA e o tempo total de utilização da MV. Além disso, um gráfico em linha associa a quantidade de VMs com o percentual de violação do SLA.

4.3 Opção de Simulação 3: Várias Máquinas Virtuais de Tipo Diferentes

Nesta terceira opção de simulação, o usuário deve fornecer como entrada: i) o tamanho do banco de dados, ii) o SLA e iii) a quantidade de instâncias a ser utilizada na simulação para cada tipo de MV (SMALL, LARGE e EXTRA). Vale destacar que neste cenário são utilizadas diversas instâncias de máquinas virtuais, as quais podem ser de tipos distintos. Como resultado da simulação é possível observar uma tabela para cada tipo de MV, conforme mostra a Figura 2. Em uma dada tabela, referente à um determinado tipo de MV, cada linha dessa tabela contém o identificador de uma das MVs utilizadas, o seu tipo, o total de registros processados pela MV, a quantidade de registros processados dentro do SLA, a quantidade de registros processados fora do SLA e o tempo total de utilização da MV. Além disso, dois gráficos em barra são utilizados, um para ilustrar o custo monetário total para cada tipo de MV e outro para mostrar o percentual de violação do SLA para cada tipo de MV.

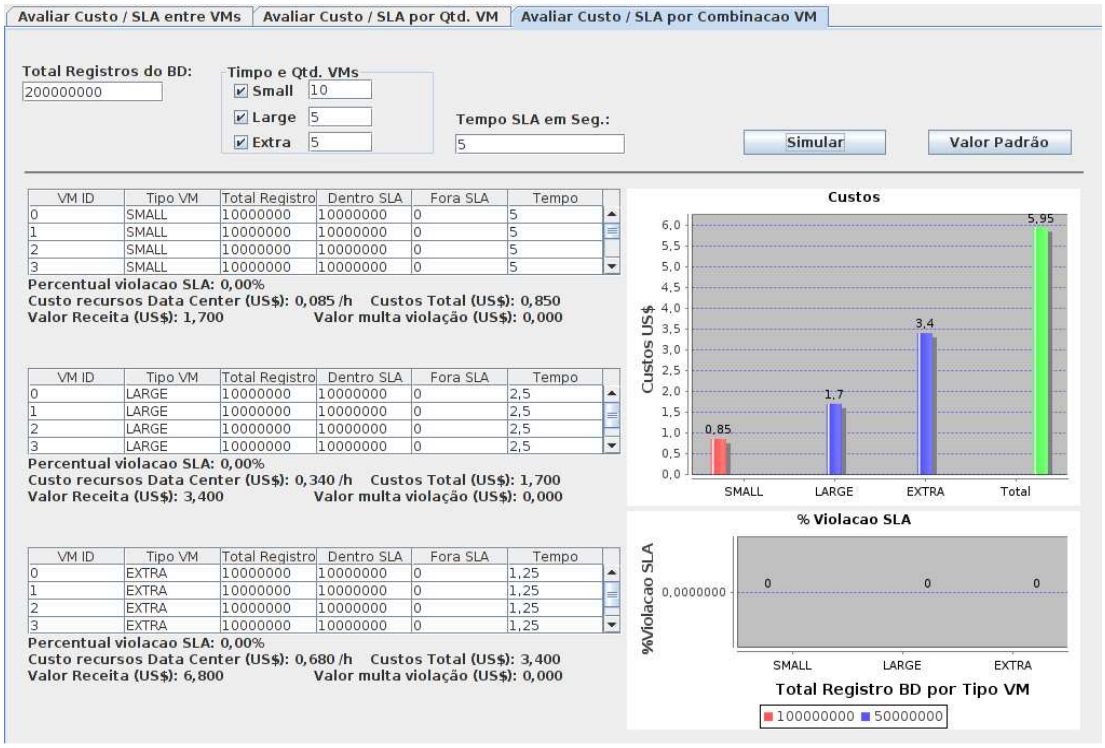


Figura 2. Estudo de Caso3: Várias Máquinas Virtuais de Tipo Diferentes

4.4. Opção de Simulação 4: Seleção Automática de uma Configuração de MVs

Naquarta opção de simulação, o usuário fornece apenas como entrada: i) o tamanho do banco de dados eii) o SLA. A saída será uma configuração de máquinas virtuais capaz de executar a carga de trabalho dentro dos limites especificados no SLA com o menor custo monetário possível, de forma similar a simulação 3.

5. Conclusões

Neste trabalho, apresentamos o CloudSimDB, um simulador para o provisionamento de recursos voltado para o processamento de aplicações centradas em bancos de dados. O CloudSimDB possibilita a definição e simulação de diversos cenários de execução, representando a infraestrutura de um ambiente de computação em nuvem. Assim, o CloudSimDB apresenta vantagens tanto para o provedor do serviço de dados quanto para as empresas contratantes deste serviço, tais como: i) um ambiente repetível e controlável para testes; ii) possibilita ajustes de gargalos no sistema antes de implantá-lo na nuvem real e iii) permite avaliar o desempenho e o custo de diferentes cenários de locação de recursos sob diferentes cargas de trabalho e distribuição de preços. Como trabalhos futuros, pretende-se adicionar heurísticas para melhorar a seleção automática da quantidade de máquinas necessárias para garantir a qualidade e implementar uma versão web do simulador.

Referências

RajkumarBuyya, Chee Shin Yeo, SrikumarVenugopal, James Broberg, and IvonaBrandic (2010). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, 2009.

Flávio R. C. Sousa, Leonardo O. Moreira, José Antônio Fernandes Macedo, andJavam C. Machado (2010). Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios. *Simpósio Brasileiro de Banco de Dados*, pages 101–130, 2010.

Flávio R. C. Sousa, Leonardo O. Moreira, andJavam C. Machado (2011). SLADB: Acordo de Nível de Serviço para Banco de Dados em Nuvem. In: *Simpósio Brasileiro de Banco de Dados*, pages 132–138, 2011.

Rodrigo N. Calheiros, RajivRanjan, Anton Beloglazov, Cesar A. F. De Rose, andRajkumarBuyya (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience (SPE)*, Volume 41, Number 1, pages 23-50, 2011.

Thiago T. Sá, José M. Soares, Danielo G. Gomes (2011). CloudReports: uma ferramenta gráfica para a simulação de ambientes computacionais em nuvem baseada no framework CloudSim. *IX Workshop em Clouds, Grids e Aplicações*, pages, 103-116, 2011.

Wickremasinghe, B., Calheiros, R. N., andBuyya, R. (2010). CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. *International Conference on Advanced Information Networking and Applications*, pages 446-452.

MyDBaaS: A Framework for Database-as-a-Service Monitoring*

David A. Abreu[1], **Flávio R. C. Sousa**[1]
José Antônio F. Macêdo[1], **Francisco J. L. Magalhães**[1]

[1]Department of Computer Science
Federal University of Ceará (UFC)
Pici - Fortaleza - CE - Brazil

{araujodavid, flavio, jose.macedo, franzejr}@lia.ufc.br

Abstract. *Environments of database-as-a-service (DBaaS) promise high performance, high availability, and elastic scalability. These characteristics created the need for innovative monitoring approaches. There are many tools/solutions available for monitoring cloud services. However, there is no standard model for monitoring of databases-as-a-service. In this paper we demonstrate a powerful and easy-to-use framework that provides monitoring for environments of DBaaS, called MyDBaaS Framework. We also present MyDBaaSMonitor application that implements the framework and a demonstration of how to consume the collected metrics by the monitoring environment created.*

1. Introduction

Cloud computing is an extremely successful paradigm of service-oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Scalability, elasticity, pay-per-use pricing, and economies of scale are the major reasons for the successful and widespread adoption of cloud infrastructures. Since the majority of cloud applications are data-driven, database management systems (DBMSs) powering these applications are critical components in the cloud software stack [Elmore et al. 2011].

With the advent of hosted cloud computing and storage, the opportunity to offer a DBMS as an outsourced service is gaining momentum, as witnessed by Amazon's RDS and Microsoft's SQL Azure. Such a DBaaS is attractive for two reasons. First, due to economies of scale, the hardware and energy costs incurred by users are likely to be much lower when they are paying for a share of a service rather than running everything themselves. Second, the costs incurred in a well-designed DBaaS will be proportional to actual usage ("pay-per-use") - this applies to both software licensing and administrative costs [Curino et al. 2011b].

When using DBaaS it is necessary to monitoring the services to verify the Service-Level Agreements (SLA) and costs. According to [Aceto et al. 2012], cloud computing involves many activities for which monitoring is an essential task. The most important ones are: i) Capacity and Resource Planning, ii) Capacity and Resource Management, iii) Data Center Management, iv) SLA Management, v) Billing, vi) Troubleshooting and vii) Security Management.

*A screencast of the tool is available at http://sbbd2013.cin.ufpe.br/screencasts.

There is a large number of works proposing solutions to monitor the cloud [Aceto et al. 2012] [NewRelic 2013]. Nevertheless, these works focus on resources on low-level computing resources (e.g., CPU speed, CPU utilization, disk speed). Furthermore, these works do not use specific metrics for monitoring DBMSs and they are not extensible. According to our research, there are no solutions that address cloud database monitoring, since previous works have focused only on some of its aspects.

In order to solve this problem, this paper proposes MyDBaaS[1], a Java-based framework that provides a comprehensive programming and configuration model for developing monitoring strategies for environments of DBaaS. The MyDBaaS Framework offers set of hook points like classes and services already implemented that make its use easy and transparent. It also comprises hotspots which make the structure flexible and extendable as required, as well as a set of metrics already defined. According to the known literature, this is the first framework that exploits the monitoring for environments of DBaaS, representing a significant contribution to the area. The paper also presents MyDBaaSMonitor, a web application implemented with MyDBaaS Framework and a demonstration of the use of MyDBaaS API.

This paper is organized as follows. Section 2 explains the MyDBaaS Framework. Section 3 illustrates the applications that we will demonstrate. Section 4 details related work and, finally, Section 5 presents the conclusions.

2. MyDBaaS Framework

The framework focuses on the creation of the monitoring of the resources allocated for the operation of DBaaS environments. Monitoring can be performed in a scenario of up to four levels: physical infrastructure (hosts), virtual machines (VM), DBMSs and database instances. The MyDBaaS enables the monitoring of metrics at all levels according to the need. It has a complete structure that provides from creating a metric until storage in the knowledge base. The MyDBaaS provides a pattern programming through nomenclatures, hook points and hotspots that make its use simple and intuitive. The communication between the server and monitoring agents is completely transparent to the user, as well as the creation and maintenance of knowledge base. The knowledge base is important because it creates a profile of the use of all the metrics monitored according to the time.

2.1. Architecture

The MyDBaaS Framework architecture is divided into four modules: *Core*, *Agent*, *API* and *Common*. Each module is composed by a set of components, as illustrated in Figure 1. The *Core* is the central module responsible for coordinating the resources of the environment, coordinate the monitoring agents and maintain the knowledge base. The **Controller** component provides the structure for the management of requests sent by the API and monitoring agents. The **MonitoringCoordinator** is the component whose function is to access the resources and set up the monitoring agents automatically. **Repository** is a major component, providing transparent and automatic features for creating and maintaining the knowledge base and for the storage of collected metrics and resources. The **Knowledge Base** has the function of storing the information generated by the framework creating a behavioral profile of resources as a function of time. The *Agent* is the module present in the resources of the environment and is responsible for to monitor,

[1] http://www.mydbaasmonitor.com

collect and interact with the *Core* module, and consists of four components. The **Monitor** provides functions to automatically load the agent settings, as well as methods that give the agent the ability to self-configure. The **Collector** is the component that provides the infrastructure to collect metrics. It enables the metrics to be collected independently and on different cycles. It also provides dynamic and automatic methods for sending the metrics to the *Core* module. The **MetricManager** is responsible for loading the settings of the metrics that the agent will use. The **ConnectionManager** is responsible to provide features of access to databases and DBMS automatically when a collector needs to measure a metric. The *API* is the module that allows external access to the monitoring environment. It provides a set of routines and patterns for the use of *Core* services. The **Client** is the component responsible for communication between API and Core, supplying the features of connection management to the other components. The **PoolManager** provides access to registered resources in the monitoring environment, allowing retrieve, manage and add new resources. **QueryMetric** is the main component of this module. It enables the consumption of collected metrics about resources through diverse methods. The **AccessCoordinator** allows the management of monitoring agents. The last module is the *Common*, which it has two components. **ResourceBucket** and **MetricBucket** are responsible for providing the pattern for creating the representation of new resources and metrics respectively.

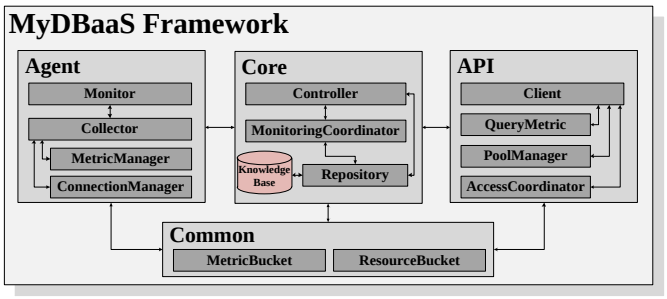


Figure 1. Overview of the MyDBaaS Framework architecture.

MyDBaaS was fully implemented in Java based on VRaptor Framework and Apache HttpComponents. Agent-side and API-side actions are processed by VRaptor on the Core-side, and the communication is done via HTTP requests. MyDBaaS's architecture is designed to easily include new metrics, as well as it was developed to be independent of the infrastructure and DBMS. The HTTP protocol was chosen because it makes easier the scalability of the architecture. The knowledge base stores a large volume of monitored data into a relational DBMS, but can be easily extended to a NoSQL (e.g., MongoDB[2] or Cassandra[3]). The framework already has some metrics implemented in the four levels. In the level of VMs: CPU, memory, network, disk, partitions and information about the VM settings. In the level of hosts: HostInfo that allows to collect the physical configuration information and the hypervisor used, HostDomains that allows to collect the amount of active and inactive VMs hosted, DomainStatus that allows to collect the use of CPU and memory of the VMs on the host side and all of the previous level. In the level of DBMS: CPU and memory used, active connections and base size. In the level of database instances are the same of the DBMS except CPU and memory.

[2]http://www.mongodb.org
[3]http://cassandra.apache.org

2.2. API

The MyDBaaS application programming interface (API) enables developers to build applications that interact directly with the monitoring environment created by MyDBaaS Framework. With these applications, users can more efficiently and creatively manage their resources and services. This API can be used to develop additional tools and applications to help the manage the service level objectives (SLOs) of the SLAs, integrate the collected data with systems of data analysis to help making decisions or applications that generate events and alerts about monitored resources. The MyDBaaS API gives access to all of the metrics data that are being monitored in real time, as well as the historical of all that has been collected. It also allows to access and manage the resources registered in the environment.

```
   MyDBaaSClient client = new MyDBaaSClient("http://localhost:8080/mydbaasmonitor");
(1) DBMS dbms = (DBMS) client.resourceLookupByID(1, "dbms");
   VirtualMachine virtualMachine = (VirtualMachine) client.resourceLookupByID(2, "machine");

(2) MyMetrics myMetrics = new MyMetrics(client);
   List<Object> memories = myMetrics.getMetricCollection(Memory.class, virtualMachine, "01-06-2013", "30-06-2013");

(3) Size size = (Size) myMetrics.getMetricSingle(Size.class, dbms, null, null);
```

Figure 2. Example of using MyDBaaS API.

Figure 2 illustrates some available functionalities by the API. The example (1) shows the creation of the connection to the server that contains the *Core* module. Also presents how it can be done to recover the record of different resources. The (2) demonstrates one way to consume the historical of the metric of Memory in a given time range about a particular virtual machine. The last example (3) presents how to retrieve the latest collection of the metric of Size about a particular DBMS.

3. Demonstration

To showcase the benefits of the described framework we propose two demonstrations of the main features provided by the MyDBaaS Framework. As a running and implementing example we show MyDBaaSMonitor, a web application for managing the monitoring of a given scenario. Finally, we demonstrate one way of using the MyDBaaS API to consume the metrics monitored in this application.

MyDBaaSMonitor is a web application developed using MyDBaaS Framework as backend and Twitter Bootstrap with Highcharts to design the graphical user interface. This application enables the monitoring of the following scenario: a cloud managed by OpenNebula, using the KVM as virtualization solution. This cloud is composed of a set of servers hosting multiple virtual machines, and each one can have multiple DBMSs. And each DBMS can have multiple instances of databases. Finally, this cloud can contain several environments of DBaaS. The application has a web interface for the user to manage the environments, allowing to configure and register the resources composing each DBaaS. Once the resources have been registered, the user can configure monitoring for each resource of the environments as needed. The user chooses which metrics will be collected, as well as the monitoring cycle for each metric. Finally, the application accesses the resource automatically and sends the monitoring agent and its configuration file, installs and starts the agent automatically and transparently. The user can track the

real-time monitoring through the graphics that are displayed when accessing the resource record. The user can also stop monitoring or reconfigure the agent as needed.

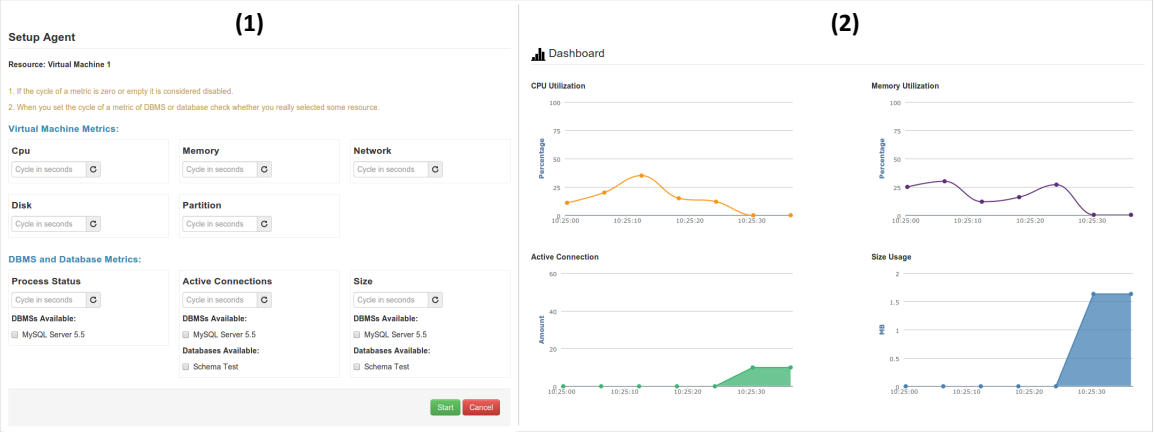


Figure 3. Overview of two application screens.

Figure 3 shows two screens available in the application. The screen (1) is where the user sets the monitoring that will run on a given virtual machine. If there are DBMSs and database instances within the VM it is also possible to configure the monitoring. On the screen (2) is where the user can view the metrics that are being monitored on a particular DBMS, and manage their information.

How to use MyDBaaS API - during this demonstration we will present how the API can be used to build applications that need to use monitoring. In a first scenario we show how to consume the metrics in different ways and at all levels of resources. To this end we employ our local cloud at Federal University of Ceará, which will have a running instance of the MyDBaaSMonitor application. And we will use the metrics collected in this scenario. In a second scenario we show an application to create alerts based on monitoring performed in the first scenario.

4. Related Work

In the last years, several systems have been developed in order to monitor cloud services. However, studies in the literature point to works and products that focus on specific points of monitoring. In [Curino et al. 2011a] the authors present Kairos, a system for database consolidation in VM-based environments. It uses monitoring on OS and DBMS about CPU, memory, disk, buffer pool utilization and log flushes. Using these metrics to estimate resource needs during the consolidation process. [Klems et al. 2012] shows a runtime quality measurement framework for cloud database service systems. This framework uses a monitoring on quality metrics for this service (eg, average throughput, average and upper-percentile response time, elastic scalability, and others). These metrics are composed of smaller granularity metrics. The authors of [Zhao et al. 2012] present a solution for database replication in virtualized environment of cloud. They also needed to monitor metrics on two levels, virtual machine (CPU, disk and network) and DBMS (throughput). Therefore as previous works showed, monitoring is essential for strategies used to provide an environment of DBaaS. Unlike these works which do not have the monitoring as the main problem, our work focuses on creating monitoring solutions according to the

need for this type of environment. Our proposed framework can be used by the previous works, supplying the need for monitoring of them at all levels. Some existing solutions in the market, such as [NewRelic 2013] and [Amazon 2013], provide monitoring services ready for the end user, different from our work that enables the user to create their own monitoring environment/service in several layers. The focus of our work was to design a framework, so we opted not to extend the current solutions. Based on our review of the literature, there is none framework for monitoring database-as-a-service that enables the creation of a range of metrics ranging from physical infrastructure until the instances of databases. The MyDBaaS Framework differs by the fact to allow extent and be flexible to adapt to the real needs of the user's monitoring. Importantly, our solution is fully open source.

5. Conclusion and Future Works

This paper presented the MyDBaaS Framework that enables the creation of monitoring solutions for environments of DBaaS. Based on an architecture divided into modules, the framework can be extended by adding new features and metrics in a practical way. The paper also presented the MyDBaaSMonitor, a web application developed using the framework, having presented interesting initial results about its use. More metrics for databases (e.g., throughput, response time, latency, buffer pool size, cache hit rate, SQL commands, transactions committed and rolled back, freeable memory, network traffic, number of reads and writes, number of bytes read and written, and others) are being developed and will be added to the framework. As future work two modules will be created. The **Driver** that allows to create and collect metrics at the level of workloads sent to the DBMSs and database instances, and **ExporterMetric** that allows the extraction of metrics in CSV file in a personalized way.

References

Aceto, G., Botta, A., de Donato, W., and Pescape, A. (2012). Cloud Monitoring: Definitions, Issues and Future directions. In *CLOUDNET '12*, pages 63–67.

Amazon (2013). Amazon Cloudwatch. `http://aws.amazon.com/cloudwatch`.

Curino, C., Jones, E. P., Madden, S., and Balakrishnan, H. (2011a). Workload-aware database monitoring and consolidation. SIGMOD '11, pages 313–324.

Curino, C., Jones, E. P. C., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011b). Relational Cloud: a Database Service for the Cloud. CIDR '11, pages 235–240.

Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. (2011). Zephyr: live migration in shared nothing databases for elastic cloud platforms. In *SIGMOD '11*, pages 301–312.

Klems, M., Bermbach, D., and Weinert, R. (2012). A Runtime Quality Measurement Framework for Cloud Database Service Systems. QUATIC '11, pages 38–46. IEEE Computer Society.

NewRelic (2013). New Relic Application. `http://newrelic.com`.

Zhao, L., Sakr, S., Fekete, K., Wada, H., and Liu, A. (2012). Application-Managed Database Replication on Virtualized Cloud Environments. ICDEW '12, pages 127–134.

Uma Ferramenta para a Sintonia de Instruções SQL*

Arlino H. M. de Araújo[1,2], José Maria Monteiro[2], José A. F. de Macêdo[2], Júlio A. Tavares[3], Ângelo Brayner[3]

[1]Universidade Federal do Piauí (UFPI) – Picos – PI – Brasil

[2]Universidade Federal do Ceará (UFC) – Fortaleza – CE – Brasil

[3]Universidade de Fortaleza – Fortaleza – CE – Brasil

{arlino,monteiro,jose.macedo}@lia.ufc.br, {julio,brayner}@unifor.br

Abstract. *The cost models used in current DBMS query optimizers depend on several factors, such as: statistics, estimates and properties. This fact does these optimizers, in many cases, inaccurate to pick plans, generating sub-optimal plans. In such situations, it is necessary to tune SQL statements to guide the optimizer in choosing a better execution plan than that which would normally be selected. However, this process requires a high level of expertise and skill of the database administrator. Thus, this paper presents a tool in order to support SQL statements tuning. The proposed tool uses two different approaches to support SQL tuning: assisted and automatic.*

Resumo. *Os modelos de custo utilizados pelos otimizadores de consulta dos SGBDs atuais dependem de diversos fatores que podem torná-los imprecisos, tais como: estatísticas, estimativas e propriedades. Este fato faz com que esses otimizadores sejam, em muitos casos, imprecisos na escolha de planos de execução, gerando planos sub-ótimos. Em tais situações, é necessário realizar a sintonia da instrução SQL a fim de orientar o otimizador na escolha de um plano de execução melhor do que aquele que normalmente seria selecionado. Entretanto, esse processo exige um elevado nível de especialização do administrador do banco de dados. Neste sentido, este trabalho apresenta uma ferramenta, denominada IQT (Interactive Query Tuning) , que fornece suporte para o processo de sintonia de instruções SQL. O IQT possui dois modos de funcionamento distintos: assistido e automático.*

1. Introdução

Os modelos de custo utilizados pelos otimizadores de consulta atuais são bastante complexos e dependem de fatores que podem torná-los imprecisos [Bruno and Chaudhuri and Ramamurthy 2009]. Por este motivo, mesmo usando métodos de acesso e estratégias de avaliação suportadas pelo SGBD, às vezes, os otimizadores não conseguem produzir planos ótimos. Nestes casos, deve ser realizada a sintonia da instrução SQL. Para isso, em geral, duas estratégias são frequentemente utilizadas: (a) reescrever a cláusula SQL; e (b) aplicar *Query Hinting* [Shasha and Bonnet 2003].

A técnica de reescrita consiste em escrever uma nova instrução SQL equivalente à instrução SQL inicialmente utilizada (ou seja, que retorne o mesmo resultado) e que apresente ganhos de desempenho [Hamakrishnan and Gehrke 2008].

* Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

Um mecanismo comum usado nos bancos de dados comerciais é denominado *Query Hinting*. Um *hint* (dica) instrui o otimizador a restringir seu espaço de busca para certo subconjunto de planos de execução (por exemplo, impondo a escolha de planos que usem um determinado tipo índice ou determinando a ordem e/ou método de junção).

O Administrador de Bancos de Dados (DBA) utiliza-se da reescrita de SQL e da inserção de *hints* para ajustar as cláusulas SQL. Entretanto, esse processo é complexo e requer conhecimentos em diferentes áreas [Herodotou and Babu 2009].

2. Solução Proposta

Este trabalho apresenta uma ferramenta, denominada IQT (*Interactive Query Tuning*), para auxiliar o processo de sintonia de instruções SQL. A ferramenta proposta utiliza duas abordagens distintas para a sintonia de instruções SQL: uma abordagem assistida e outra automática. Em ambas as abordagens um conjunto de 11 heurísticas é utilizado para realizar a sintonia (reescrita) das instruções SQL. As heurísticas são constituídas de regras para identificar oportunidades de sintonia nas instruções SQL.

A Tabela I ilustra as onze heurísticas para reescrita de instruções SQL utilizadas pela ferramenta IQT. Além disso, indica para cada heurística se esta é ou não atualmente implementada nos três principais SGBDs comercias: PostgreSQL 8.3, Oracle 11g e SQL Server 2008.

2.1. Abordagem Assistida

A abordagem assistida consiste em um *advisor* capaz de: (i) capturar as instruções SQL anteriormente executadas, (ii) analisar essas instruções e (iii) sugerir (por meio de *alertas*, *wizards* ou relatórios) oportunidades de sintonia. Assim, o *advisor* identifica instruções SQL que se fossem reescritas poderiam fazer com que o otimizador de consultas escolhesse planos de execução melhores, reduzindo o tempo de execução das instruções SQL. Adicionalmente, permite ao DBA interagir com o processo de sintonia, por exemplo, selecionando um subconjunto das heurísticas disponibilizadas a fim de que somente essas sejam utilizadas para sintonizar as cláusulas SQL em geral ou um determinado comando SQL em particular. Se o DBA achar que algumas heurísticas são desnecessárias ou inadequadas para o seu banco ou para uma determinada consulta ele pode desativá-las.

No modo assistido, o *advisor*, periodicamente, irá analisar as instruções SQL previamente capturadas. Caso o *advisor* consiga reescrever alguma das instruções SQL analisadas, este gera um relatório contendo as recomendações de reescritas que foram aplicadas em cada uma das instruções ajustadas. Opcionalmente, o DBA pode visualizar também a lista de instruções SQL coletadas pelo *advisor*. Adicionalmente, o DBA pode fornecer um arquivo contendo um conjunto de instruções SQL a fim de que o processo de sintonia seja aplicado às instruções presentes nesse arquivo.

O DBA pode interagir com as sugestões de sintonia geradas pelo IQT. Para isso, ele deve selecionar uma das instruções SQL capturadas automaticamente pelo *advisor* ou fornecidas via arquivo e iniciar o processo de sintonia. Após a conclusão desse processo, caso alguma das heurísticas tenha sido aplicada com sucesso, a instrução SQL reescrita será exibida. O DBA pode ainda comparar a instrução SQL

original e a instrução SQL reescrita, visualizando, por exemplo, os planos de execução e os tempos de resposta gerados para cada uma delas, bem como o tempo gasto no processo de reescrita. O IQT exibe também uma descrição de todas as transformações sofridas pela instrução SQL original durante o processo de sintonia, até a geração da consulta reescrita final.

A ferramenta IQT permite ao DBA escolher e definir quais heurísticas devem ser utilizadas no processo de reescrita de uma determinada instrução SQL. Para isso, o DBA deve selecionar a instrução SQL ou digitar a instrução desejada. Em seguida, o DBA deve selecionar as heurísticas que deseja aplicar durante o processo de sintonia da instrução SQL selecionada. Além disso, o DBA pode definir um subconjunto de heurísticas a serem aplicadas às instruções SQL em geral, ou seja, para as instruções SQL que não possuam definições específicas. Vale destacar que tanto as configurações definidas para uma instrução SQL em particular quanto para as instruções SQL em geral podem ser utilizadas pelo *Middleware* que compõe a abordagem automática.

2.2. Abordagem Automática

A abordagem automática consiste em um *middleware* que atua entre a aplicação e o SGBD. Este *middleware* é responsável por: (i) receber as cláusulas SQL enviadas pelas aplicações; (ii) analisar e ajustar (reescrever) as cláusulas SQL recebidas (se necessário); (iii) enviar as cláusulas(reescritas ou não) para o SGBD e (iv) receber do SGBD o resultado da execução do comando SQL e enviá-lo à aplicação.

O middleware desenvolvido consiste em uma classe Java que pode ser utilizada pelos desenvolvedores de aplicação a fim de que uma determina instrução SQL seja reescrita antes de ser enviada ao SGBD. Assim, os desenvolvedores não mais utilizariam a API JDBC (pacote java.sql) diretamente, e sim por meio do *middleware*. Assim, antes de executar uma determinada instrução SQL, o *middleware* verifica se a instrução possui oportunidades de sintonia e a reescreve (se necessário). Após esta etapa, a instrução SQL (reescrita ou original) é enviada ao SGBD. Após executar a instrução SQL, os resultados são retornados para a aplicação. Opcionalmente, o *middleware* pode checar se o custo de execução da instrução SQL original é menor que o da instrução reescrita, com a finalidade de garantir que o tempo de execução da instrução reescrita seja menor que o da original.

As abordagens assistida e automática apresentam ainda as seguintes características:

Não-intrusiva: completamente desacoplada do código do SGBD. Isso permite que a solução concebida possa ser utilizada com qualquer SGBD. Além disso, a solução não está sujeita a modificações em seu código a cada nova versão do SGBD.

Independente de localização: pode executar em uma máquina distinta daquela utilizada para hospedar o SGBD, não consumindo recursos do servidor onde o SGBD está hospedado.

3. Trabalhos Relacionados

Em [Herodotou and Babu 2009] é apresentada uma ferramenta denominada *zTuned* cuja finalidade é facilitar experimentos relacionados à sintonia de consultas SQL. A ferramenta produz conjuntos de planos que possuem operadores com a

mesma cardinalidade (chamados de planos da vizinhança) e escolhe o plano ótimo entre os melhores planos de cada vizinhança, utilizando-se do mecanismo de estimativa de custos do próprio SGBD. No Oracle 10g é introduzido um componente denominado *Automatic SQL Tuning Advisor* que fornece recomendações de oportunidades de reescrita de consultas utilizando equivalências semânticas [Dageville and Dias 2006].

Existem algumas ferramentas tais como o *IBM Optim Development Studio* [Studio 2010], o *Embarcadero DB Optimizer XE* [Optimizer 2010] e o *Quest SQL Optimizer for Oracle* [Oracle 2010] que já realizam sintonia de consultas através de recomendações de reescrita, de uso de *hints* e/ou de criação de índices. Entretanto, elas adotam uma abordagem *offline* na solução do problema e transferem para o DBA a tarefa de capturar e fornecer a carga de trabalho, dentre outras.

4. Resultados Experimentais

A fim de avaliar a eficácia da ferramenta IQT utilizando as abordagens assistida e automática investigamos dois cenários distintos. No primeiro cenário utilizamos o *benchmark* TPC-H, o qual consiste um *benchmark* voltado para aplicações de suporte à decisão. O TPC-H possui um conjunto de 23 consultas *ad-hoc*. No segundo cenário utilizamos a base de dados do *benchmark* TPC-H e uma carga de trabalho sintética formada por 30 consultas (com problemas de sintonia).

Para cada cenário, três testes foram executados: i) executou-se a carga de trabalho contendo as consultas originais (sem reescrita), o que será utilizado como *baseline*; ii) a carga de trabalho original (sem reescrita) foi submetida ao *advisor* (abordagem assistida) e, em seguida, a nova carga de trabalho (contendo as consultas ajustadas) foi executada; e iii) cada consulta da carga de trabalho original foi enviada para o *middleware* (abordagem automática com todas as 11 heurísticas habilitadas), o qual procedeu à reescrita da consulta (quando necessário) e, em seguida, executou a consulta reescrita. Para cada teste executamos 1, 2, 4, 8, 16 e 32 iterações da carga de trabalho (ou seja, 1, 2, 4, 8, 16 e 32 execuções das 23 consultas do TPC-H).

Em cada teste, as cargas de trabalho foram submetidas de três formas distintas: sequencial, aleatória, aleatória fixa (onde define-se uma sequencia aleatória e depois mantem-se essa sequencia). Devido às limitações de espaço iremos discutir somente os resultados dos testes sequenciais. O ambiente de execução foi composto por uma estação Core i3-2100 3.10GHz, com 4GB de RAM e 500 GB de HD. Os SGBDs utilizados foram PostgreSQL 8.3, Oracle 11g e SQL Server 2008.

4.1. Cenário 1: Benchmark TPC-H

Das 23 consultas que compõem o *benchmark* TPC-H, duas (consultas 18 e 20) foram reescritas (sintonizadas) pelo IQT, por meio da heurística H8, para os SGBDs SQL Server e PostgreSQL. Observando as Figuras 1 e 2 podemos concluir que a abordagem assistida apresentou uma pequena diminuição no tempo de execução da carga de trabalho. O que é explicado pelo fato de somente duas consultas do TPC-H terem apresentado oportunidades de sintonia. Porém, essas duas consultas não foram reescritas pelo IQT para o Oracle, uma vez que este já implementa essa heurística (H8). Assim, observamos na Figura 3 que a abordagem automática apresentou uma leve piora em relação ao *baseline*. O que é explicado pelo fato da abordagem

automática ter tido o *overhead* de tentar reescrever cada uma das consultas recebidas e nenhuma delas possuir oportunidade de sintonia no Oracle.

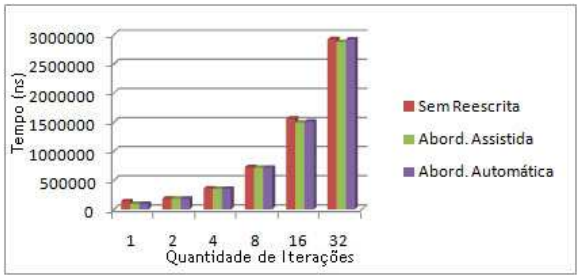


Fig. 1. Benchmark TPC-H no PostgreSQL

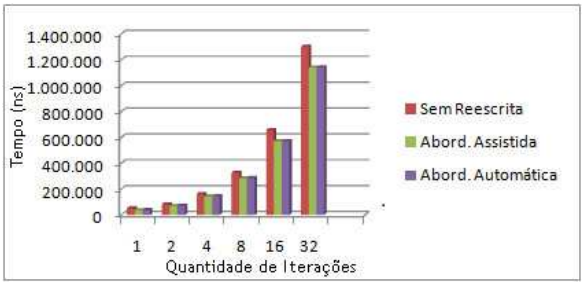


Fig. 2. Benchmark TPC-H no SQL Server

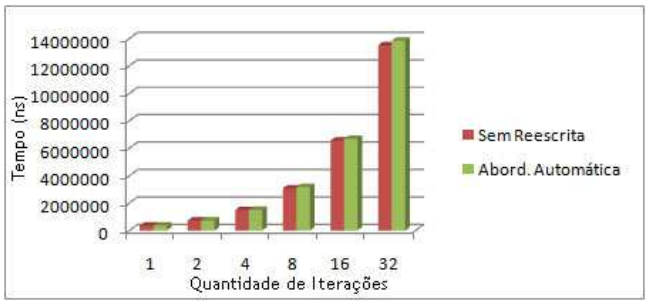


Fig. 3. Benchmark TPC-H no Oracle

4.2. Cenário 2: Base do TPC-H + Carga de Trabalho Sintética

As Figuras 4, 5 e 6 mostram que a abordagem assistida proporcionou uma grande redução no tempo de execução das cargas de trabalho. Já a abordagem automática proporcionou benefícios menores uma vez que esta envolve o *overhead* de sintonizar as instruções SQL recebidas em tempo de execução.

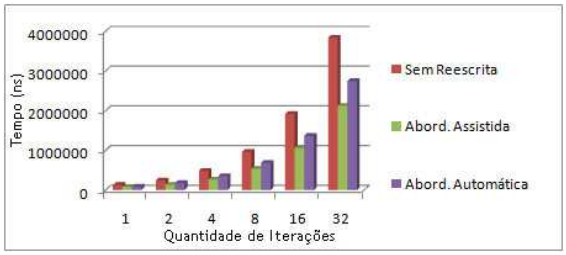


Fig. 4. Consultas sintéticas na base TPC-H no PostgreSQL

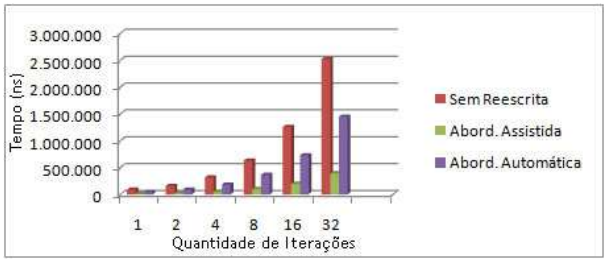


Fig. 5. Consultas sintéticas na base TPC-H no SQL Server

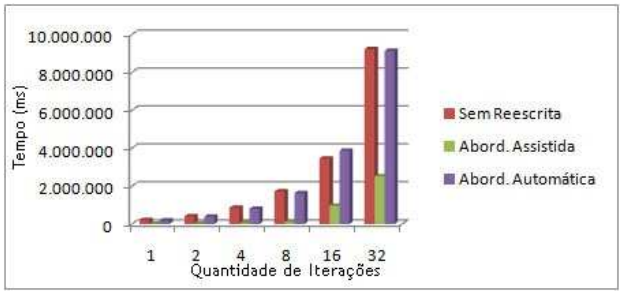


Fig. 6. Consultas sintéticas na base TPC-H no Oracle

5. Conclusões

Neste trabalho apresentamos uma ferramenta, denominada IQT, cujo objetivo consiste em auxiliar o processo de sintonia de instruções SQL em bancos de dados relacionais. A ferramenta proposta utiliza duas abordagens distintas para a sintonia de instruções SQL: uma abordagem assistida e outra automática. A ferramenta IQT pode ser utilizada em situações onde o otimizador de consultas não consegue produzir planos ótimos, mesmo usando métodos de acesso e estratégias de avaliação suportadas pelo SGBD. Os resultados dos experimentos realizados indicam que as duas abordagens utilizadas pela ferramenta IQT podem proporcionar ganhos de desempenho.

Referências

Boulic, R. and Renault, O. (1991) "3D Hierarchies for Animation", In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Bruno, N., Chaudhuri, S. and Ramamurthy, R. (2009) "Power Hints for Query Optimization", In: Proceedings of the 2009 IEEE International Conference on Data Engineering, ACM, Washington, DC, USA.

Dageville, B. and Dias, K. (2006) "Oracle's Self-Tuning Architecture and Solutions", In: IEEE Computer Society Technical Committee on Data Engineering, Oracle, USA.

Dyer, S., Martin, J. and Zulauf, J. (1995) "Motion Capture White Paper", http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Hamakrishnan, R. and Gehrke, J. (2008) "Sistemas de Bancos de Dados", São Paulo: McGraw-Hill.

Herodotou, H. and Babu, S. (2009) "Automated SQL Tuning through Trial and (Sometimes) Error", In: DBTest'09, ACM, Rhode Island, USA.

Holton, M. and Alexander, S. (1995) "Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials", Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press, p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Krishnaprasad, M., Liu, Z. H., Manikutty, A., Warner , J. W., Arora, V. and Kotsovolos, S. (2004) "Query Rewrite for XML in Oracle XML DB", In: Proceedings of the Vldb Conference, Toronto, Canada.

Moro, M. M. The Role of Structural Aggregation for Query Processing over XML Data. Ph.D. thesis, University of California, Riverside (UCR), USA, 2007.

Optimizer, D. X. (2010) "Embarcadero DB Optimizer XE", Available: http://www.embarcadero.com/products/db-optimizer-xe.

Oracle, S. O. (2010) "Quest SQL Optimizer for Oracle", Available: http://www.quest.com/SQL-Optimizer-for-Oracle, June.

Shasha, D. and Bonnet, P. (2003) "Database Tuning: Principles, Experiments and Troubleshooting Techniques", Morgan Kaufmann.

Smith, A. and Jones, B. (1999). "On the complexity of computing." In *Advances in Computer Science*, pages 555–566. Publishing Press.

Studio, O. D. (2010) "IBM Optim Development Studio", Available: http://www-01.ibm.com/software/data/optim/development-studio.

SCM-BT^2: Uma ferramenta para avaliação e ajuste de políticas de buffer no contexto da mídia *SCM*[*]

Júlio A. Tavares[1], **José de Aguiar Moraes Filho**[1], **Angelo Brayner**[1], **Saulo Medeiros**[1]

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Caixa Postal 1258 – 60.811-905 – Fortaleza – CE – Brasil

{julio,jaguiar,brayner,saulomedeiros}@unifor.br

Abstract. *Database system tuning activity aims at maximizing system resource utilization in order to provide performance gains for database applications. On the other hand, it is remarkable the popularization of a new media type, called Storage Class Memory (SCM). This type of media has emerged with the purpose of decreasing the latency found in the access time of HDDs, thus increasing the throughput which processors can consume more data. Nonetheless, DBMSs and performance tuning tools are not prepared yet to deal with the various underlying characteristics of these new media. In this paper, we propose a tool* SCM-BT^2*, which makes possible for DBAs to undertake performance monitoring related to the use of different buffer management policies, making it possible to compare the performance gains obtained in the context of SCM.*

Resumo. *As atividades de ajuste de desempenho em SGBDs têm como objetivo melhorar a utilização dos recursos do sistema, proporcionando benefícios para as aplicações que dependem desta infraestrutura de armazenamento. Neste contexto, e de forma ortogonal, é notável a popularização de um conjunto de novas mídias, denominadas Storage Class Memory (SCM). Este tipo de mídia surgiu com o propósito de diminuir a latência encontrada no tempo de acesso dos HDDs, para assim aumentar a vazão com a qual os processadores podem consumir mais dados. Entretanto, as ferramentas de ajuste de desempenho e os SGBDs ainda não estão preparados para tratar as diversas características subjacentes destas novas mídias. Neste trabalho, é proposta a ferramenta* SCM-BT^2*, permitindo aos DBAs empreenderem investigações de desempenho relacionadas ao uso de diferentes políticas de gerenciamento de buffer, tornando possível comparar os ganhos de desempenho obtidos no contexto da SCM.*

1. Introdução

A tarefa de manter um sistema gerenciador de banco de dados (SGBD) em níveis aceitáveis de desempenho é uma atividade desafiadora. Neste contexto, as equipes de TI buscam soluções eficientes para tratar os problemas relacionados à otimização de desempenho, que se tornam ainda mais complexos quando novas mídias (p.e., *SCM*) entram no rol dos dispositivos usados e os SGBDs não exploram todo o seu potencial.

É bem conhecido que o código do componente de gerenciamento de *buffer* do SGBD é responsável por mais de 1/3 da quantidade total de instruções executadas e

[*]Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

por cerca de 30% dos ciclos de *CPU* usados por um SGBD [Harizopoulos et al. 2008]. Todavia, a grande maioria das ferramentas que estão disponíveis para apoiar o DBA no processo de ajuste (*tuning*) do SGBD não levam em consideração os aspectos diretamente relacionados com a política de *buffer* e com a mídia de armazenamento subjacente. As ferramentas de ajuste de desempenho disponíveis atualmente, sejam de código aberto ou não, não têm como foco a política de *buffer*, mas fornecem uma visão geral do funcionamento de todos os componentes do SGBD. Além disso, as características da mídia (em especial, mídia *SCM*), não são levadas em consideração por elas.

Para tentar suprir esta lacuna, é proposta a ferramenta *SCM-BT*^{2}. Esta ferramenta[1] permite que o DBA realize investigações relacionadas ao uso de diferentes políticas de buffer, visando evidenciar os ganhos de desempenho que a mídia utilizada para armazenamento dos dados poderia proporcionar. Em especial, a *SCM-BT*^{2} realiza investigações de desempenho sob o ponto de vista da *SCM*, quando usada em SGBDs. A ferramenta possibilita a realização de análises *what-if*, que levam em consideração aspectos como *(i)* a quantidade de escritas realizadas na mídia; *(ii)* a taxa de acertos (*hit ratio*) da política de buffer escolhida; e *(iii)* o tempo de execução do *workload*. Através de uma interface simples e intuitiva, o DBA pode modificar os parâmetros das políticas avaliadas e comparar os resultados obtidos.

Um exemplo de uso da ferramenta pode ser encontrado no contexto dos dispositivos móveis, onde normalmente existe uma maior flexibilidade na implementação e personalização da camada de acesso aos dados. Neste caso, o gerenciamento dos dados pode ser implementado pela própria aplicação. Assim, a possibilidade de simular as perdas e os ganhos de desempenho obtidos com determinadas políticas de *buffer* pode ajudar na escolha de uma estratégia específica de implementação. Outro exemplo pode ser encontrado no contexto dos bancos de dados comerciais, onde a ferramenta permite realizar comparações consultando diretamente os metadados dos SGBDs.

O suporte fornecido pela *SCM-BT*^{2} se faz importante na simulação de workloads em SGBDs comerciais e em dispositivos móveis, onde usualmente utiliza-se a *SCM* (p.e., memória *FLASH*). A ferramenta disponibiliza mecanismos para a realização de análises *what-if* através de uma interface simples e intuitiva, permitindo que o DBA ou o arquiteto de aplicações realize comparações em poucas etapas. Apesar de ter os indicadores de desempenho voltados para a mídia *SCM*, a *SCM-BT*^{2} não se restringe apenas ao uso desta.

2. Arquitetura da ferramenta *SCM-BT*^{2}

A ferramenta *SCM-BT*^{2} utiliza uma arquitetura típica cliente/servidor, disponibilizando um serviço chamado *SCM DBT2 Simulation Engine*, responsável por *(i)* atender as requisições dos clientes contendo os parâmetros para a simulação de uma carga de trabalho (*workload*), utilizando uma política de *buffer* específica; *(ii)* executar a simulação de uma *workload* de acordo com os parâmetros recebidos na requisição; e *(iii)* enviar os resultados da simulação para o cliente, que então vai exibir estas informações em forma de gráficos. A Figura 1 (etapa ①), ilustra este fluxo.

Quando, todavia, a política de gerenciamento de buffer não é abertamente conhecida (como é o caso dos SGBDs comerciais), a *SCM-BT*^{2} possibilita a execução da

[1] Informações adicionais em http://goo.gl/IqOXq.

carga de trabalho diretamente no banco de dados, dessa forma não sendo mais necessário realizar a simulação da *workload*. Neste cenário, para obter os resultados, a ferramenta consulta diretamente o catálogo de dados do SGBD e assim obtém os dados relacionados ao desempenho (p.e., *hit* e *miss ratios*). A Figura 1 (etapa ②), ilustra este fluxo.

É importante ressaltar que o fluxo demonstrado na etapa ② (Figura 1) só ocorre quando o usuário opta pela execução da *workload* diretamente no SGBD. Neste caso, o serviço de simulação atua apenas como um gerador de carga, que irá coletar os resultados no dicionário de dados do SGBD avaliado. Esta funcionalidade foi implementada inicialmente para o *Oracle* e serão incluídos também o SQL Server e o PostgreSql.

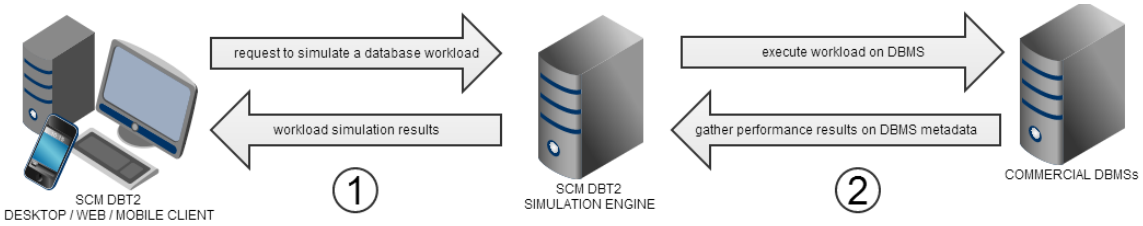


Figura 1. Arquitetura da ferramenta *SCM-BT²*.

Além das políticas de *buffer* já implementadas na ferramenta, é possível que o próprio usuário implemente uma política de *buffer* específica, de acordo com as suas necessidades. Com isso, a ferramenta permite que o usuário utilize o serviço de simulação de *workload* para avaliar a política implementada. Isto é possível devido a interface de plugins disponibilizada pela ferramenta. Com um conjunto de métodos bem definidos, expostos através de interfaces, é possivel prover novas políticas para a realização de *benchmarks*. Dessa forma, oferecendo esta flexibilidade, a *SCM-BT²* pode prover uma plataforma centralizada para a realização de testes de desempenho.

Implementamos uma aplicação cliente para a plataforma *Desktop*, utilizando tecnologias Java, e estão sendo desenvolvidas aplicações clientes para a plataforma web e para a plataforma móvel (*Android*/*iOS*). O serviço de simulação de *workloads* foi escrito utilizando a linguagem ANSI C e atualmente é executado no sistema operacional Linux.

3. Interface e Uso da Ferramenta

O uso da ferramenta *SCM-BT²* ocorre de forma bastante simplificada. Inicialmente, o usuário deve informar os parâmetros globais. Estes parâmetros são utilizados com os mesmos valores em ambas as políticas comparadas. Exemplos deste tipo de configuração são: (i) tamanho do banco de dados (quantidade de páginas), (ii) tamanho do *buffer* (quantidade de páginas), (iii) quantidade total de operações no *workload*, (iv) percentual de operações de escrita e (v) padrão (ou perfil) da carga de trabalho.

Este último parâmetro permite que o usuário escolha se a carga de trabalho irá conter rajadas de operações sequenciais ou se conterá apenas um conjunto de operações com o perfil essencialmente randômico. A configuração de perfil de operações sequenciais ou randômicas é uma opção bastante significativa quando se analisa políticas de *buffer* em um SGBD.

Para realizar as análises, o usuário deve escolher as políticas de buffer que deseja avaliar. A seleção de cada política poderá resultar na exibição de parâmetros específicos,

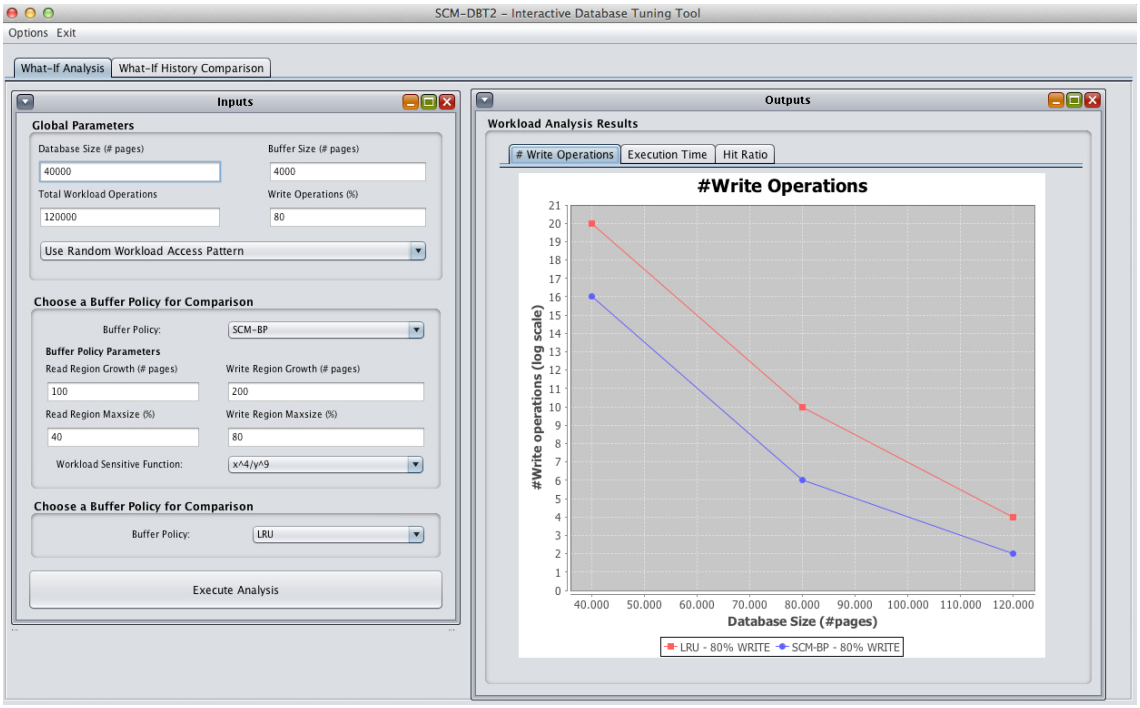


Figura 2. *SCM-BT²* Interface - Configuração dos parâmetros e exibição da quantidade de operações de escrita.

Figura 3. *SCM-BT²* Interface - Comparando a quantidade de escritas geradas pelas políticas de *buffer*.

de acordo com o seu modo de funcionamento. Inicialmente, disponibilizamos uma política "*SCM aware*"[Tavares et al. 2013] e uma política tradicional (*LRU*), para que seja possível comparar a execução de uma *workload* sob o ponto de vista de sistemas que utilizam a mídia *SCM* e de sistemas que utilizam uma política clássica.

Logo que a simulação é finalizada, os gráficos são atualizados na ferramenta, permitindo que o DBA faça o diagnóstico de desempenho e verifique a influência dos

parâmetros nos resultados obtidos. Neste momento, a ferramenta disponibiliza uma interface para que o usuário possa realizar análises *what-if*, verificando então o impacto da mudança nas configurações das políticas avaliadas.

A Figura 2 ilustra um gráfico comparando a quantidade de operações de escrita obtidas em uma simulação utilizando duas políticas de *buffer* distintas. Similarmente, através de outro gráfico (utilizando a aba *Execution Time*), é possível verificar o consumo de tempo na execuçao da *workload*, permitindo comparar as diferenças nas políticas analisadas.

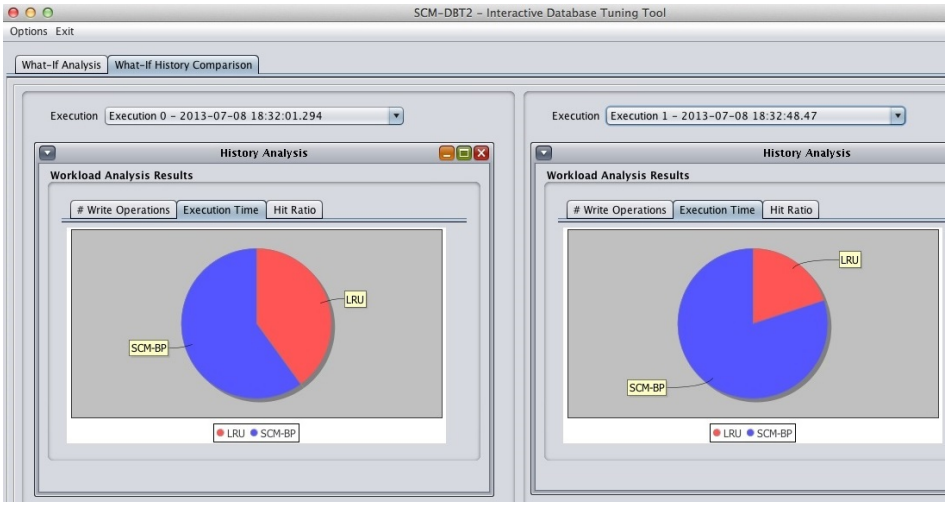


Figura 4. *SCM-BT²* Interface - Comparando os tempos de execução da *workload*.

Por fim, a ferramenta disponibiliza uma interface (ver Figuras 3 e 4) para comparação das simulações executadas previamente, realizando a análise sob o ponto de vista de parâmetros importantes no contexto da mídia *SCM*.

4. Trabalhos Relacionados

As ferramentas de monitoramento e ajuste de desempenho de SGBD atualmente disponíveis (p.e., [Dell 2013b, Dell 2013a, SoftTreeTech 2013, ION-DBA 2013]) têm como foco a verificação da ”saúde”do banco. Em outras palavras, estas ferramentas pretendem dar uma visão geral de cada componente em execução do SGBD. Atividades, tais como, análise arquivos de log, diagnóstico de problemas relacionados ao sistema operacional, análise de indicadores como o tempo de resposta, transações por segundo e a carga total do sistema, monitoramento das sessões dos usuários e de consumo de recursos computacionais, são comuns a cada ferramenta, diferenciando apenas no grau de informação exibido e na usabilidade de suas respectivas interfaces. Algumas como [Dell 2013a] podem, ainda, realizar simulações de cenários de carga de trabalho.

Existem outras ferramentas, tais como a *Toad DBA Suite* [Dell 2013c] e *Sql Developer* [Oracle 2013], que têm o objetivo de realizar uma administração centralizada do banco de dados e apoiar a implementação de sistemas de informação. Porém, o foco das funcionalidades disponibilizadas está voltado para a equipe de desenvolvimento de aplicações, por exemplo, a realização do ajuste de desempenho em consultas *SQL*.

Estas ferramentas, todavia, por apresentarem uma visão bastante abrangente do banco de dados, não disponibilizam suporte efetivo para o *tuning* de componentes específicos do SGBD. Elas não permitem, por exemplo, a realização de ajustes utilizando análises *what-if* modificando os parâmetros do *buffer* do SGBD, e também não levam em consideração o contexto específico de novas mídias *SCM* durante o seu uso.

5. Conclusões e Trabalhos Futuros

Neste trabalho, é proposta a ferramenta *SCM-BT*^{2}, que tem o objetivo de apoiar o processo de ajuste de SGBDs sob o ponto de vista das políticas de gerenciamento de *buffer*, dentro do contexto da memória *SCM*. A *SCM-BT*^{2} propõe apoio à área acadêmica, auxiliando o projeto e a experimentação de novas políticas de *buffer*, e para a área industrial, por exemplo na realização do *tuning* de aplicações que são executadas em dispositivos móveis e em SGBDs comerciais, através da execução ou simulação de cargas de trabalho, comparando os benefícios obtidos no uso de uma dada política de *buffer*.

Como trabalhos futuros incluímos: *(i)* adicionar monitoramento em tempo real dos indicadores de desempenho específicos das políticas de *buffer* dos principais SGBDs comerciais; *(ii)* implementar *drivers* que permitam fazer chamadas de baixo nível diretamente ao gerenciador de *buffer* do SGBD (no caso de SGBDs de código aberto), para realizar a simulação de *workloads* utilizando o próprio mecanismo de *buffer* do SGBD; e *(iii)* disponibilizar acesso à dispositivos móveis, provendo maior flexibilidade à ferramenta.

Referências

Dell (2013a). Benchmark factory. http://www.quest.com/benchmark-factory/. accessed at 20 June 2013.

Dell (2013b). Spotlight on oracle. http://www.quest.com/spotlight-on-oracle/. accessed at 20 June 2013.

Dell (2013c). Toad dba suite. http://www.quest.com/toad-dba-suite-for-oracle/. accessed at 20 June 2013.

Harizopoulos, S., Abadi, D. J., Madden, S., and Stonebraker, M. (2008). Oltp through the looking glass, and what we found there. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 981–992, New York, NY, USA. ACM.

ION-DBA (2013). Ion intelligent oracle tuning. http://www.ion-dba.com/index.html. accessed at 22 June 2013.

Oracle (2013). Oracle sql developer. http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html. accessed at 10 June 2013.

SoftTreeTech (2013). Db tools. http://www.softtreetech.com/dbtools/. accessed at 21 June 2013.

Tavares, J. A., de Aguiar Moraes Filho, J., Brayner, A., and Lustosa, E. B. S. (2013). Scm-bp: An intelligent buffer management mechanism for database in storage class memory. In *Proceedings of the XVIII Brazilian Symposium on Databases*, SBBD2013. accepted to publish.

XChange: Compreensão de Mudanças em Documentos XML*

Guilherme Martins[1], Celio Larcher Junior[1], Alessandreia Oliveira[1,2], Leonardo Murta[2], Vanessa Braganholo[2]
[1]Universidade Federal de Juiz de Fora
[2]Universidade Federal Fluminense
{guilherme, celio}@ice.ufjf.br,
{alessandreia, leomurta, vanessa}@ic.uff.br

Abstract. *Web Applications are increasingly using XML documents nowadays. Those documents evolve over time, so managing these changes becomes fundamental. The focus of existing research for comparing XML documents resides in identifying syntactic changes, which is not always enough. This paper presents the XChange approach to support the evolution of XML documents based on inference, using Prolog. Differently from existing approaches, XChange uses the syntactic changes in two versions of an XML document, which usually have a common purpose, to infer the reason of the changes.*

Resumo. *Documentos XML têm sido utilizados cada vez mais no contexto de aplicações Web. Um problema relacionado é que os documentos XML evoluem ao longo do tempo e gerenciar estas mudanças torna-se fundamental. Abordagens existentes têm seu foco na identificação de mudanças sintáticas para a comparação de versões de documentos XML, o que nem sempre é suficiente. Diante disto, este artigo apresenta a abordagem XChange para apoiar a evolução de documentos XML baseada em inferência, usando a linguagem Prolog. Diferente das abordagens existentes, XChange utiliza as mudanças sintáticas de duas versões de um documento XML, que usualmente têm um propósito em comum, para inferir a razão das mudanças.*

1. Introdução

Cada vez mais, é possível perceber a crescente utilização de documentos XML tanto na área de desenvolvimento quanto na troca e no armazenamento de informações [Moro *et al.* 2009]. Com isso, surge a necessidade de gerenciá-los, o que não é uma tarefa trivial. Tal dificuldade se deve, por exemplo, ao fato destes documentos apresentarem uma estrutura hierárquica e marcadores definidos pelo usuário. Apesar de permitir flexibilidade na representação dos dados, isto dificulta o acompanhamento de sua evolução, principalmente em grandes repositórios de dados.

Na literatura este problema já vem sendo estudado há algum tempo [Cobena *et al.* 2002; Wang *et al.* 2003]. Porém, o foco dessas abordagens está na detecção de mudanças sintáticas entre as versões de um documento XML e, em alguns casos, isso não é suficiente. De fato, existem situações onde não basta detectar o que mudou, mas também é necessário inferir a razão das modificações. Por exemplo, num cenário onde

* Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

um documento XML representa o cadastro de funcionários de uma empresa, pode ser necessário verificar o cargo de um funcionário num dado momento no passado ou quando determinados funcionários foram contratados ou demitidos. Se essa informação não existir explicitamente no documento, a sua percepção é dificultada se abordagens de detecção de mudanças puramente sintáticas forem utilizadas.

Diante disso, este trabalho apresenta o XChange [Oliveira *et al.* 2012], que usa inferência para apoiar a compreensão das mudanças entre versões de um documento XML. As versões são transformadas em fatos Prolog e, a partir de um conjunto de regras de inferência, pode-se deduzir a intenção do usuário ao criar a segunda versão. Enquanto uma abordagem puramente sintática apresentaria um conjunto de elementos adicionados ou removidos, o XChange visa extrair desse conjunto o significado de alto nível da mudança, facilitando a sua compreensão pelos especialistas do domínio.

Este trabalho está organizado como a seguir. A Seção 2 descreve alguns trabalhos relacionados a esta proposta. A Seção 3 apresenta a abordagem XChange, enquanto a Seção 4 descreve a sua implementação. Para finalizar, a Seção 5 apresenta as considerações finais e algumas sugestões de trabalhos futuros.

2. Comparação de Documentos XML

Existem iniciativas relacionadas à proposta deste artigo, que foram identificadas e classificadas em três contextos: detecção de mudanças em páginas Web frequentemente acessadas, mineração de mudanças em documentos XML e *diff* de documentos XML.

Existem ferramentas com o intuito de detectar mudanças em páginas Web frequentemente acessadas. WebVigiL [Chamakura *et al.* 2005] é um sistema de monitoramento de mudanças para páginas Web escritas em XML e HTML. CX-Diff [Jacob *et al.* 2005], por outro lado, é uma abordagem específica para detectar mudanças de conteúdo de *tags* de documentos XML. Por fim, o algoritmo heurístico proposto por Lim e Ng [2004], tem como objetivo descobrir mudanças entre dois arquivos XML ou HTML, hierarquicamente estruturados e representados por uma árvore ordenada. Estas abordagens analisam os aspectos sintáticos, não considerando a semântica das modificações dos documentos XML, como proposto pelo XChange.

Em relação à mineração de dados em documentos XML, a proposta de Zhao et al. [2006] é baseada na identificação de estruturas que se alteram frequentemente através de uma abordagem denominada mineração de delta estrutural. O objetivo é extrair informações através de sequências de mudanças estruturais no documento XML. Rusu et al. [2006] e Sonawane e Tambe [2013] utilizam regras de associação extraídas de versões de um documento XML para realizar previsões sobre alterações futuras do documento. Estas propostas estão mais concentradas na descoberta de regras de associação ou de estruturas que mudam com frequência e não na razão das mudanças.

Algumas propostas procuram descobrir a sequência de operações necessárias para transformar uma versão anterior de um documento XML em sua versão atual. Wang et al. [2003] propõem o algoritmo X-Diff que procura atingir a minimalidade da sequência de operações e usa árvores não ordenadas. O algoritmo Xy-Diff [Cobena *et al.* 2002] foi proposto para realizar a detecção de diferenças entre versões distintas de documentos XML. Sundaram e Madria [2012] apresentam uma abordagem de detecção de mudanças entre versões de documentos XML não ordenados, armazenados em um

banco de dados relacional. Estas abordagens têm seu foco no *diff* sintático ao invés de se preocuparem com a semântica das mudanças, como XChange propõe.

3. Visão Geral do XChange

Esta seção apresenta o XChange (http://gems.ic.uff.br/xchange) para apoiar a evolução de documentos XML. A partir do processamento de duas versões de um documento XML, é possível compreender as mudanças ocorridas. Para isso, são utilizadas as informações explícitas associadas a cada versão para deduzir conhecimento implícito sobre as mudanças. Esta dedução é realizada através de um conjunto de regras.

A Figura 1 apresenta o XChange. Os dados de entrada são constituídos por duas versões de um documento XML, de domínio qualquer, que são pré-processados e transformados em um conjunto de fatos Prolog, e por um conjunto de regras definidas por um especialista de domínio no início do processo. Estes dados de entrada são utilizados pelo motor de inferência Prolog, que, ao aplicar as regras sobre os fatos, retorna o significado de alto nível da modificação (razão da evolução do documento XML, de v1 para v2), que é a principal contribuição deste trabalho. Maiores detalhes sobre a abordagem estão disponíveis em Oliveira *et al*. [2012].

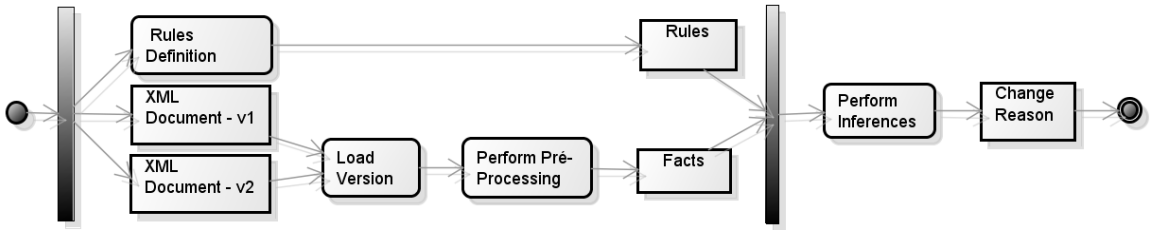


Figura 1: XChange [Oliveira *et al.* 2012]

4. Detalhamento Técnico do XChange

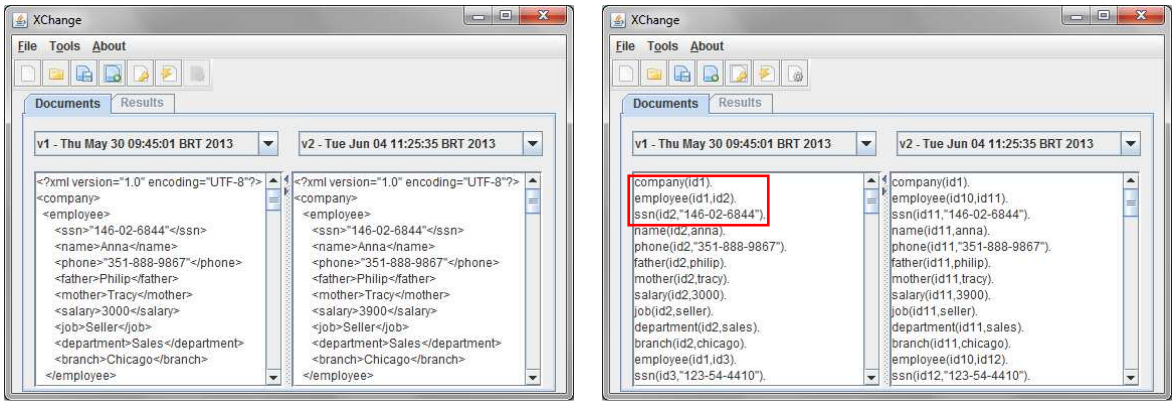
XChange foi desenvolvido, utilizando a linguagem Java, para criar um ambiente conciso onde o usuário pode monitorar as mudanças em diferentes versões de um documento XML. A biblioteca tuProlog [Denti *et al.* 2001] foi incorporada para apoiar a inferência. A abordagem pode ser dividida em três etapas que são descritas a seguir: geração de fatos Prolog, definição das regras e inferência de dados.

4.1. Geração de fatos Prolog

Para se realizar inferências lógicas sobre as informações contidas nas versões do documento XML, é necessário que estas sejam transformadas em fatos Prolog. Para isto foi utilizada a abordagem proposta por Lima *et al*. [2012]. O processo de tradução gera vários fatos Prolog a partir de um único documento XML, transformando elementos em predicados e seus conteúdos em constantes.

No cenário de cadastro de funcionários (Figura 2), o processo de tradução é realizado em 3 passos [Lima *et al.* 2012]. O primeiro traduz a raiz (*<company>)* em um fato com o nome do elemento e argumento único igual a um identificador gerado para estabelecer o vínculo com seus elementos filhos (*company* (id1)). Para relacionar predicados distintos, são criadas constantes Prolog, que atuam como identificadores que preservam a ligação pai/filho entre os elementos XML correspondentes. O segundo passo traduz os elementos simples sem atributos (por ex., *<ssn>146-02-6844</ssn>)*. O

resultado é um fato com nome igual ao nome do elemento e argumentos iguais ao identificador do elemento pai e o conteúdo do elemento corrente (*ssn (id2, "146-02-6844")*). O terceiro passo é a tradução dos elementos complexos. Um novo identificador é criado para relacionar os filhos ao pai. O resultado é a geração de um fato com o nome do elemento e dois argumentos: o identificador do elemento pai e um novo identificador gerado para referenciá-lo (por ex., *employee* (id1, id2)). XChange estende o método de tradução original [Lima *et al.* 2012], para utilizar duas versões de um documento XML (v1 e v2) como entrada (Figura 2.a) e gerar os fatos correspondentes (Figura 2.b).

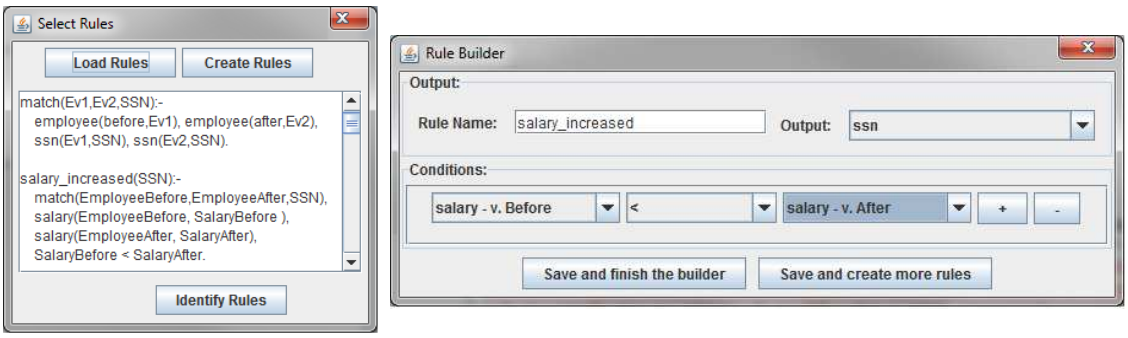


(a) Documento XML (v1 e v2) (b) Fatos Prolog (v1 e v2)

Figura 2: XChange

4.2. Definição de Regras

Um especialista com conhecimento no domínio pode, no início do processo, criar regras que permitam a inferência sobre fatos Prolog. XChange oferece a possibilidade de se criar regras ou carregar regras previamente. A Figura 3.a apresenta duas regras, previamente definidas, no cenário do cadastro de funcionários. A regra *salary_increased* analisa duas versões do documento XML (v1 e v2) e identifica os funcionários que receberam um aumento de salário. A regra auxiliar (*match*) é usada para vincular elementos de v1 e v2. A regra *match*, neste cenário, utiliza o SSN (equivalente ao CPF nos EUA) como um identificador (chave) para estabelecer a correspondência entre elementos de diferentes versões de um documento XML. Esta estratégia garante a unicidade do elemento *employee* e deve ser usada também em outros cenários/domínios (nesse caso, com chave diferente, dependendo do domínio).



(a) Regras (b) Criação de Regras

Figura 3. Criação de Regras

O XChange utiliza uma interface gráfica para facilitar a geração das regras pelo usuário. Com o módulo de criação de regras ativado, é solicitada a seleção do

identificador para estabelecer a correspondência entre os elementos nas diferentes versões. Escolhida a chave (neste cenário, *ssn*), o próximo passo é a definição da regra (Figura 3.b). É necessário informar o nome da regra (*salary_increased*), a saída (*ssn*) e as condições a serem satisfeitas *(salary – v.Before < salary – v.After)*. Um conjunto inicial, envolvendo os elementos e relação entre eles, é fornecido para a definição das condições. Os demais elementos que compõem a regra e se referem à identificação de elemento correspondente são inseridos automaticamente após a definição da regra.

4.3. Inferência de dados

Após a transformação das versões de um documento XML em fatos Prolog e da definição das regras, a máquina de inferência opera sobre os fatos combinando-os com as regras. Esta operação aplica cada regra definida aos fatos gerados, obtendo seus respectivos resultados. Com isso, é possível identificar o motivo da evolução de um documento XML a partir de suas versões (de v1 para v2). Para exemplificar, a Figura 4 mostra o SSN dos 4 funcionários que receberam aumento. O primeiro *ssn* mostrado corresponde à funcionária *Anna*. Como pode ser visto na Figura 2.a, *Anna* realmente recebeu aumento (*<salary>3000</salary>* em *v1 e <salary>3900</salary>* em *v2)*.

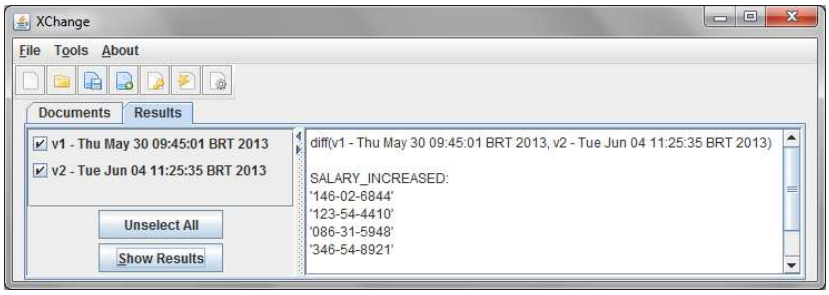


Figura 4. Resultados

5. Considerações Finais

Este trabalho apresentou o XChange, desenvolvido para viabilizar a identificação da razão real das modificações em documentos XML, tomando como base a análise das modificações sintáticas granulares em atributos e elementos. Para isto, o XChange usa um mecanismo de inferência baseado na linguagem Prolog e, a partir de um conjunto de regras e de fatos Prolog, pode-se inferir a razão das mudanças. A principal contribuição deste trabalho é, portanto, apoiar a compreensão de mudanças sobre documentos XML. Em documentos XML grandes, que passaram por muitas modificações entre suas versões, pode haver uma redução significativa no número de modificações a serem apresentadas ao usuário ao migrar de modificações sintáticas para semânticas. Essa relação pode permitir que centenas de modificações sintáticas sejam transformadas em dezenas de modificações semânticas. Essa característica tem potencial para contribuir com a facilidade de análise e compreensão sobre a evolução de documentos XML.

Além disso, esta abordagem não exige que o usuário seja especialista em Prolog, uma vez que utiliza uma interface gráfica que gera consultas Prolog a partir de uma seleção de opções em um alto nível de abstração, facilitando o processo de consulta. Por fim, as regras geradas são válidas para todos os documentos pertencentes ao mesmo esquema, o que faz com que essa etapa ocorra uma única vez para um dado esquema.

Uma limitação da abordagem atual é a identificação automática de elementos correspondentes. Hoje, no contexto do exemplo de utilização, o usuário utiliza a regra *match* (Figura 3.a) para identificar elementos correspondentes em v1 e v2. Esta regra usa um atributo-chave (SSN no exemplo). Dependendo da forma como os documentos XML são gerenciados, não há garantia de que o valor permanece o mesmo entre as versões (por ex., pode acontecer um erro de digitação no valor do SSN na v2). Para atenuar esse problema, uma abordagem baseada em análise de similaridade [Dorneles *et al.* 2009] está sendo desenvolvida. Outros aspectos a serem explorados: uso desta proposta na combinação de versões e na geração automática de regras semânticas.

Agradecimentos. Os autores agradecem ao CNPq e à FAPERJ pelo apoio financeiro.

Referências

Chamakura S., Sachde A., Chakravarthy S., Arora A. WebVigiL: Monitoring Multiple Web Pages and Presentation of XML Pages. In Data Engineering, 2005.

Cobena G., Abiteboul S., Marian A. Detecting changes in XML documents. In International Conference on Data Engineering (ICDE), IEEE, p. 41–52, 2002.

Denti E., Omicini A., Ricci A. tuProlog: A light-weight Prolog for Internet applications and infrastructures. Pr. Asp. Declar. Lang.: 184–198, 2001.

Dorneles C. F., Nunes M. F., Heuser C. A., Moreira V. P., Silva A. S. da, Moura E. S. de. A strategy for allowing meaningful and comparable scores in approximate matching. Inf Syst 34(8): 740–756, 2009.

Jacob J., Sachde A., Chakravarthy S. CX-DIFF: a change detection algorithm for XML content and change visualization for WebVigiL. Data Knowl. Eng. 52(2), 2005.

Lim S., Ng Y.-K. Change Discovery of Hierarchically Structured, Order-Sensitive Data in HTML/XML Documents. In Symposium on Applications and the Internet (SAINT), 2004.

Lima D., Delgado C., Murta L., Braganholo V. Towards querying implicit knowledge in XML documents. J. Inf. Data Manag. JIDM: 51–60, 2012.

Moro M. M., Braganholo V., Dorneles C. F., Duarte D., Galante R., Mello R. S. XML: Some papers in a haystack. SIGMOD Rec.: 29–34, 2009.

Oliveira A., Murta, L., Braganholo V. Uso de Inferência na Compreensão das Modificações em Documentos Semiestruturados. In Simpósio Brasileiro de Banco de Dados (SBBD), 2012.

Rusu L. I., Rahayu W., Taniar D. Mining changes from versions of dynamic XML documents. Knowl. Discov. XML Doc. KDXD: 3–12, 2006.

Sonawane V., Tambe S. Extracting interesting konowledge from versons of dynamic XML documents. IJRET Int. J. Res. Eng. Technol. 2(4), 2013.

Sundaram S., Madria S. K. A change detection system for unordered XML data using a relational model. Data Knowl. Eng. 72: 257–284, 2012.

Wang Y., DeWitt D. J., Cai J.-Y. X-Diff: an effective change detection algorithm for XML documents. In Intern. Conf. on Data Engineering (ICDE), p. 519 – 530, 2003.

Zhao Q., Chen L., Bhowmick S. S., Madria S. XML Structural Delta Mining: Issues and Challenges., 2006.

TripTag: Ferramenta de planejamento de viagens baseada em experiências de usuários de redes sociais*

Antônio H. G. Leite, Fabrício Benevenuto, Mirella M. Moro[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Belo Horizonte, MG

{antonioh, fabricio, mirella}@dcc.ufmg.br

Abstract. *The new Internet-connected generations of tourists search for destinations and plan their trips through online services. A new way of doing so is through "travel social networks", in which regular people share their opinion about touristic places. In such networks, the information is always updated and not biased according to agencies interests. However, there are two major problems: searching for a place in such networks the user to know where she or he wants to go, and acquiring knowledge about it requires reading its reviews, which vary from a handful to hundreds of texts. In this paper, we describe a new, exciting tool that automatically crawls different travel social networks, aggregates their content, tags them, allows to search for such tags, performs sentiment analysis and summarizes the reviews through word clouds.*

Resumo. *As novas gerações de turistas buscam por destinos e planejam suas viagens através de serviços online. Uma das mais novas e populares maneiras de explorar tais serviços é através das recém criadas "redes sociais de viagens", nas quais usuários compartilham suas opiniões sobre pontos turísticos. A informação é sempre atualizada, e as opiniões não tendem de acordo com os interesses de agências. Porém, existem dois problemas: buscar por um local requer que o usuário saiba para onde quer ir, e adquirir conhecimento sobre ele requer a leitura de suas avaliações, o que pode variar desde um pequeno número até centenas de textos. Neste artigo, descrevemos uma ferramenta nova que automaticamente coleta dados de diferentes redes sociais de viagem, agrega seus conteúdos, adiciona tags, permite a busca por tags, realiza análise de sentimento e resume as avaliações em nuvem de palavras.*

1. Introdução

As novas gerações de turistas, que vivem em um mundo conectado, planejam suas viagens e buscam por destinos através de serviços online. Uma maneira nova e que tem se tornado popular é explorar tais serviços através de *travel social networks*, nas quais usuários compartilham suas opiniões sobre pontos turísticos famosos, bem como sobre lugares desconhecidos. Uma vantagem é que a informação é sempre nova e atualizada, diferente das tradicionais agências de viagens e publicações específicas. Outra é a possibilidade de identificar o que várias pessoas estão pensando e qual o sentimento delas sobre o lugar, ao invés de apenas ter acesso à opinião do autor de um livro ou dos agentes de viagem.

*Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

Entretanto, existem dois grandes problemas: buscar por um local em tais redes requer que o usuário saiba para onde quer ir, e adquirir conhecimento sobre ele requer a leitura de suas avaliações, o que pode variar de um pequeno número a centenas de textos. Em outras palavras, é possível buscar pela cidade de Nova Iorque, mas não por uma "cidade grande e cosmopolita". Uma vez encontrada a cidade, o usuário precisa ler vários textos antes de formar sua opinião. Desse modo, apesar das redes sociais de viagem serem um grande avanço, elas ainda estão longe de facilitar a vida do usuário, que precisa de muito tempo de leitura para entender sobre os possíveis destinos.

O objetivo deste artigo é demonstrar as etapas de criação e funcionamento de uma ferramenta, chamada *TripTag*[1], para auxiliar na busca por pontos turísticos. As suas principais contribuições são assim resumidas: (*i*) a *TripTag* reune informações turísticas a partir das *várias* redes sociais existentes em um banco de dados; (*ii*) a partir dos dados, são geradas anotações (tags) que resumem as avaliações (escritas pelos usuários das redes) de cada local; (*iii*) com base nessas anotações, a ferramenta gera uma nuvem de termos mais frequentes e a análise de sentimento das avaliações; e (*iv*) a busca por locais pode ser agora realizada através das tags. É importante notar que a nuvem de termos e a análise de sentimento são consideradas excelentes resumos sobre as avaliações dos locais, facilitando assim a vida do usuário durante sua busca. Além da fácil visualização, a busca também é aperfeiçoada através das tags, com as quais o usuário pode buscar pelas características dos locais que deseja visitar.

A seguir, apresentamos uma breve descrição de esforços relacionados, a arquitetura da ferramenta, sua interface e funcionamento, e, finalmente, apresentamos conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

As redes sociais podem ser usadas como valiosa fonte de dados sobre perfis e preferências de usuários devido ao grande volume e tipo de informação compartilhada. Nesse contexto, várias aplicações surgiram na tentativa de explorar informações postadas em redes sociais, incluindo a análise de campanhas políticas [Tumasjan et al. 2010], a repercusão de variações em bolsas de valores [Bollen et al. 2010], a detecção e síntese da repercussão de desastres naturais [Sakaki et al. 2010] e epidemias [Gomide et al. 2011].

Entretanto, nenhum dos trabalhos citados aborda a construção de um sistema que, de forma automática, coleta avaliações de várias redes sociais relacionadas a viagens, gera tags e possibilita que sejam feitas buscas baseadas nessas tags. Geralmente, os sistemas existentes permitem que a busca seja realizada pelo nome da cidade ou região desejada e em apenas uma rede social. As tags geradas são utilizadas para mostrar ao usuário uma visão resumida (em nuvem de palavras) sobre o que as pessoas estão escrevendo sobre o local e qual o sentimento delas em relação aos locais. Sendo assim, nosso aplicação é inovadora e complementar aos esforços existentes na literatura.

3. Arquitetura da Ferramenta

O funcionamento da ferramenta é dividido nas seguintes etapas: obtenção dos dados, mineração de dados e análise de sentimentos. A Figura 1 ilustra tais etapas e o fluxo de dados entre elas. Cada etapa é detalhada a seguir.

[1]TripTag: `http://www.triptag.dcc.ufmg.br`

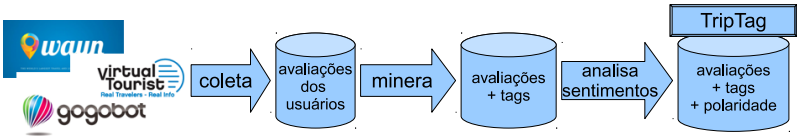


Figura 1. Fluxo de dados e processos

3.1. Coleta de Dados

O princípio fundamental para o desenvolvimento da ferramenta é utilizar apenas informações turísticas retiradas de redes sociais. Dessa forma, o primeiro passo foi escolher as fontes de dados mais adequadas dentre as várias redes sociais existentes. Inicialmente, foram escolhidas três redes sociais de viagem que são populares e possuem dados que facilitam a extração das informações dadas pelos usuários. São elas: *Where are you now?* (WAYN)[2] - a maior com 21 milhões de usuários, Gogobot[3] e Virtual Tourist[4].

Nessas redes sociais, os usuários se cadastram e possuem perfis com fotos e outras informações. As cidades também possuem perfis que são alimentados exclusivamente com informações que os próprios usuários postam através de suas contas. Dessa forma, todas as cidades nas redes sociais possuem uma página com uma enorme quantidade de informações provindas de pessoas que já as visitaram. Tal característica é a principal justificativa para termos escolhidos esses sites.

Porém, tais redes sociais não possuem APIs para a extração de dados. Desse modo, a primeira etapa é **coletar** os dados de tais redes através de *crawlers* específicos para cada website. Para viabilizar uma coleta mais rápida, foi definido um subconjunto de cidades para serem as sementes da coleta. O critério utilizado para a escolha das cidades foi a seleção das 200 cidades com o maior número de visitantes no mundo, pois essas seriam as com maior número de avaliações, provendo assim um bom volume de dados a serem processados. As coordenadas geográficas de cada cidade foram obtidas através da API do Google Maps e armazenadas no banco de dados. Em resumo, foram desenvolvidos três *crawlers* (um por website) que visitam o perfil das 200 cidades na respectiva rede social, identificam as avaliações dos usuários dessa cidade e as salvam em um banco de dados relacional.

3.2. Mineração de Dados

A segunda etapa é a mineração dos dados obtidos na etapa anterior. Tal etapa visa **gerar anotações** (tags) para cada cidade. Para cada cidade, são identificadas as palavras-chave e expressões que aparecem mais vezes nos textos e no maior de número de textos. A hipótese é: se uma palavra ou expressão é muito utilizada nas avaliações dos usuários sobre uma cidade, ela se relaciona fortemente com as experiências vividas pelo turista quando a visitou. As palavras ou expressões mais frequentes definem as tags para cada cidade. No contexto deste trabalho, não foram exploradas as relações entre os usuários das redes sociais escolhidas, o que será feito em trabalhos futuros. Nosso intuito foi utilizar as revisões dos usuários como uma fonte *crowsourced* de informações turísticas.

[2]WAYN: `http://www.wayn.com`
[3]Gogobot: `http://www.gogobot.com`
[4]Virtual Tourist: `http://www.virtualtourist.com`

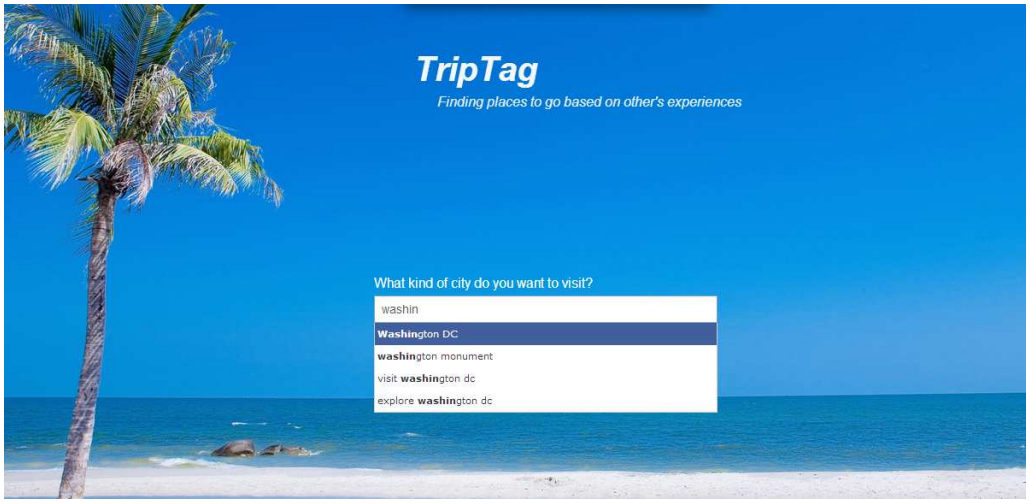


Figura 2. Interface inicial da ferramenta

O algoritmo utilizado é uma modificação do Apriori [Zaki and Meira Jr. 2013]. Inicialmente, o algoritmo faz a contagem das expressões com uma palavra e elimina todas as que forem menos frequentes que um determinado limiar. Em seguida, é calculada a frequência das expressões com duas palavras, e novamente são excluídas expressões infrequentes. O algoritmo segue elevando o grau desse *n-grama* a cada ciclo, e faz cortes até não ter mais expressões frequentes. Duas observações importantes: só são considerados os n-gramas formados por palavras adjacentes, e o processo também se encarrega de eliminar *stopwords* como artigos e conjunções.

3.3. Análise de Sentimentos

Finalmente, é realizada a **análise de sentimento** das avaliações. Existem várias formas de se realizar análise de sentimento, que variam desde abordagens léxicas até abordagens que utilizam técnicas de aprendizagem de máquina [Gonçalves et al. 2013]. Neste trabalho, por simplicidade, utilizamos uma biblioteca que implementa um classificador Naive Bayes que se encontra disponível em `http://text-processing.com/docs/sentiment.html`. Em trabalhos futuros, pretendemos investigar outras abordagens e avaliar qual a mais adequada para o nosso contexto.

Cada avaliação textual é submetida ao classificador de sentimentos que retorna a probabilidade com a qual o sentimento associado ao texto é positivo, neutro ou negativo. Esses três valores associados a cada avaliação são então armazenados no banco de dados.

4. Interface e Funcionamento

Após o banco de dados ser populado com as avaliações e suas tags, a ferramenta TripTag processa as consultas do usuário sobre tais dados. O sistema é baseado na linguagem PHP e no framework CakePHP[5], que utilizada o modelo de desenvolvimento de software *Model-view-controller* (MVC).

A ferramenta foi projetada para ser simples e fornecer visualizações para facilitar a compreensão das avaliações. Consequentemente, ela permite identificar facilmente o que

[5]CakePHP: `http://cakephp.org/`

Figura 3. Resultado para a cidade de Las Vegas

os visitantes pensam sobre cada cidade. A tela inicial é composta apenas por uma barra de busca na qual é possível fazer dois tipos de busca: pelo nome da cidade ou por tags. Ao digitar qualquer texto, o sistema busca no banco de dados nomes de cidades ou tags que contenham a palavra digitada. Os nomes de cidades ou tags encontradas são exibidos em uma lista, conforme ilustrado na Figura 2. Ao escolher uma palavra ou expressão específica, o usuário é redirecionado para o perfil da cidade. Se o usuário optar por uma tag, é criado um token na barra de buscas e novas tags podem ser escolhidas conforme o desejado. Depois de escolhidas as tags, ao clicar no botão <Search>, é exibida uma lista de cidades que contêm todas as tags utilizadas na busca e, ao clicar no nome de uma cidade, o usuário é direcionado para o perfil dela.

O resultado para uma busca que identificou a cidade de Las Vegas é ilustrado na Figura 3. É importante notar que o perfil de cada cidade também exibe um mapa com a sua localização no planisfério, uma nuvem de termos com as tags associadas a essa cidade e um gráfico que sumariza as análises de sentimento de todas as avaliações daquela cidade. Além disso, é exibida uma listagem de todas as avaliações na qual é possível que sejam feitas buscas através do campo <search>.

Outro exemplo é ilustrado na Figura 4, com as informações sobre a cidade de Chennai (anteriormente conhecida como Madras, na Índia). Diferentemente de Las Vegas, Chennai é conhecida pelos seus templos e possui avaliações mais positivas: acima de 60%, ao contrário de Las Vegas que ficou abaixo de 45%.

5. Conclusão

Este artigo apresentou a *TripTag*, uma ferramenta para busca de destinos de viagem que funciona em um banco de dados criado a partir de avaliações turísticas provenientes de redes sociais de viagens. A partir da contagem da frequência das palavras nas avaliações, são definidas tags relevantes baseadas apenas nas experiências relatadas pelos usuários das redes sociais. Essas tags serão agrupadas em uma nuvem de termos. As avaliações

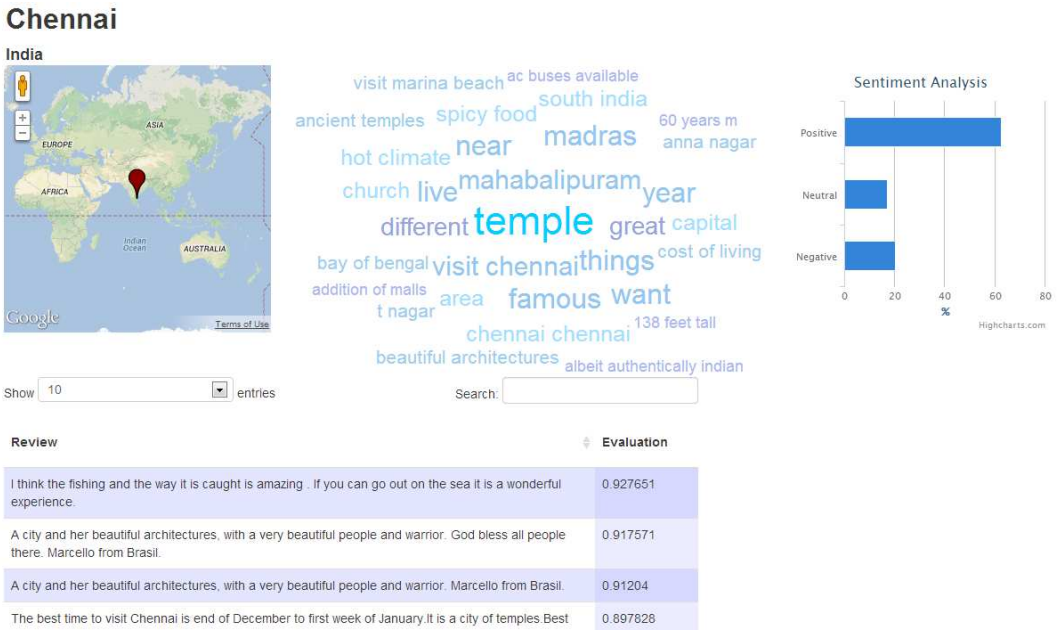


Figura 4. Resultado para a exótica Chennai

também passam pelo processo de análise de sentimentos a fim de sumarizar as incontáveis opiniões sobre cada cidade.

Observamos que existe um grande número de spams e avaliações com erros ortográficos, os quais dificultam a análise dos dados. Desse modo, como trabalho futuro, existe a possibilidade de melhorar os resultados das tags e da análise de sentimentos através da utilização de técnicas de *data cleaning*, por exemplo. Além disso, serão estudadas as relações existentes entre os usuários das redes sociais de viagem e como elas podem ser exploradas para o melhoramento da ferramenta. Também será desenvolvida uma forma de avaliar os resultados obtidos através da ferramenta.

Agradecimentos: Esse trabalho foi financiado por CNPq, FAPEMIG e InWeb, Brasil.

Referências

Bollen, J., Mao, H., and Zeng, X.-J. (2010). Twitter mood predicts the stock market. *CoRR*, abs/1010.3003.

Gomide, J., Veloso, A., Jr., W. M., Almeida, V., Benevenuto, F., Ferraz, F., and Teixeira, M. (2011). Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *Procs. of ACM WebSci*.

Gonçalves, P., Araújo, M., Benevenuto, F., and Cha, M. (2013). Comparing and combining sentiment analysis methods. In *Procs. of ACM COSN*.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Procs. of WWW*.

Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welpe, I. M. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Procs. of ICWSM*.

Zaki, M. and Meira Jr., W. (2013). *Data Mining and Analysis: Foundations and Algorithms*. Cambridge Un. Press (Draft available at `http://bit.ly/12WMuem`).

Análise de Sentimentos para Previsão das Condições de Trânsito*

Bernardo Lauand[1], Jonice Oliveira[1]

[1] Universidade Federal do Rio de Janeiro – DCC/IM, PPGI

Caixa Postal: 68.530, Cep: 21.941-909 – Rio de Janeiro – RJ – Brasil

belauand@ufrj.br, jonice@dcc.ufrj.br

Abstract. *Due to several buildings and routes changes related with preparations for the World Cup and Olympic Games, as well as the growing number of vehicles, the Rio de Janeiro's citizens constantly suffer with traffic jam. This project, called TweeTraffic, aims to use Twitter to predict traffic conditions, by mining the notifications provided by drivers and official sources.*

Resumo. *Devido às inúmeras obras e mudanças de vias por conta dos preparativos da Copa do Mundo e Jogos Olímpicos, além do crescimento do número de veículos em circulação, os moradores do Rio de Janeiro sofrem constantemente com congestionamentos. Este projeto, chamado TweeTraffic, visa utilizar o Twitter para a predição das condições do trânsito, minerando os relatos dos motoristas, bem como notificações de fontes oficiais.*

1. Introdução

Atualmente, a cidade do Rio de Janeiro apresenta, em média, 94 quilômetros de trânsito lento por dia [PDTU/RMRJ 2012]. Condição que deve piorar pelo aumento constante de sua frota, com previsão que alcance em 2016 a 3.000.000 (três milhões) de veículos [Oliveira 2011]. Fora isso, a cidade enfrenta inúmeras alterações devido aos preparativos de dois grandes eventos mundiais: a Copa do Mundo (2014) e os Jogos Olímpicos (2016). Além dos congestionamentos típicos da hora do *rush*, os condutores precisam lidar também com situações imprevistas (ex: acidentes e enchentes). Através da análise dos sentimentos, o projeto *TweeTraffic* prevê as condições de trânsito na cidade do Rio de Janeiro, de maneira que os cidadãos possam planejar melhor as suas rotas.

2. Trabalhos Relacionados

O Google Maps Brasil [Google 2012] informa as condições de trânsito nas principais vias de algumas cidades brasileiras, além de disponibilizar um histórico do tráfego com base em cada dia da semana. O Google Maps obtém informações do trânsito de autoridades locais e também utiliza dados dos próprios usuários para ajudar a medir a

* Agradecemos ao CNPq e à FAPERJ. Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts. Para executar a ferramenta, execute o arquivo disponível em https://www.dropbox.com/s/3wffpmoa5ouka2z/TweeTrafficInterface.jar.

velocidade, coletando de forma anônima os dados como posição e velocidade dos usuários que estejam utilizando o aplicativo [Hardware 2012].

O Endernoto [2011] é uma proposta para divulgar as condições de trânsito das ruas da cidade de Jakarta, na Indonésia. A polícia de Jakarta utiliza o Twitter para divulgar notícias sobre as condições de trânsito locais, que são apresentadas em um mapa através de um aplicativo para plataforma móvel Android.

Através do Waze [2012] o usuário pode compartilhar as informações sobre as condições de trânsito e utilizar as informações disponibilizadas por outros usuários como: localização de paradas policiais (*blitz*), congestionamentos, acidentes e obras.

O trabalho de Ribeiro et al. [2012] descreve um sistema de mineração de textos aplicado ao *Twitter* procurando por padrões de texto relacionados ao trânsito da cidade de Belo Horizonte utilizando apenas fontes oficiais. Para auxiliar o geoprocessamento, utiliza um *gazetteer* que contém os nomes, geometria e geolocalização de logradouros, bairros, cruzamentos e trechos de vias.

As contribuições do *TweeTraffic*, em comparação ao estado da arte, podem ser resumidas da seguinte forma:

- Esta proposta se baseia exclusivamente no conteúdo textual provido por livre participação da população (*crowdsourcing*) ou por fontes oficias, sem utilizar qualquer outro recurso. Embora tenha sido construído e testado com as ruas do Rio de Janeiro, esta abordagem facilitaria a ampliação do seu uso em outras cidades brasileiras sem grandes mudanças arquiteturais ou de conteúdo.
- Utiliza informações providas por fontes oficiais e por fontes não-oficiais (população). Este é um ponto a se ressaltar, pois a população é um importante mecanismo na detecção e divulgação rápida de informação, muitas vezes mais rápida que as fontes oficiais. Fora isso, para localidades poucos conhecidas, a população torna-se o único meio de sensoriamento da região. Comparando-o especificamente com o Waze, que também trabalha com o conceito de *crowdsourcing*, seus mapas do Rio de Janeiro não estão bem construídos e não há muitas informações sobre o trânsito já que a comunidade brasileira no Waze é pequena. Isso torna o programa difícil de ser usado.
- As regras utilizadas na inferência das condições de trânsito foram construídas após observação de uma massa de dados.

3. TweeTraffic: A Ferramenta

O TweeTraffic é uma ferramenta que consiste em um cliente e um servidor. O servidor é responsável por extrair e analisar informações provenientes do *Twitter*, onde se conecta periodicamente em busca de informações relevantes sobre as principais ruas. Ele seleciona o conteúdo, o registra no banco de dados e realiza a análise de sentimentos. Esta é a etapa chamada “análise estática”. O servidor, quando consultado pelo cliente, realiza a “análise dinâmica”, inferindo a condição de trânsito da via consultada. O cliente (Figura 1) é um aplicativo *Android* onde o usuário pode verificar as condições de trânsito nas ruas da cidade do Rio de Janeiro e avaliar a resposta dada pelo sistema. Na Figura 2 temos a arquitetura conceitual do projeto TweeTraffic.

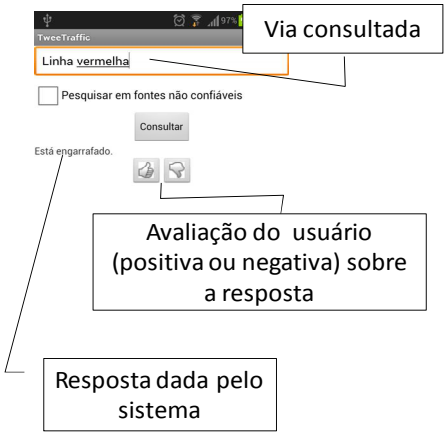


Figura 1 – Interface do Cliente [Lauand 2013]

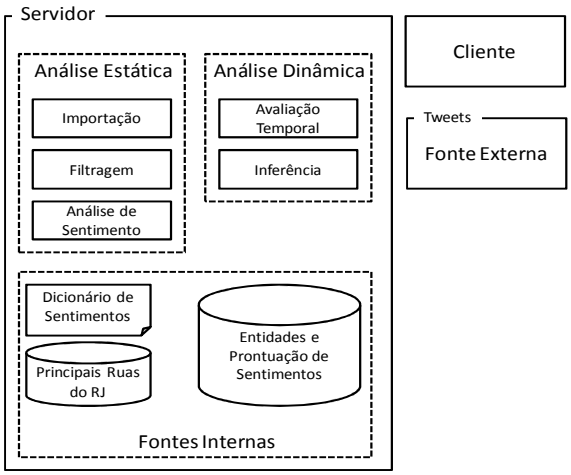


Figura 2 – Arquitetura Conceitual [Lauand 2013]

4. Verificação do estado atual do trânsito

Para realizar uma consulta, o cliente se conecta ao servidor e envia a rua a ser consultada (Figura 1). A seguir são realizadas as seguintes ações:

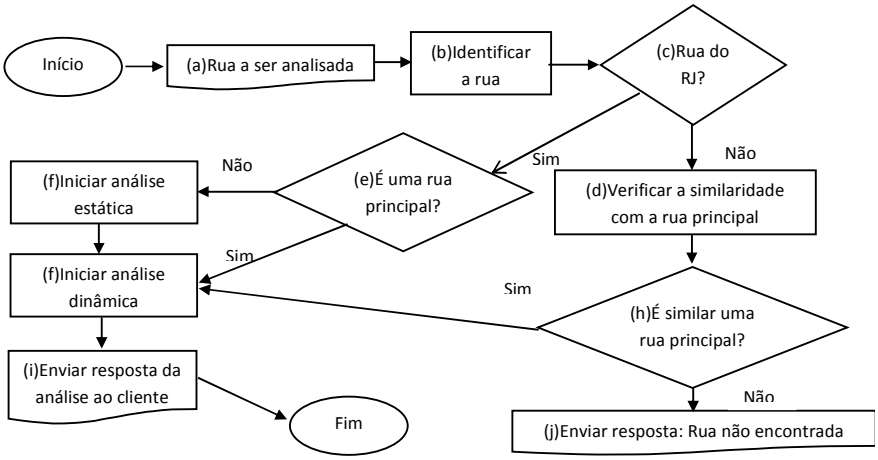


Figura 3. Fluxograma da identificação do estado [Lauand 2013]

4.1. Identificação da Rua

Ao receber o nome da rua a ser consultada, o servidor procura por abreviaturas comuns e as substitui pela sua palavra correspondente (figura 3.b), já que diferentes logradouros possuem nomes iguais e o que os distinguem são as denominações de via. Por exemplo: Avenida Rui Barbosa (bairro Flamengo, zona sul do Rio de Janeiro), Rua Rui Barbosa (bairro Guaratiba, zona oeste) e Travessa Rui Barbosa (zona norte).

4.2. Verificação da Região

Após a fase de Identificação da Rua (figura 3.b), o servidor verifica se o nome recebido é de uma rua da cidade do Rio de Janeiro (figura 3.c). Caso a rua pesquisada não seja uma rua principal, será feita uma Análise Estática (seção 4.3), caso contrário, a Análise Dinâmica será feita (seção 4.4). Consideramos como ‘ruas principais’: "linha amarela",

"linha vermelha", "avenida presidente vargas", "avenida rio branco", "rua humaitá", "rua são clemente", "avenida nossa senhora de copacabana", "praça da bandeira", "ponte rio niteroi", "avenida brasil", "avenida das americas", "perimetral", "av nossa senhora de copacabana" e "rebouças".

4.3. Análise Estática

A análise estática é realizada continuamente sobre os dados captados periodicamente, processando-os e armazenando o resultado da análise de sentimento sobre cada entidade. Caso a via a ser consultada não seja uma via principal, a análise estática é acionada após o usuário informar a rua procurada.Esta análise é composta de três etapas: importação, filtragem e análise de sentimento.

- **Importação**: O servidor importa do Twitter todas as mensagens que contenham o nome da rua, postadas nos últimos 60 minutos e que não estejam armazenados no banco de dados.

- **Filtragem**: Caso o usuário opte por fazer a pesquisa considerando apenas os autores confiáveis, os *tweets* dos autores que não constam na relação de autores confiáveis são eliminados nessa etapa. Consideramos um "autor confiável" os perfis do Twitter de importantes veículos de comunicação como perfis oficiais de revistas, jornais, estações de rádio e televisão, além de perfis de entidades da prefeitura, como CET-RIO, Rio Trânsito e o Centro de Operações da Prefeitura do Rio de Janeiro.

- **Análise de Sentimento**: O servidor possui um Dicionário de Sentimentos, que é composto por um conjunto de palavras e suas respectivas classificações. Uma palavra pode ser classificada em: positiva, negativa ou modificadora. Uma palavra positiva é aquela que sugere uma boa condição de trânsito (ex: livre, bom, rápido). Já uma negativa denota uma má condição de trânsito (ex: ruim, parado, batida). Uma palavra modificadora por sua vez é aquela que altera a classificação de uma palavra de positiva para negativa ou vice versa (ex: sem, não, nenhum). A primeira fase da análise de sentimento separa o texto dos *tweets* em frases e as analisam individualmente, descartando as frases interrogativas. Para cada frase, contabiliza-se o número de palavras positivas e negativas e verifica-se a existência de palavras modificadoras. Caso haja, altera-se o valor das palavras positivas ou negativas próximas (distância de até 2 palavras) à palavra modificadora. O sistema classifica um *tweet* de acordo com a quantidade de palavras negativas e positivas**.**

4.4. Análise Dinâmica

Ela consiste em avaliar os *tweets* da última hora levando em consideração o resultado da análise estática e o horário em que o *tweet* foi publicado. A partir daí é feita uma inferência, baseando-se nas quatro situações distintas:

- **Caso 1:** Caso não haja nenhum *tweet* sobre a via consultada nos últimos 60 minutos, o sistema retorna: "Não há informações suficientes sobre essa via.".

- **Caso 2:** Caso haja *tweets* na última hora com classificações divergentes de acordo com a análise estática, calcula-se o intervalo de tempo entre os *tweets* mais recentes e conflitantes. Se esse intervalo for superior a 15 minutos, o conflito é ignorado e considera-se a classificação do *tweet* mais recente como certa. Caso este seja positivo, o sistema retorna: "Não está engarrafado.". Caso contrário: "Está engarrafado.".

•**Caso 3:** Caso haja *tweets* na última hora com classificações divergentes de acordo com a análise estática e o intervalo de tempo entre os *tweets* conflitantes seja menor do que 15 minutos, é contabilizado o total de *tweets* positivos e negativos. Caso haja mais *tweets* positivos, o sistema retorna: "Provavelmente não está engarrafado.". Caso haja mais *tweets* negativos, o sistema retorna: "Provavelmente está engarrafado.". Caso o número de *tweets* positivos seja igual ao número de *tweets* negativos o sistema considera a classificação do *tweet* mais recente como certa. Se positivo, retorna: "Provavelmente não está engarrafado.". Caso contrário: "Provavelmente está engarrafado."

•**Caso 4:** Caso haja *tweets* na última hora e não haja divergência de classificação de acordo com a classificação estática, o sistema retorna: i) "Não está engarrafado" caso haja ao menos um *tweet* positivo na última hora, ii) "Está engarrafado." caso haja ao menos um *tweet* negativo na última hora e iii) "Provavelmente não está engarrafado" caso todos os *tweets* da última hora sejam neutros.

5. Estrutura do banco de dados

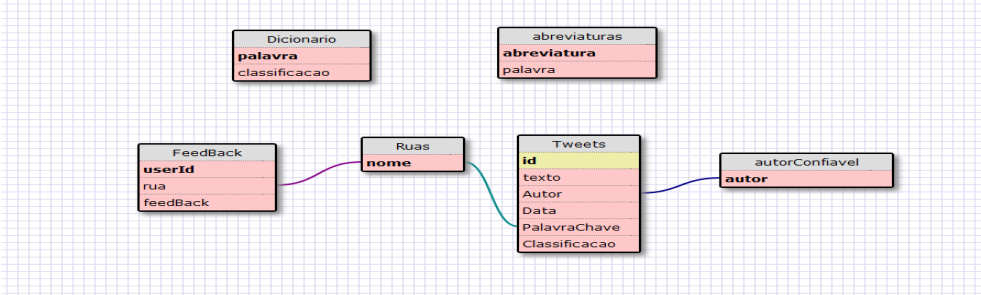


Figura 3 - Estrutura do banco de dados

A tabela dicionário armazena o conjunto de palavras predeterminadas e suas respectivas classificações, a tabela abreviaturas contém abreviaturas normalmente utilizadas em nomes de ruas e os nomes associados a elas.

Na tabela ruas há o nome de todas as ruas da cidade do Rio de Janeiro e a tabela autor confiável contém uma lista de usuários do *Twitter* que possuem uma relação direta com o trânsito. Já na tabela Tweets há os *tweets* dos últimos 60 minutos que o sistema identifica como relevante na análise da condição de trânsito do Rio de Janeiro.

6. Exemplo de Uso

Para facilitar a avaliação da ferramenta, criamos uma interface *desktop*[1]. Assim, qualquer leitor poderá testar a solução sem precisar instalá-la em um dispositivo móvel. Neste exemplo, mostramos o resultado da consulta realizada no dia 10/07 às 22h58min sobre as condições da Avenida Presidente Vargas utilizando apenas as fontes oficiais (Figura 4), todos os *tweets* (Figura 5) e o estado real da via (Figura 6).

7. Conclusão

O *TweeTraffic* mostrou-se uma ferramenta prática, trazendo bons resultados. Pelo tratamento ser exclusivamente textual, enfrentamos algumas dificuldades como a distinção de ruas de outras localidades, mas com o mesmo nome de ruas cariocas. Em vias extensas e de mão dupla, ocorre a dificuldade em se descobrir em que região e

sentido da rua se encontra congestionado. Outra dificuldade encontrada foi o fato de os usuários do *Twitter* não georeferenciarem seus *tweets*, tornando impossível descobrir a real localização dos usuários na hora de filtra-los. Outro problema encontrado foi a recente mudança na API do *Twitter* que só permite 180 requisições a cada 15 minutos. Alterações no código foram necessárias e limitações do número de requisições que o programa pode atender apareceram.

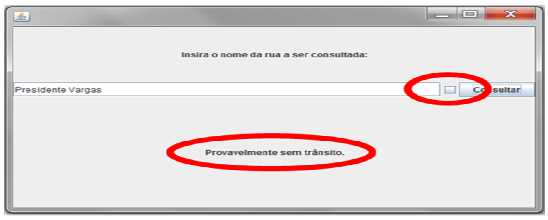


Figura 4 – Previsão com fontes confiáveis (oficiais)

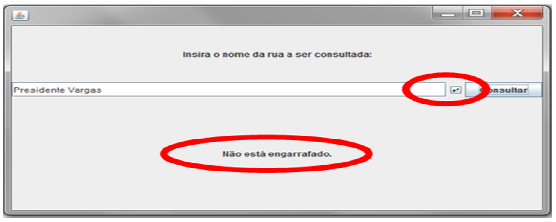


Figura 5 - Previsão com todas as fontes (oficiais e não-oficiais)

Figura 6 – Condição real, mostrada pela Câmera da CET-Rio[1]: Via totalmente livre.

Referências

Endarnoto, S. K. (2011). Traffic Condition Information Extraction & Visualization from Social Media Twitter for Android Mobile Application. In *Electrical Engineering and Informatics (ICEEI)*, pages 1 – 4.

Google (2012), “Google Maps”, http://maps.google.com.br, Setembro

Hardware (2012) "Dados de trânsito em tempo real no Google Maps para mais cidades brasileiras", http://www.hardware.com.br/noticias/2012-08/maps-transito-brasil.html, Setembro

Lauand, B., Oliveira, J. (2013). TweeTraffic: ferramenta de análise das condições de trânsito baseado nas informações do Twitter. In *II Brazilian Workshop on Social Network Analysis and Mining (BraSNAM 2013)*. SBC

Oliveira, A. (2011). O Árduo Desafio de Ir e Vir na Cidade do Rio de Janeiro. In *Revista Eletrônica Novo Enfoque*, v. 13, n. 13, p. 170 – 187. UCB

PDTU (2012) “Plano Diretor de Transporte Urbano da Região Metropolitana do Rio de Janeiro”, http://www.pdtu.rj.gov.br/indexini.php, Julho.

Ribeiro, S. et al. (2012). Observatório do Trânsito: sistema para detecção e localização de eventos de trânsito no Twitter. In *Simpósio Brasileiro de Bancos de Dados*. SBC

[1] http://www.rio.rj.gov.br/web/riotransito

Hermes: Identificação de Menores Rotas em Dispositivos Móveis*

Eduardo Augusto Sobral Junior[1], Jonice Oliveira[1]

[1] Universidade Federal do Rio de Janeiro

Caixa Postal: 68.530, Cep: 21.941-909 – Rio de Janeiro – RJ – Brasil

easobral@gmail.com, jonice@dcc.ufrj.br

Abstract. *Traffic jam is a frequent problem in Brazilian metropolises. Part of this problem could be solved if drivers knew alternative routes. In addition, we note the increase of computational power and versatility of phones and mobile devices. This project, entitled Hermes, aims to recommend new routes from the original-planned route and data from what has been traveled. All data collection, processing and recommendation are made locally. In this article, we also show the results obtained by one simulation, where we analyzed the solution's effectiveness and efficiency.*

Resumo. *Trânsito lento é um problema recorrente nas grandes cidades brasileiras. Parte do problema poderia ser resolvido se os motoristas conhecessem alternativas de rotas. Junto a este fato, notamos o aumento do poder computacional e versatilidade dos celulares e dispositivos móveis. Este projeto, intitulado Hermes, visa recomendar novas rotas a partir da rota planejada e dados do percurso realizado. Toda a coleta de dados, processamento e recomendação são feitos localmente. Neste artigo, também são mostrados os resultados obtidos através de uma simulação, onde foram analisados a eficácia e eficiência da solução.*

1. Introdução

Nos grandes centros urbanos, problemas como congestionamentos se tornam cada vez maiores. Além da perda de tempo e impacto financeiro, este problema impacta diretamente a qualidade de vida das pessoas. Congestionamentos também atrapalham serviços de resgate e prestação de socorro de pessoas em situações de emergência.

O problema de encontrar rotas de trânsito na malha rodoviária é análogo ao de encontrar caminhos em grafos, o que exige um bom poder computacional. No caso de dispositivos móveis, ocorre a dependência de um servidor (que realizará todo o processamento e devolverá o resultado), bem como uma conexão à Internet. Uma abordagem cliente-servidor para dispositivos móveis possui algumas vantagens como: i) Coleta e agregação de uma maior quantidade de informações sobre a cidade e ii)Custo menor de processamento e menor gasto da bateria nos dispositivos móveis. Porém, em situações de emergência, onde os desastres podem afetar a infraestrutura local de comunicação ou em áreas onde não há conexão à Internet, o modelo cliente-servidor não se aplica, exigindo o processamento local das rotas.

* Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

Hermes é um aplicativo desenvolvido para dispositivos móveis Android que visa recomendar o caminho de menor tempo até um determinado local. Neste trabalho iremos abordar o aspecto de encontrar o menor caminho (considerando o custo em tempo) entre dois pontos da malha rodoviária de uma cidade.

2. Trabalhos Correlatos

Ding et al. [2007] descrevem uma solução que sugere rotas, a qual se baseia em VANET[1] para conseguir informação do estado local das vias e deduzir um bom caminho.

Thiagarajan et al. [2009] exploram um mecanismo de coletar dados do GPS durante um percurso e mapeia os pontos em ruas, estimando o tempo gasto por trechos. Os autores apenas demonstram o mapeamento, mas não oferecem soluções para a recomendação de rotas.

Schultes [2008] faz uma revisão dos algoritmos mais comumente utilizados no roteamento em estradas. Diferentemente do Hermes, as malhas rodoviárias são extensas e o autor não considera a execução dos algoritmos em dispositivos móveis, que possuem processamento e memória limitados.

O Waze[2] é um aplicativo cliente-servidor que permite que os usuários compartilhem colaborativamente as informações sobre as condições de trânsito: localização de paradas policiais (*blitz*), congestionamentos, acidentes e obras. Ao informar o ponto inicial e final, o aplicativo dimensiona a melhor rota.

O sistema Hermes, apresentado neste artigo, atua no principal desafio da área: capturar, processar e recomendar rotas utilizando unicamente os recursos computacionais de um dispositivo móvel.

3. Arquitetura

Como mostrado na Figura 1, o sistema trabalha com fontes externas e possui 3 módulos, os quais são explicados a seguir.

3.1. Fontes Externas

GPS[3] - Os trechos já percorridos são coletados através do GPS do próprio dispositivo. Para isto, utilizamos a API do Android de acesso a esse dispositivo.

Mapas – Para obtermos as informações sobre as vias e suas direções, utilizamos o OpenStreetMaps[4]. O banco de dados da aplicação se limitou à região contida entre as latitudes -23.099 e -22.743 e entre as longitudes -43.798 e -43.137, o que corresponde à cidade do Rio de Janeiro. Foi obtido um arquivo .osm e importado em um banco de dados postgresql, utilizando a ferramenta Osmosis para isso. Como esse banco de dados possui muito mais informação que a necessária para este trabalho (há informação de rota de ônibus, localização de determinados edifícios, alguns nomes de ruas, traçado de lagos e rios) foi retirada dele apenas o traçado da malha rodoviária (devido à tentativa

[1] VANET - Vehicular Ad-Hoc Network
[2] http://www.waze.com/
[3] GPS - Global Positioning System
[4] http://www.openstreetmap.org/

de ocupar o menor espaço possível nos dispositivos). Essa informação se limita aos nós (id, latitude e longitude) e arestas (quais nós são ligados e uma estimativa do tempo de viagem baseada na distância).

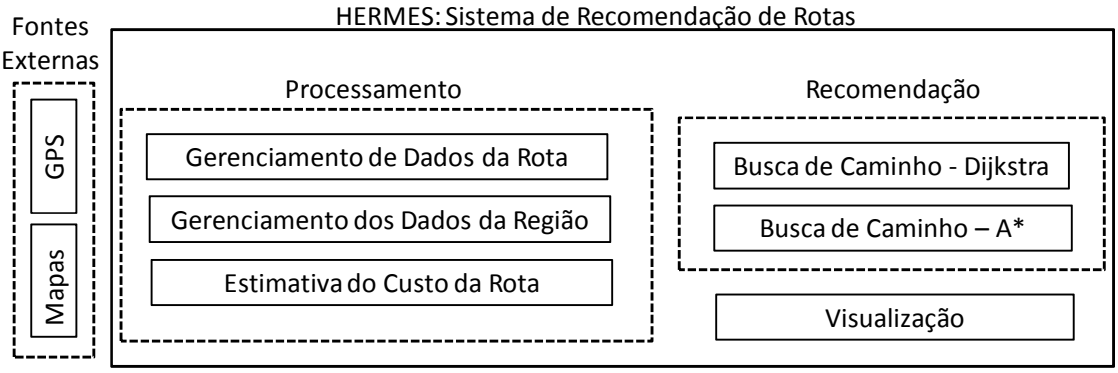


Figura 1: Arquitetura Conceitual

3.2. Processamento

Gerenciamento dos Dados da Rota – este é o módulo responsável por gerenciar os dados de uma viagem em uma malha/localidade. Os dados utilizados são representados através de um grafo direcionado com custos nas arestas, onde cada vértice é um ponto representativo da rota, como os cruzamentos. Podemos mencionar como responsabilidades deste módulo:

- coletar as informações da viagem - são obtidas iniciando-se um serviço que fica em espera e periodicamente captura a posição indicada pelo GPS e o horário.

- operações básicas do grafo – atualizar e prover informações em resposta a solicitações dos demais módulos, como detalhes dos vértices (identificação, nome e posição), listas de adjacências, distância entre nós, dentre outras.

- manter as informações sobre os custos – os custos, representados como pesos nas arestas, estão relacionados ao tempo gasto em se deslocar em um ponto a outro.

Figura 2: Modelagem do Banco de Dados

Nosso grafo é mantido em um banco de dados relacional SQLite (o único disponível para a plataforma no início do desenvolvimento), como mostrado na Figura 2. Cada aresta representa um segmento de rua e alguns nós representam as esquinas (outros nós existem para fins de visualização, com o intuito de traçar caminhos sinuosos de forma correta). O custo de uma aresta representa o tempo necessário para atravessar um segmento. Por falta de dados precisos sobre o comportamento das vias, foi estimado o tempo necessário para atravessar a via da seguinte forma: tamanho do segmento / velocidade de 39,6 km/h. Essa velocidade é utilizada por ser uma estimativa da

velocidade média na cidade do Rio de Janeiro, baseado em notícias divulgadas em jornais.

As principais classes utilizadas, mostradas na Figura 3, são:

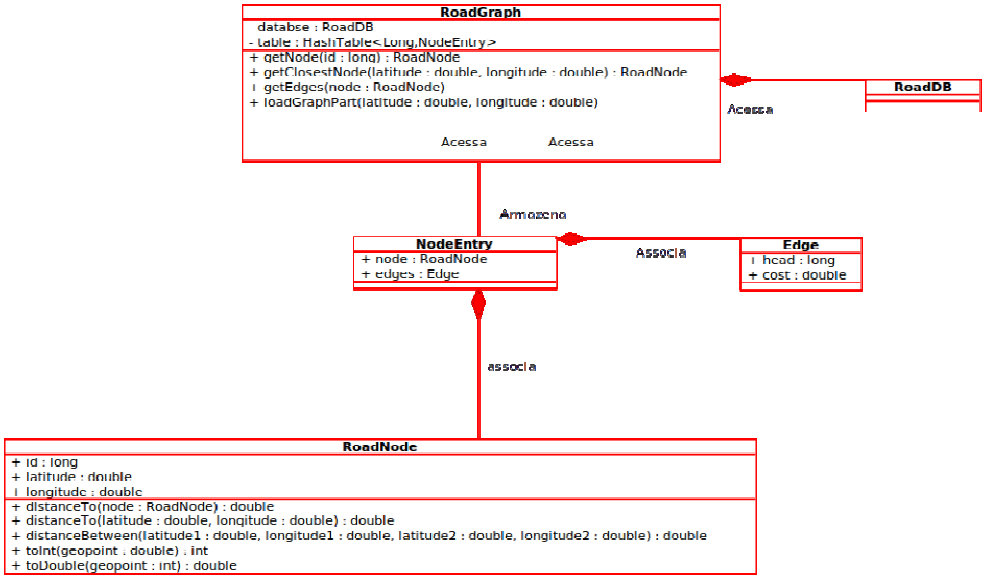


Figura 3: Diagrama de classes

RoadGraph – 'singleton' responsável pelas abstrações para converter as informações gravadas no banco de dados em um grafo em memória.

RoadNode - responsável pelas informações que envolvem apenas nós (por exemplo, distância baseada na latitude e longitude).

Edge - Classe que guarda o custo e o destino de uma aresta.

Gerenciamento dos Dados da Região – importação e gerencia os dados relativos à informação da malha rodoviária da região.

Estimativa do Custo da Rota – As informações são guardadas do inicio até o fim da viagem. A sequência de nós visitados e o horário em que foram visitados são armazenados. O tempo de viagem entre nós é calculado como a diferença entre os instantes de visita. O custo (em tempo) pode ser então atualizado no banco de dados, assim como o tempo da última atualização.

3.3. Cálculo dos menores caminhos

Foram utilizados dois algoritmos de busca em grafos: O algoritmo Dijkstra e o algoritmo A* simples, ambos explicados detalhadamente em (Cormen, 2011). Ambos foram desenvolvidos como 'threads' e após o término de suas execuções, devolvem o resultado ao processo original para serem exibidos.

3.4. Visualização

Com o resultado obtido, a rota alternativa é mostrada, como exemplificado pela Figura 4. A visualização é feita utilizando a API do Google Maps para Android.

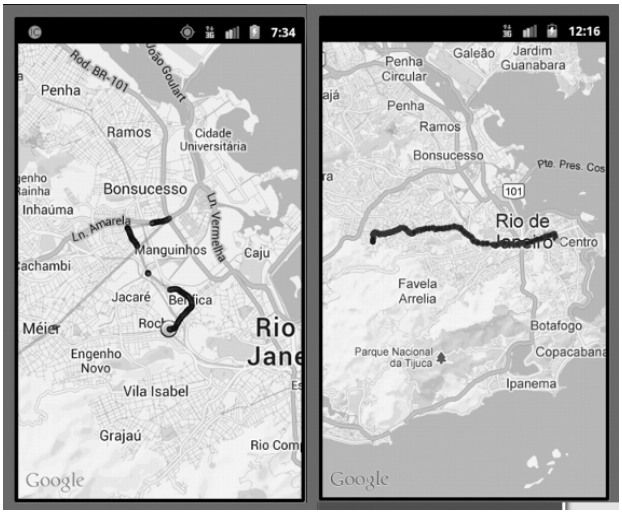


Figura 4: Esquerda: a coleta de dados da viagem. Direita: a resposta de recomendação

4. Avaliação

Para esta avaliação utilizamos um telefone celular Sony Ericsson Xperia X8, processador ARM6 com 600Mhz e 48Mb de memória virtual para a máquina Java e rotas entre diversos pontos da cidade do Rio de Janeiro.

Nos gráficos abaixo é mostrada a relação da distância entre nós em quilômetros e tempo de execução em segundos para os algoritmos A* (Figura 5) e Dijkstra (Figura 6). As rotas foram geradas aleatoriamente.

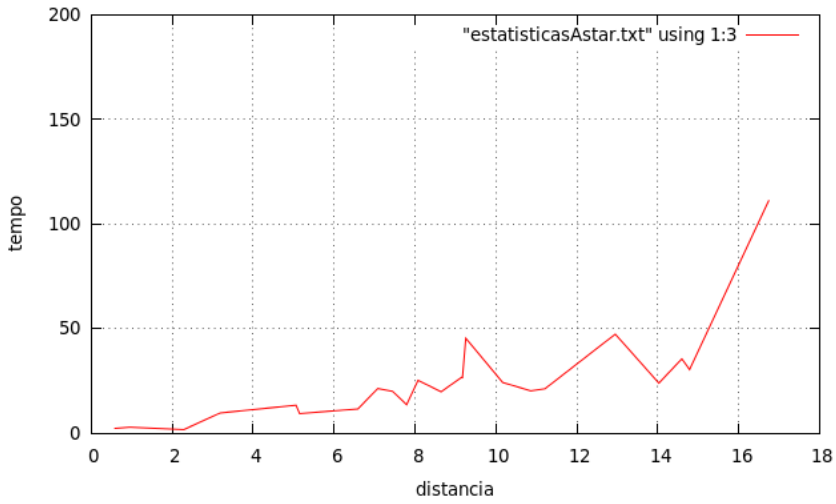


Figura 5: Desempenho A*

Para efeito de comparação, o software Waze, que utiliza conexão com servidor para obter sua recomendação de rota possui tempo independente da distância. O gargalo da resposta do tempo, neste caso, se encontra na rede. Mantendo origem e destinos iguais, a demora na resposta variou de 1 a 5 segundos, em redes Wifi e 3G. Por ser um produto fechado, utilizou-se um cronômetro para realizar a medição, podendo trazer imprecisão na medida de tempo.

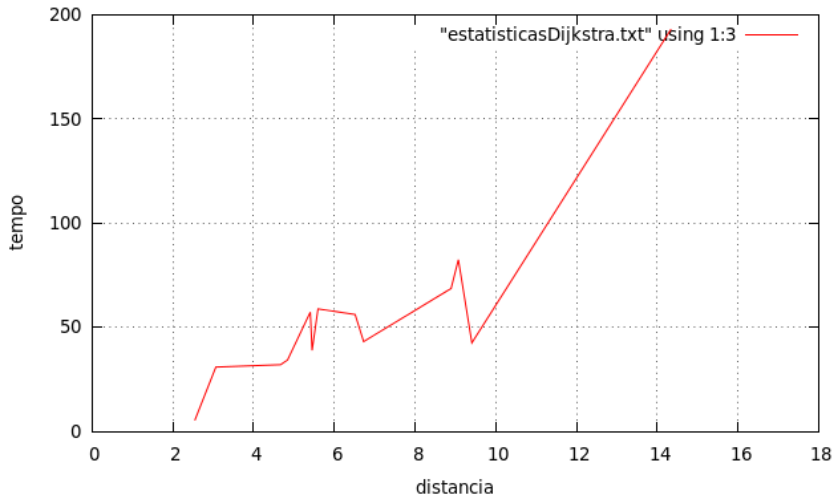


Figura 6: Desempenho Dijkstra

5. Conclusão

Este artigo descreveu o Hermes, um sistema para recomendação de rotas em dispositivos móveis. A execução totalmente local é possível, mas ainda há alguns desafios a serem superados como o consumo de memória e tempo de execução. Além disso, utilizaremos outros tipos de custos para as arestas, como a periculosidade da área, custo monetário (pedágios, por exemplo), acidentes e probabilidade de congestionamento, integrando o Hermes com outros ambientes que possam prover tais informações, como (Lauand, 2013). Outro objetivo é obter heurísticas mais precisas.

Agradecimentos. Ao CNPq e à FAPERJ.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to Algorithms. MIT Press and McGraw-Hill, 2nd edition.

Ding, Y., Wang, C., Xiao,L. (2007). A static-node assisted adaptive routing protocol in vehicular networks. In *ACM VANET*.

Jerbi, M., Senouci, S., Doudane, Y., and Beylot, A. L. (2008). Geo-localized virtual infrastructure for urban vehicular networks. In *8th International Conference on ITS Telecommunications*, pages 305–310.

Lauand, B., Oliveira, J. (2013). TweeTraffic: ferramenta de análise das condições de trânsito baseado nas informações do Twitter. In *BraSNAM*.

Schultes, D. (2008). Route planning in road networks. In *Ausgezeichnete Informatikdissertationen*, pages 271–280.

Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., and Eriksson, J. (2009). Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys ’09, pages 85–98, New York, NY, USA. ACM.

QualiDados: Jogo Didático para Consolidação de Conceitos de Qualidade de Dados*

Geovane F. Piccinin, Salatiel R. Santos, Suelen L. Romano, Mirella M. Moro

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Belo Horizonte, MG

{geovanepcc,salatiel.ribeiro,suelen.loiola,mirella}@dcc.ufmg.br

Abstract. *Data from real systems are often dirty, inconsistent, duplicated, inaccurate, incomplete or outdated. These problems generate incorrect results and analyzes, putting companies at risk of losing money, reputation and customers. The solution is then to count on data quality management concepts. This paper overviews such concepts and introduces a prototype for an educational game that simulates a data quality management environment.*

Resumo. *Sistemas reais frequentemente apresentam dados sujos, inconsistentes, duplicados, imprecisos, incompletos ou desatualizados. Tais problemas geram resultados e análises incorretos, criando risco de perda de receitas, credibilidade e clientes. Uma solução é utilizar conceitos de gerência de qualidade de dados. Este trabalho resume tais conceitos e introduz o protótipo de um jogo didático que simula um ambiente de gerência de qualidade de dados.*

1. Introdução

Sistemas reais frequentemente apresentam dados sujos, o que pode comprometer a confiabilidade dos dados e levar a erros. Estima-se que, em média, empresas possuem entre 1% e 5% de erro em seus dados e, em algumas delas, essa taxa pode chegar a 30%. Tais taxas elevadas podem prejudicar os sistemas de maneiras diferentes. Por exemplo, na maioria dos projetos de Data Warehouse, o processo de limpeza dos dados é responsável por 30% a 80% do tempo de desenvolvimento e do orçamento [Fan and Geerts 2012].

Dados sujos, especialmente os duplicados, custam à economia dos EUA aproximadamente três trilhões a cada ano. De acordo com *Larry English*, entre 15% e 20% do orçamento operacional de uma empresa pode ser desperdiçado devido a dados sujos. Isso demonstra a necessidade de melhorar métodos que visam detectar e tratar dados inconsistentes, incompletos, imprecisos, desatualizados e duplicados. Em resumo, dados sujos são responsáveis por prejuízos financeiros e redução da produtividade, devido ao tempo gasto na adequação desses dados.

Aprender a lidar com tais dados é fundamental para assegurar a qualidade dos mesmos. Com diversos conceitos ligados à qualidade de dados, o aprendizado dos mesmos bem como a correta utilização das ferramentas relacionadas são desafios para estudantes e profissionais. Para auxiliar em tal tarefa, este artigo apresenta o *QualiDados*, um jogo didático com situações de baixa qualidade de dados que podem ser corrigidas com a utilização de certas ferramentas. A ideia de utilizar jogos didáticos como auxílio

*Um vídeo de demonstração da ferramenta está disponível em http://sbbd2013.cin.ufpe.br/screencasts.

educacional não é nova. De fato, nós pensamos e entendemos melhor quando podemos imaginar um problema e nos preparar para respondê-lo [Gee 2003]. Nesse contexto, jogos são perfeitos para apresentar situações similares através de simulação, fornecendo então a oportunidade para pensar, entender, preparar e executar as devidas ações necessárias.

A detecção e o tratamento de dados sujos não são tarefas triviais devido ao caráter subjetivo. Neste artigo, são apresentados alguns conceitos básicos sobre qualidade de dados na Seção 2, e o caráter subjetivo de tais conceitos é discutido na Seção 3. Para facilitar o aprendizado e simular situações reais de controle de qualidade de dados, a Seção 4 introduz o *QualiDados*, protótipo de um jogo didático.

2. Conceitos Básicos sobre Qualidade de Dados

Esta seção resume alguns dos principais conceitos sobre qualidade de dados. Tais conceitos são explorados nas diversas características do jogo *QualiDados*.

Consistência de Dados. A consistência de dados se refere à validade e integridade da representação do dado no mundo real. Bancos de dados relacionais costumam apresentar inconsistência dentro de uma mesma tupla, entre diferentes tuplas em uma mesma relação e até mesmo entre diferentes tuplas de relações diferentes. Portanto não se trata apenas de restrição de integridade, é preciso levar em conta a semântica dos dados. Conforme o exemplo mostrado na Tabela 1, existem conflitos dentro de uma mesma tupla e entre tuplas diferentes. Na série de livros "Game of Thrones", homens da Patrulha da Noite abdicam da família, não podendo ser casados como informado na tupla $t1$. Igualmente, o símbolo da família Lannister é um leão, e não um lobo como informado na tupla $t2$, que também entra em atrito com o valor correto armazenado na tupla $t3$.

Tabela 1. Exemplo sobre consistência de dados

	nome	**sobren**	**símbolo**	**posição**	**civil**
t1	John	Snow	–	patrulha	casado
t2	Jaime	Lannister	lobo	escudeiro	solteiro
t3	Jaime	Lannister	leão	cavaleiro	solteiro
t4	Aria	Stark	lobo	–	solteira
t5	Aria	Stark	lobo	herdeira	solteira
t6	Aria	Bolton	lobo	herdeira	casada

Eliminação de Duplicatas. A eliminação de duplicatas identifica tuplas em uma ou mais relações que se referem à mesma entidade do mundo real. Por exemplo, considere as tuplas $t4$, $t5$ e $t6$ da Tabela 1: uma consulta para descobrir os dados da Aria retorna três tuplas. Porém, é possível que todas essas tuplas se refiram à mesma pessoa. Para descobrir se é o caso, uma alternativa é verificar se existe outro dado indicando que *Aria Stark* e *Aria Bolton* têm, por exemplo, os mesmos genitores, o que indicaria que são a mesma pessoa. Tal averiguação é fundamental para realizar não apenas a limpeza de dados mas também a integração de fontes de dados diferentes [Carvalho et al 2011]. Nesse caso, os dados são coletados a partir de diferentes bases com o intuito de reunir informações sobre uma mesma entidade. Como essa entidade pode estar representada de maneiras diferentes, ao integrar é necessário eliminar duplicatas também.

Precisão. A precisão dos dados se refere a quão próxima a representação no banco de dados está do valor da entidade no mundo real. Por exemplo, considere a Tabela 2 cujas tuplas especificam o nome, sobrenome, idade, peso e estado civil de uma pessoa, e $t1$ apresenta os dados verdadeiros da personagem Daenerys Targaryen, ao iniciar a série de livros. Pode-se concluir que $t2(idade, altura)$ é mais preciso do que $t3(idade, altura)$, já que tais valores estão mais próximos aos valores verdadeiros; enquanto $t3(nome, civil)$ é mais preciso do que $t2(nome, civil)$. Entretanto é mais desafiador determinar a precisão relativa de $t2$ e $t3$ quando *não* existe $t1$ como referência. Porém, mesmo sob essas circunstâncias, é possível descobrir que em certos atributos o valor em uma tupla é mais preciso do que em outra através da análise semântica dos dados e outras informações.

Tabela 2. Exemplo sobre precisão de dados

	nome	sobren	idade	altura	civil
t1	Daenerys	Targaryen	13	1,50	solteira
t2	D.	Targaryen	13	1,49	casada
t3	Daenerys	Targaryen	31	1,40	solteira

Completude da Informação. Um banco deve apresentar dados completas em resposta a uma consulta. Para uma base de dados D e uma consulta C, é necessário identificar quando C retornará um resultado correto usando-se apenas os dados fornecidos por D. Se D possui informação incompleta, é provável que retorne um resultado impreciso ou até mesmo incorreto [Fan and Geerts 2012].

Atualidade. Esse princípio identifica se as entidades apresentam valores atuais e retornam valores atuais para as consultas. Esse é um problema que, em teoria, poderia ser resolvido facilmente através da inserção de *timestamps* em todos os dados. No entanto, em bases de dados reais, os dados constantemente possuem *timestamps* imprecisas ou simplesmente não as possuem. Além disso é comum que dados sejam copiados ou importados de outras fontes, que nem sempre suportam o mesmo tipo de representação. Isso torna a tarefa de identificar os valores mais recentes mais difícil [Fan and Geerts 2012]. Por exemplo, para a Tabela 1, deseja-se descobrir o sobrenome e a posição de Aria. Após um processo de detecção de dados duplicados, descobriu-se que todas as três tuplas cujo nome é Aria representam a mesma pessoa. Se os dados não possuem *timestamp* fica difícil saber se o sobrenome atual de Aria é Stark ou Bolton, e se sua posição é nenhuma ou herdeira. Porém, mesmo sem um indício direto da atualidade do dado é possível descobrir qual é a tupla mais recente. A partir do contexto desses dados, é possível inferir que a posição atual de Mary é herdeira, pois normalmente as personagens passam de nada à herdeiras. De forma similar, pode-se deduzir que o sobrenome de Aria é Bolton, pois para esse valor o campo *civil* está definido como *casada*, ou seja, Aria é casada. Tal dedução considera o fato de que uma pessoa solteira pode mudar seu estado civil para casada, mas uma casada não muda seu estado para solteira, e sim para divorciada.

3. Qualidade de Dados: Aspectos Subjetivos

A seção anterior apresentou alguns conceitos relevante para analisar qualidade de dados. É importante notar que existem diversas interações entre tais conceitos. Por exemplo, a atualidade dos dados pode ser melhorada se mais informações temporais puderem ser

obtidas em um processo de melhoria da completude dos dados. Outro exemplo é que às vezes, para determinar qual dado é o mais atual, é preciso recorrer a um processo de identificação de duplicatas para saber quando várias tuplas se referem à mesma entidade. Tais interações, enfatizam o caráter *subjetivo* da qualidade de dados. Em muitos casos, a correção de uma informação depende de outros campos, outras tabelas e em alguns casos até mesmo da correção prévia de outros problemas.

Em várias situações, os bancos de dados tentam representar entidades muito complexas, onde cada entidade possui seus requisitos específicos de qualidade (precisão, representação, mecanismos de obtenção dos dados, etc). A construção do banco de dados envolve várias etapas, as quais podem ser críticas para a determinação da qualidade final dos dados. Essas etapas vão desde a coleta (por exemplo em pesquisas de campo, medições com instrumentos mecânicos ou digitais) até a apresentação final dos dados.

Nesse contexto, problemas de qualidade dos dados podem surgir em qualquer etapa. Exemplos de etapas e seus problemas incluem os seguintes:

- *Coleta dos dados* (amostragem) com falha dos instrumentos, erros de operação, dados sem qualidade;
- *Processamento dos dados amostrados* com procedimentos imprecisos, erros de processamento, falhas de processamento;
- *Representação* com a representação proposta distorcendo a realidade;
- *Apresentação* com a visualização gerada compromentendo a avaliação do usuário.

Portanto, não há uma definição genérica sobre qualidade de dados que norteie como o tratamento deve ser realizado. Há a certeza de que cada aplicação deve receber uma avaliação particular, e os métodos devem ser adequados ao objetivo dos usuários, a cada etapa de constituição da base de dados.

4. *QualiDados*: Definição e Características

Cabe ao analista de banco de dados definir processos e ferramentas adequados para manter os dados com a precisão, consistência e atualidade necessárias para o bom funcionamento do sistema. Conforme discutido nas seções anteriores, o caráter subjetivo da qualidade de dados e da necessidade de tomar decisões de acordo com o banco de dados e as características do sistema, é muito difícil que estudantes e profissionais consigam praticar tais conceitos sem auxílio devido.

Por outro lado, as teorias que unem educação ao entretenimento têm ganhado muita força nos últimos anos, apresentando uma tendência para um modelo de educação que emerge: a educação apoiada no uso de jogos digitais. De fato, projetistas de jogos e educadores argumentam que jogos capturam a atenção de seus jogadores, engajando-os em raciocínio e resolução de problemas complexos [Barab and Dede 2007]. Nesse contexto, o nosso objetivo é apresentar o protótipo de um jogo didático que simula diversas situações para as quais o jogador terá de avaliar a qualidade dos dados.

O propósito do jogo é fornecer um ambiente didático de competição onde os estudantes (e profissionais) tenham a oportunidade de praticar conceitos aprendidos na sala de aula ou na prática. O ambiente de competição é estimulante e incentiva os alunos a solidificar seu aprendizado e a buscar soluções para os problemas apresentados [Reis et al. 2012]. Além disso, jogos educacionais, incluindo os digitais estimulam o

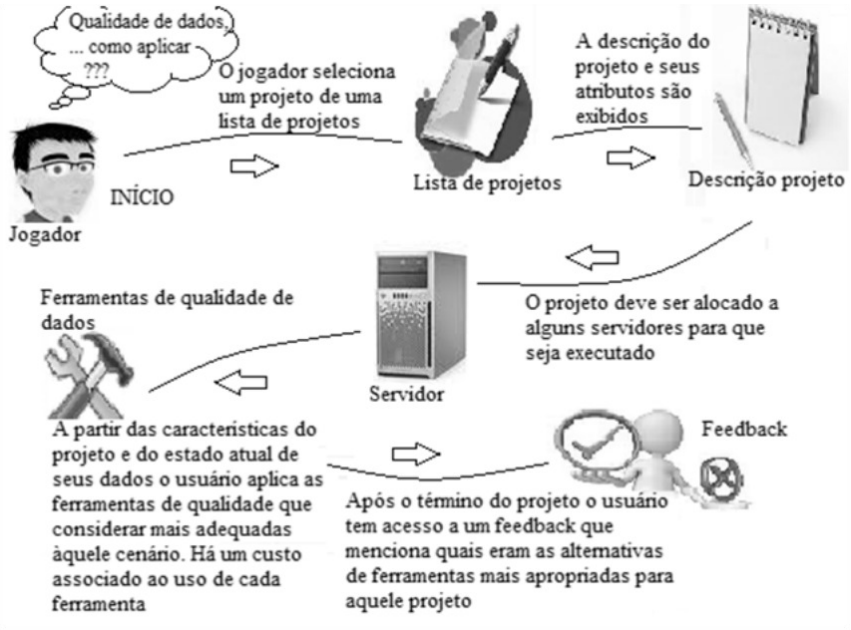


Figura 1. Fluxo do jogo didático QualiDados

desenvolvimento cognitivo, auxiliando na criação de estratégias para a solução de problemas [Hopf et al. 2007].

O jogo *QualiDados* simula uma empresa de TI na qual o jogador é o "analista de banco de dados". O jogador tem à sua disposição: *servidores*, os quais são alocados a uma lista de *projetos*, e *ferramentas* de qualidade de dados. Cada Projeto possui uma descrição com seus objetivos e atributos que revelam a qualidade de dados atual e a desejada. O jogador deve, então, selecionar um projeto, alocar servidores para a sua realização e aplicar as ferramentas de qualidade de dados para que atinja os objetivos pretendidos e ganhe os pontos. Dentre a lista de Ferramentas de qualidade de dados estão: backup, comparar com base externa, completar dados, criptografar, eliminar dados duplicados, inserir meta-dados, limpar dados sujos, notificar fonte de dados. Nessa perspectiva, o jogador é desafiado a escolher e aplicar as ferramentas de qualidade de dados mais adequadas ao seu projeto. Além de gerenciar os recursos para que otimize a quantidade de projetos realizados.

A Figura 1 ilustra o funcionamento do jogo, conforme explicado a seguir.

1. Início do jogo. Elementos disponibilizados ao jogador: lista de Servidores, lista de Projetos, Ferramentas de qualidade de dados, Saldo inicial;
2. Andamento do jogo. O jogador seleciona um Projeto para executar e aloca Servidores para realizarem o processamento do Projeto. Nessa etapa, o jogador verifica quais são as características e demandas do Projeto e aplica as Ferramentas de qualidade de dados que considerar adequadas.
3. Processamento dos pontos (ao final de uma rodada). Após alocar os Servidores e aplicar as Ferramentas de qualidade de dados sobre o Projeto é feita a contabilização dos pontos obtidos avaliando-se quais as Ferramentas aplicadas frente às características do Projeto.
4. Feedback. Após a conclusão de um Projeto torna-se disponível ao usuário um

relatório explicando quais eram as Ferramentas adequadas àquele Projeto. Esse relatório é fundamental no jogo porque ele será o gabarito do jogador para medir e qualificar sua atuação.

5. Ações que podem ser realizadas a qualquer momento do jogo. O jogador pode comprar novos Servidores para aumentar sua capacidade de processamento. Acessar o tutorial.
6. Tutorial. Acompanha o jogo com o objetivo de elucidar os princípios, fundamentos, regras e objetivos.
7. Editor de Projetos. A lista de Projetos disponibilizada no jogo é criada com um programa auxiliar e armazenada em um arquivo, o qual é carregado no início do jogo. O Programa auxiliar consiste em um formulário onde o usuário define todos os atributos da entidade Projeto: Título, Descrição, Valor do Pagamento, peso que será aplicado a cada quesito de qualidade de dados, Quantidade de dados a serem processados, etc. Com essa ferramenta pode-se criar qualquer instância de Projeto.

5. Conclusões

O objetivo do *QualiDados* é ser uma opção didática para interessados no tema qualidade de dados. Sua base está na simulação de um ambiente para praticar e desenvolver o senso crítico a respeito dos principais conceitos e ferramentas de qualidade de dados. O protótipo pode ser estendido para que permita a interação entre os jogadores e que ofereça mais desafios de gerenciamento, típicos de um ambiente informatizado, como problemas de segurança dos dados e problemas de hardware. Adquirindo essas novas características e funcionalidades, ele pode se tornar um recurso de aprendizagem mais robusto e que estimule, através da competitividade, o aperfeiçoamento teórico dos jogadores.

Agradecimentos. Projeto parcialmente financiado por CNPq.

Referências

Barab, S. and Dede, C. (2007). Games and Immersive Participatory Simulations for Science Education: An Emerging Type of Curricula. *Journal of Science Education and Technology*, 16(1):1–3.

Carvalho et al, A. P. (2011). Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries. *JIDM*, 2(3):289–304.

Fan, W. and Geerts, F. (2012). *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment*, 1(1):20.

Hopf, T., Falkembach, G. A. M., and Araújo, F. V. (2007). Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries. *Novas Tecnologias na Educação*, 5(2):289–304.

Reis, G. L., Souza, L. F. F., Carvalho, F. C. T., Abdalla Jr., M. A., Nepomuceno, E. G., Barroso, M. F. S., and Pereira, E. B. (2012). As Competições Universitárias e a Carreira Profissional do Aluno de Graduação: Um Estudo de Caso Sobre a Equipe UAIrobots-SEK. In *Workshop de Robótica Educacional*.