REx - NoSQL Redis Schema Extraction Module*

Angelo A. Frozza[1,2], Geomar A. Schreiner[1], Bruno R. L. Machado[1], Ronaldo dos S. Mello[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Campus Universitário Trindade – CP 476 – 88.040-900 – Florianópolis (SC), Brasil

[2]Instituto Federal Catarinense (IFC) - Campus Camboriú
Rua Joaquim Garcia, S/N – 88.340-055 – Camboriú (SC), Brasil

angelo.frozza@ifc.edu.br, geomarschreiner@gmail.com

brunoh.rafael.leal@gmail.com, r.mello@ufsc.br

Abstract. *This paper describes the REx (**R**edis Schema **Ex**traction) module, which allows schema generation for NoSQL key-value databases, and it is coupled to the JSON Schema Discovery tool. REx implements the first step (Generation of Raw Schemas) of the schema extraction process developed in JSON Schema Discovery. The extraction strategy accomplished by REx is the main contribution of this paper.*

Resumo. *Este artigo descreve o módulo REx (**R**edis Schema **Ex**traction), que permite a geração de esquemas para bancos de dados NoSQL chave-valor, sendo um componente da ferramenta JSON Schema Discovery. REx implementa a primeira etapa (Geração de Esquemas Brutos) do processo de extração de esquemas realizado pela JSON Schema Discovery. A estratégia de extração adotada pelo REx é inovadora na área de extração de esquemas de dados NoSQL.*

1. Introdução

Na atual era do Big Data há a produção de grandes volumes de dados, em uma velocidade muito alta, armazenados de forma distribuída e compartilhados em diferentes formatos por vários tipos de aplicação [Lomotey and Deters 2014]. Entretanto, bancos de dados relacionais (BDRs) não são adequados ao gerenciamento de Big Data, principalmente por causa da rigidez dos seus esquemas de dados [Chickerur et al. 2015]. Empresas como Google e Amazon foram pioneiras no desenvolvimento de novas tecnologias para o gerenciamento de Big Data, sendo uma destas famílias de tecnologias denominadas *NoSQL (Not-Only SQL)* [NoSQL 2019]. Sistemas de BD NoSQL são capazes de representar dados complexos, são escaláveis para gerenciar grandes conjuntos de dados, adotam modelos de dados não-relacionais e, geralmente, não exigem esquemas para os dados (*schemaless*) [Sadalage and Fowler 2013].

Apesar de o esquema não ser mandatório para BDs NoSQL, a validação de dados é um requisito importante para aplicações de Big Data com alguma consistência. Assim, o gerenciamento de esquemas NoSQL torna-se um tópico pertinente para a integração

*Este trabalho foi desenvolvido com o apoio de bolsa de IC (Ed. PROPESQ 01/2017 PIBIC/CNPq UFSC) e bolsas de doutorado (PRODOUTORAL-Ed. 231/2017/REITORIA/IFC CAPES e CAPES/UFSC).

de dados, interoperabilidade entre diferentes bases de dados (até mesmo entre diferentes modelos de BDs NoSQL) ou mesmo para validação e manutenção da integridade dos dados [Scavuzzo et al. 2014, Klettke et al. 2015, Ruiz et al. 2015]. Neste contexto, foi desenvolvida a ferramenta JSON *Schema Discovery* (JSD) [Frozza et al. 2018], que realiza a extração de esquemas de coleções de documentos JSON, que é o formato de armazenamento tipicamente adotado por BDs NoSQL que seguem o modelo de dados de documento. O processo de extração gera um único esquema que representa a estrutura completa da coleção no formato JSON *Schema*. Este artigo estende as funcionalidades da JSD apresentando um módulo denominado REx - NoSQL ***R**edis Schema **Ex**traction*, que dá suporte à extração de esquemas de BDs NoSQL baseados no modelo de dados chave-valor (*Redis*). Até onde se conhece, não há trabalhos relacionados propondo a extração de esquemas de BDs NoSQL chave-valor.

O restante deste artigo está organizado conforme segue. A Seção 2 apresenta algumas informações preliminares, a Seção 3 descreve o módulo REx e como ele está integrado à JSD e, por fim, as considerações finais encontram-se na Seção 4.

2. Preliminares

O padrão JSON (*JavaScript Object Notation*) destaca-se como modelo de dados de BD NoSQL de documentos [T. Bray 2014]. Ele permite a definição de objetos cujos atributos possuem domínios atômicos ou estruturados, além de fornecer uma representação de dados flexível, pois os atributos de um objeto JSON podem ser opcionais ou de valor múltiplo. Ainda, um atributo estruturado pode conter, de maneira recursiva, um conjunto de objetos aninhados, o que é adequado à representação de entidades complexas.

Outros modelos de dados NoSQL (chave-valor, colunar e grafo) podem ser representados em JSON, uma vez que esses modelos têm um poder de expressão inferior ou equivalente ao JSON [Sadalage and Fowler 2013]. Assim sendo, pode-se afirmar que JSON serve como um formato canônico [Sheth e Larson 1990] de representação de dados de BD NoSQL com modelos de dados heterogêneos. No caso específico do modelo de dados chave-valor, cada instância pode ser representada como um atributo JSON, sendo que a *chave* do dado é o nome do atributo e o *valor* do dado é o conteúdo do atributo. Também pode-se mapear a chave do dado para o ID de um documento JSON e o valor de dados pode ser deserializado e convertido em pares atributo-valor no corpo deste documento. Esta segunda estratégia é viável quando o valor do dado é um conteúdo estruturado.

JSD é uma ferramenta para extração de esquemas de coleções de documentos JSON. O processo de extração percorre os documentos JSON de uma coleção e analisa suas propriedades para identificar o esquema bruto de cada documento. O esquema bruto é um documento JSON com a mesma estrutura hierárquica do documento JSON original, no entanto, os valores dos atributos são substituídos pelos tipos de dados correspondentes (Etapa 1 - Geração do esquema bruto). Em seguida, JSD unifica esses esquemas brutos e gera um único esquema, no formato JSON *Schema*, o qual representa a coleção de documentos como um todo (Etapa 2 - Agrupamento e unificação de esquemas brutos). Para unificar as informações estruturais de todos os esquemas brutos, uma estrutura de dados em árvore é definida, a qual mantém propriedades como: objetos aninhados, *arrays*, tipos de dados JSON primitivos ou JSON estendido. A JSD e sua abordagem de extração foi alvo de um trabalho anterior realizado por este grupo de pesquisa [Frozza et al. 2018].

A JSD foi originalmente concebida para a extração de esquemas do *MongoDB*, um BD NoSQL orientado a documentos. Recentemente, decidiu-se estender a ferramenta para lidar com a extração de esquemas de outros modelos de dados NoSQL. Para tanto, foram necessárias adaptações na sua Etapa 1. Essa adaptação resultou no módulo REx, para o caso do modelo de dados chave-valor, que é descrito na próxima seção.

3. O módulo REx

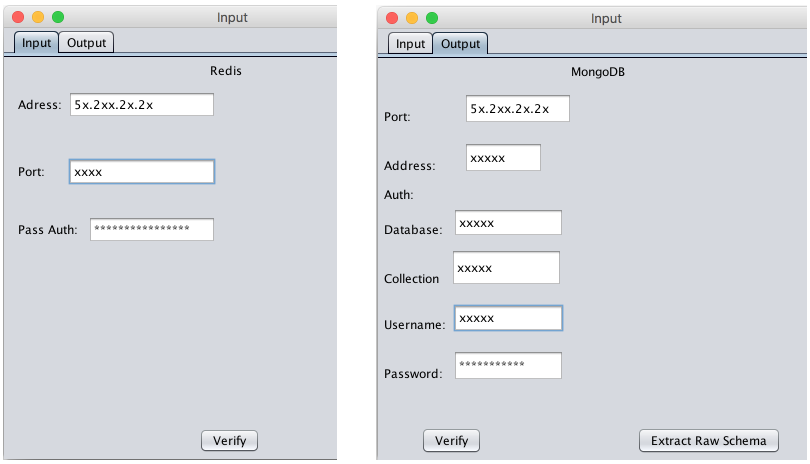
O módulo **REx** (*NoSQL **R**edis Schema **Ex**traction*)[1] é responsável pela geração do esquema bruto, no formato JSON, para instâncias de dados de um BD NoSQL chave-valor (no caso, o BD *Redis*). Para tanto, considerando que uma instância de dados em um BD chave-valor possui apenas os conceitos *key* e *value*, de acordo com o conteúdo do campo *value* a geração do esquema bruto prevê dois casos possíveis:

(a) **Documento JSON** - o conteúdo do campo *value* está estruturado no formato JSON. Neste caso, um esquema bruto para essa instância é criado considerando a estrutura hierárquica dos atributos do documento JSON. Atualmente, o REx consegue identificar seis tipos de dados JSON: *Object*, *Array*, *Number*, *Boolean*, *String* e *null*.

(b) **Sequência de *bytes*** - o conteúdo do campo *value* não é estruturado (conteúdo binário). Neste caso, o esquema bruto da instância tem apenas dois atributos: a chave *key* e o valor *value*, sendo ambos do tipo *string*.

O REx disponibiliza uma interface com o usuário, a qual é mostrada na Figura 1. A aba *Input* solicita os dados de conexão com um BD *Redis* de origem. Já a aba *Output* solicita os dados de conexão com um BD *MongoDB* no qual são armazenados os esquemas brutos produzidos. Após as informações de conexão serem verificadas, o botão *"Extract Raw Schema"* fica habilitado para iniciar o processamento.

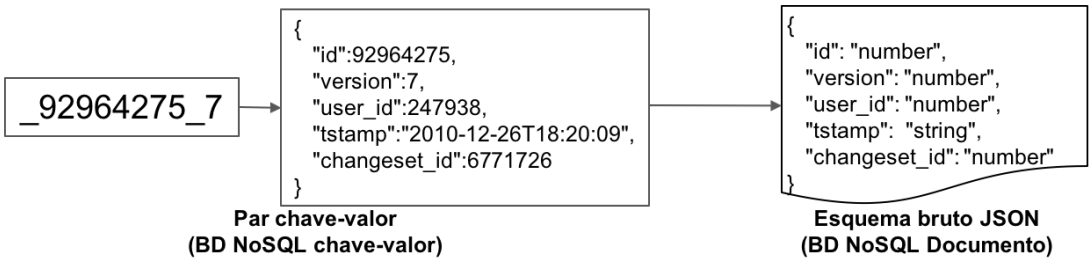
Figura 1. Interfaces do REx.



O processo de extração adotado pelo REx é o seguinte: *(i)* Carrega-se uma instância de dados do *Redis*; *(ii)* Verifica-se o conteúdo do campo *value* e procede-se a extração para cada um dos dois casos descritos anteriormente; *(iii)* Armazena-se o documento do esquema bruto gerado no *MongoDB*; *(iv)* Retorna-se ao passo *(i)*. A Figura 2 exemplifica a criação do esquema bruto a partir de um documento JSON armazenado

[1]Disponível em http://lisa.inf.ufsc.br/wiki/index.php/REx

Figura 2. Mapeamento de uma instância chave-valor para um esquema bruto.



como valor em uma instância chave-valor. Uma vez que todas as instâncias *Redis* foram processadas e foram criados os respectivos esquemas brutos, a JSD executa as suas demais etapas visando gerar o esquema único em JSON *Schema*.

4. Considerações finais

Este artigo apresenta o módulo **REx**, desenvolvido como um componente da ferramenta JSD para a extração de esquemas brutos no formato JSON *Schema* a partir de instâncias de dados presentes em BDs NoSQL chave-valor (*Redis*). Uma avaliação preliminar do funcionamento do REx concluiu que o mesmo foi capaz de atender plenamente os dois casos previstos na seção 3, desde que, no primeiro caso, a estrutura JSON esteja bem formada. Trabalhos futuros incluem o suporte à conexão com outros BDs NoSQL chave-valor, a capacidade de identificar tipos de dados geográficos e a criação de módulos para extrair esquemas dos modelos NoSQL colunar e de grafos.

Referências

Chickerur, S., Goudar, A., and Kinnerkar, A. (2015). Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications. In *8th Int. Conf. on Advanced Software Engineering and Its Applications (ASEA)*, pages 41–47.

Frozza, A. A., Mello, R. d. S., and da Costa, F. d. S. (2018). An Approach for Schema Extraction of JSON and Extended JSON Document Collections. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 356–363. IEEE.

Klettke, M., Storl, U., and Scherzinger, S. (2015). Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores. In *BTW*, volume 241 of *LNI*.

Lomotey, R. K. and Deters, R. (2014). Towards Knowledge Discovery in Big Data. In *8th IEEE International Symposium on Service Oriented System Engineering, SOSE 2014*.

Ruiz, D. S., Morales, S. F., and Molina, J. G. (2015). Inferring Versioned Schemas from NoSQL Databases and its Applications. *LNCS*, 9381:467–480.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL Distilled: a Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley, 1st edition.

Scavuzzo, M., Nitto, E. D., and Ceri, S. (2014). Interoperable Data Migration Between NoSQL Columnar Databases. In *18th IEEE Int. Enterprise Distributed Object Computing Conference, EDOC, Ulm, Germany, September 1-2*, pages 154–162.

T. Bray, E. (2014). The JavaScript Object Notation (JSON) Data Interchange Format - RFC 7159.