UNIVERSIDADE FEDERAL DE SANTA CATARINA

HBASI - HBASE SCHEMA INFERENCE TOOL: UMA FERRAMENTA PARA EXTRAÇÃO DE ESQUEMAS DE BANCOS DE DADOS NOSQL COLUNARES

Eduardo Dias Defreyn

Florianópolis - SC

2019/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

HBASI - HBASE SCHEMA INFERENCE TOOL: UMA FERRAMENTA PARA EXTRAÇÃO DE ESQUEMAS DE BANCOS DE DADOS NOSQL COLUNARES

Eduardo Dias Defreyn

Trabalho de conclusão de curso
apresentado como parte dos requisitos
para obtenção do grau de Bacharel em
Ciências da Computação.
Orientador: Profº Ronaldo dos Santos
Mello, Dr.
Coorientador: Profº Angelo Augusto Frozza,
M.Sc.

Florianópolis - SC

2019/1

Eduardo Dias Defreyn

HBASI - HBASE SCHEMA INFERENCE TOOL: UMA FERRAMENTA PARA EXTRAÇÃO DE ESQUEMAS DE BANCOS DE DADOS NOSQL COLUNARES

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Profº Ronaldo dos Santos Mello, Dr.
Orientador

Profº Angelo Augusto Frozza, M.Sc.
Coorientador

Banca Examinadora:

Profª Carina Friedrich Dorneles, Drª

Profº Renato Fileto, Dr.

RESUMO

Bancos de Dados (BDs) NoSQL estão ganhando espaço no mercado por serem adequados ao processamento de grande volume de dados e não exigirem a definição de um esquema para serem usados. Embora os BDs NoSQL não obriguem a existência de um esquema *a priori*, geralmente existe alguma estrutura implícita no BD para facilitar a manipulação adequada dos dados. O conhecimento do esquema dos dados é importante para soluções que desejam realizar integração de dados ou mesmo busca e análise de diversas fontes de dados em um mesmo domínio de aplicação. Assim sendo, este trabalho apresenta uma ferramenta para inferência de esquema para BDs NoSQL colunares, com foco no Sistema de Gerência de BD (SGBD) HBase. O HBase foi escolhido por ser um dos BDs colunares mais populares que se declara *schemaless*. A ferramenta, intitulada HBaSI (*HBase Schema Inference*), infere o esquema de um BD HBase e o apresenta no formato de um JSON *Schema*. Existem alguns trabalhos que inferem esquemas para o HBase. Entretanto, eles apresentam processos mais complexos e não possuem interface gráfica com o usuário. Outra contribuição do trabalho é a forma como é feita a inferência dos tipos de dados das colunas, pois o HBase armazena apenas *arrays* de *bytes* como conteúdo das colunas.

Palavras-chave: NoSQL, HBase, BD colunar, Extração de esquema, Inferência de esquema.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Com a evolução da Internet e da computação, a demanda por dados e seus meios de acesso tem evoluído. Os tradicionais bancos de dados (BDs) relacionais, baseados em tabelas e tuplas, acabam por satisfazer as aplicações onde os dados se adequam a estrutura de relações entre dados simples (SADALAGE; FOWLER, 2012). Contudo existem aplicações que necessitam de maior flexibilidade sobre a estrutura dos dados, podendo conter estruturas aninhadas complexas assim como se aproveitar da estrutura dos dados em memória.

Para suprir as necessidades destas aplicações surgiram novos modelos de BDs denominados NoSQL. O termo surgiu inicialmente como um BD relacional na década de 90 criado por *Carlo Strozzi* que não utilizava SQL (SADALAGE; FOWLER, 2012). Ainda que este seja seu primeiro registro, este trabalho adota o conceito proposto por *Eric Evans*, ao organizar um encontro em 2009 com o intuito de apresentar trabalhos sobre BDs que não precisassem estar ligados a BDs convencionais, especialmente por existirem diversos projetos de BDs não convencionais inspirados em *BigTable* (CHANG et al., 2008), da Google, e *Dynamo*, da Amazon (SADALAGE; FOWLER, 2012). Os BDs NoSQL adotam modelos de dados que podem ser agrupados em quatro categorias: chave-valor, documento, colunar e grafos (SADALAGE; FOWLER, 2012).

Uma das limitações do modelo relacional é que antes de qualquer operação ser executada sobre os dados, primeiramente deve existir

uma estrutura de tabelas previamente definida, que leva em conta os requisitos de dados da aplicação. A isto é dado o nome de *esquema* (SADALAGE; FOWLER, 2012). Já os BDs NoSQL geralmente são *schemaless*, ou seja, não exigem um esquema prévio para o armazenamento de dados (SADALAGE; FOWLER, 2012; RUIZ; MORALES; MOLINA, 2015). A ausência de esquema torna muito mais flexível o armazenamento de dados, além de que, em situações de dados não uniformes, o espaço ocupado por cada conjunto de campos é apenas o necessário, sem precisar alocar espaço para os campos vazios (SADALAGE; FOWLER, 2012; HAN et al., 2011).

Ainda que não seja necessária a criação de um esquema explícito, geralmente existe algum esquema implícito nos dados que é conhecido pela aplicação que os manipula (RUIZ; MORALES; MOLINA, 2015). Sabendo-se que existe um esquema para os dados deve então ser possível explicitar este esquema.

Como o esquema é conhecido apenas pela aplicação e não pelo BD existem duas alternativas para inferir o esquema, fazer a análise sobre a aplicação, o que dificulta a automação, ou fazer a análise sobre o BD, na qual é possível explorar as características da arquitetura, da implementação e do modelo de dados porém, por estar em mais baixo nível, a quantidade de informação sobre o esquema é menor. No caso do HBase, um BD colunar, para lidar bem com qualquer tipo de dado os dados são armazenados em binário e por causa disso perde-se informações relevantes sobre os dados, como o tipo, que dificultam a obtenção do esquema sendo essa a maior dificuldade para a inferência,

pois a estrutura pode ser facilmente obtida.

1.1 MOTIVAÇÃO

Conforme mencionado anteriormente, uma das características mais atrativas dos BDs NoSQL é a ausência de um esquema explícito, pois oferece mais flexibilidade no armazenamento e menos *overhead* com garantia de consistência do esquema (SADALAGE; FOWLER, 2012; RUIZ; MORALES; MOLINA, 2015). Mesmo assim, a ausência de esquemas explícitos gera problemas, por exemplo, quando a estrutura do BD é relevante para aplicações que realizam tarefas de integração ou análise de dados e, em função disso, necessitam executar tarefas de extração de esquemas ou engenharia reversa de fontes de dados (SADALAGE; FOWLER, 2012; KLETTKE; STöRL; SCHERZINGER, 2015).

Com isso, pode-se afirmar que mesmo em BDs NoSQL sem esquema fixo, a estrutura dos dados é essencial para a manipulação adequada de dados por aplicações que realizam as tarefas supracitadas (KLETTKE; STöRL; SCHERZINGER, 2015).

Na literatura existem propostas para inferência de esquemas principalmente de BDs NoSQL documento (RUIZ; MORALES; MOLINA, 2015), uma vez que eles apresentam estruturas de dados mais complexas. Entretanto, há uma carência de soluções para inferência de esquemas para BDs colunares, que também podem apresentar estruturas bastante heterogêneas compostas por agrupamentos de colunas diferentes.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem por objetivo desenvolver uma ferramenta que realiza a inferência de esquema a partir de um BD NoSQL colunar HBase e sua representação na recomendação JSON *Schema*.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são os seguintes:

- Propor um processo para inferência de esquemas em BDs NoSQL Colunares;

- Propor um método para inferir os tipos de dados no BD NoSQL HBase;

1.3 JUSTIFICATIVA

O surgimento dos BDs NoSQL tem aumentado o número de aplicações que os utilizam. No entanto, a falta de esquemas explícitos é um fator limitante para aplicações que lidam com processos de integração e análise dos dados. Existem SGBDs NoSQL colunares que geram e mantêm esquemas, por exemplo, o Cassandra e o Apache Phoenix. Mesmo assim, não existe uma padronização para a representação destes esquemas. Por outro lado, outros SGBDs NoSQL colunares não oferecem recursos para a definição de esquemas, como é o caso do HBase

(totalmente *schemaless*). Por esse motivo, ele foi escolhido como o foco deste trabalho.

Desta forma, este trabalho visa inferir o esquema de um BD colunar mantido no HBase, e representá-lo em JSON *Schema* (JSON-SCHEMA, 2018). JSON *Schema* é uma recomendação para a definição esquemas em JSON (*JavaScript Object Notation*) que, por sua vez, é amplamente utilizado como formato para intercâmbio de dados (IETF, 2017).

1.4 METODOLOGIA

Como etapa inicial do trabalho, foi feito o levantamento de dados do estado da arte, através de livros, artigos e publicações pertinentes ao assunto "inferência/extração de esquemas em BDs NoSQL", especialmente sobre BDs colunares. Com isso, foi feito um estudo sobre as tecnologias e técnicas envolvidas no processo de inferência de esquema.

Na segunda etapa foi feita a análise dos trabalhos relacionados provenientes da primeira etapa, de modo a obter melhor entendimento sobre as características e requisitos de uma ferramenta de inferência.

A terceira etapa, consiste na definição da ferramenta proposta, com arquitetura, requisitos e tecnologias necessárias à sua implementação. Por fim, na quarta etapa, foi realizada uma avaliação dos resultados obtidos e uma análise comparativa entre os trabalhos relacionados e a ferramenta desenvolvida.

1.5 ESTRUTURA DO TRABALHO

Este trabalho está organizado em mais cinco capítulos. O segundo capítulo descreve a fundamentação teórica necessária para o desenvolvimento do trabalho. No terceiro capítulo são descritos e analisados os trabalhos correlatos. No quarto capítulo é descrito o projeto da ferramenta, incluindo as tecnologias utilizadas e processo de inferência. No quinto capítulo é apresentada uma avaliação da ferramenta e uma breve comparação com os trabalhos relacionados. Por fim, o sexto e último capítulo apresenta as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para a compreensão deste trabalho.

2.1 NOSQL

Com o desenvolvimento da Web 2.0 e com o aumento no número de usuários e serviços, tornaram-se perceptíveis as limitações dos BDs relacionais. Para contornar estas limitações, surgiu uma nova família de SGBDs, denominados NoSQL (SADALAGE; FOWLER, 2012; TUDORICA; BUCUR, 2011). Um BD NoSQL pode ser definido como um BD distribuído que normalmente não possui esquema fixo, evitam operações de junção, são escaláveis, nem sempre apresentam uma interface de acesso SQL e tendem a ser de código aberto (TUDORICA; BUCUR, 2011). Ainda que bastante divergentes, BDs relacionais e NoSQL não são opostos.

Tendo em vista que BDs NoSQL tendem a serem distribuídos o teorema CAP (*Consistency, Availability, Partition tolerance*), apresentado por Brewer (2000) e comprovado posteriormente por Lynch e Gilbert (2002), afirma que em um sistema distribuído existem três características básicas, sendo elas: consistência, disponibilidade e tolerância a particionamento, mas apenas duas podem ser garantidas simultaneamente (HAN et al., 2011; SADALAGE; FOWLER, 2012; CATTELL,

2011).

Considerando o teorema CAP, normalmente a consistência não é priorizada, ainda que seja balanceada com a disponibilidade, visto que a tolerância ao particionamento é essencial em BDs NoSQL (SADALAGE; FOWLER, 2012; CATTELL, 2011). Como a consistência não é priorizada, as características ACID são perdidas e as características BASE se adéquam mais à eventual consistência das escritas, sem limitar as leituras.

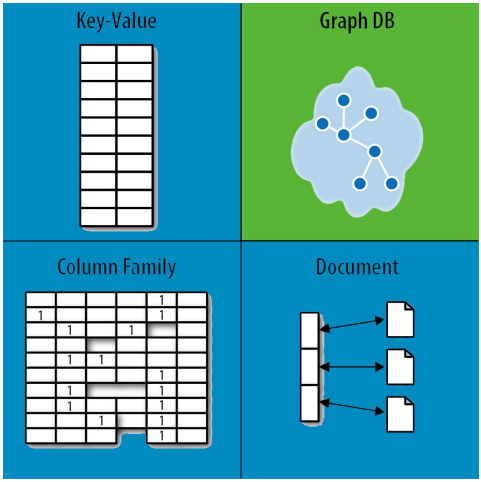
2.1.1 Modelos de dados

Um modelo de dados pode ser definido como o modo que se percebe e se manipula os dados (SADALAGE; FOWLER, 2012). O modelo mais comum ainda hoje é o modelo relacional, que pode ser descrito como um conjunto de tabelas, construídas através de relações entre as linhas, também chamadas de *tuplas*, que representam as entidades, e cada coluna representa um atributo da entidade. Estas entidades são distribuídas e se associam através de junções entre tabelas.

Já no contexto de BDs NoSQL, o modelo de dados de alto nível mais adotado é o *modelo de agregados*, ainda que cada uma das quatro categorias adote uma implementação diferente dele, conforme mostrado na Figura 1.

Diferente do modelo relacional, que não permite armazenar tuplas dentro de outras tuplas, o modelo orientado a agregados reconhece que podem existir dados mais complexos que apenas tuplas, e permite que estruturas, como listas ou tuplas, sejam aninhadas (SADALAGE;

Figura 1 – Principais modelos de dados de BDs NoSQL



Fonte: neo4j (2015)

FOWLER, 2012). Deste modo, a consistência entre os dados aninhados pode ser atômica. Esse modelo de dados é utilizado pelos BDs NoSQL chave-valor, documento e colunar, sendo que, no primeiro caso, um conteúdo complexo mantido no BD é de conhecimento apenas da aplicação. Os quatro modelos de dados NoSQL, dos quais derivam os principais tipos de BDs, são detalhados a seguir.

2.1.1.1 Banco de dados chave-valor

O modelo chave-valor é o mais simples dentre os modelos NoSQL. Ele é similar a um sistema de *cache* em memória no qual todos os dados são indexados apenas por uma chave única (CATTELL, 2011) e normalmente o SGBD não tem controle sobre seu conteúdo (simples ou

complexo) para realizar operações, ou seja, o conteúdo é uma "caixa preta" para o SGBD (SADALAGE; FOWLER, 2012).

Por não fazer operações sobre o conteúdo dos dados, as operações sobre esse modelo são simples e rápidas, por exemplo, armazenar (*put*) e recuperar (*get*) com base em uma chave. Os BDs chave-valor que mais se destacam são: Redis, Riak, Projeto Voldemort, MemcachedDB, BerkeleyDB, Scalaris e DynamoDB (SADALAGE; FOWLER, 2012; CATTELL, 2011; SCHERZINGER; KLETTKE; STöRL, 2013).

2.1.1.2 Banco de dados de documento

Esses BDs se assemelham muito aos BDs chave-valor, tendo uma chave para identificar um documento. Os documentos têm um formato padrão, como JSON ou XML (*eXtensible Markup Language*), e por conta disso, é possível manipular seu conteúdo e realizar consultas (SADALAGE; FOWLER, 2012).

BDs de documentos não são voltados para leituras e escritas concorrentes de alto desempenho, mas sim, para garantir o armazenamento eficiente de um grande volume de dados, bem como uma boa performance para operações de consulta (HAN et al., 2011). Como os documentos são hierárquicos, um documento pode conter diversos tipos de dados, como escalares, listas ou até outros documentos, já que geralmente não há esquema fixo. Por não haver um esquema fixo, propriedades podem ser adicionadas ou removidas de um documento sem afetar os demais documentos (KLETTKE; STöRL; SCHERZINGER, 2015). Os BDs de documento mais populares são o MongoDB e o CouchDB.

2.1.1.3 Banco de dados de grafo

A maioria dos BDs NoSQL são pensados para executarem em *clusters* e, por isso, facilmente incorporam o modelo de dados de agregados, sobre um grande número de registros complexos com conexões implícitas (aninhadas). No caso dos BDs de grafo, o modelo de dados explora os relacionamentos entre os dados de forma explícita, definindo, assim, registros com várias conexões (SADALAGE; FOWLER, 2012). BDs de grafo são projetados para lidar com dados complexos e altamente conectados, que ultrapassem a capacidade de representação dos BDs relacionais, como é o caso de uma rede social (SADALAGE; FOWLER, 2012).

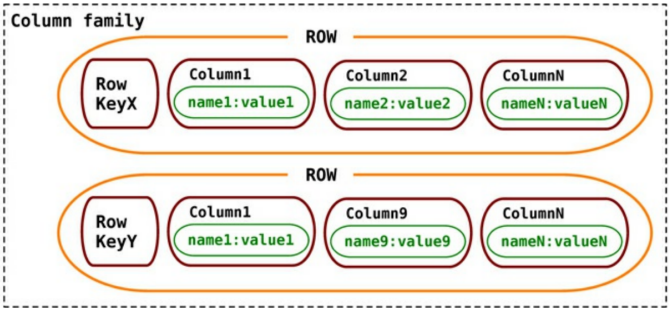
O modelo de dados de um BD de grafo é composto por nós e arestas. Nós representam entidades e arestas representam seus relacionamento, sendo que ambos podem conter atributos (SADALAGE; FOWLER, 2012). Os BDs orientados a grafos que mais se destacam são: Neo4J, Infinite Graph e OrientDB.

2.1.1.4 Banco de dados colunar

BDs colunares são também conhecidos como BDs de família de colunas, BD de armazenamento de registros extensíveis ou BDs de colunas extensas, todos relacionados à estrutura de colunas esparsas normalmente sem esquema. A Figura 2 mostra um exemplo de estrutura de um BD colunar. Ele pode ser visto como um valor ao qual se associa um nome, sendo este par (nome, valor) chamado *coluna*. Um conjunto

de colunas define uma *linha*, que é acessada por um identificador (*Row-Key*) (HEWITT, 2010). Apesar da Figura 2 mostrar linhas com o mesmo número de colunas, linhas diferentes podem ter número de colunas diferentes, sendo adequado à representação de dados heterogêneos.

Figura 2 – Exemplo de estrutura de um BD colunar



Fonte: Sadalage e Fowler (2012)

Um outro modo de pensar o modelo colunar é como uma estrutura agregada de dois níveis. O primeiro nível se comporta como a chave de um BD chave-valor e o segundo nível representa o conjunto de colunas (SADALAGE; FOWLER, 2012). Alguns SGBDs colunares suportam o conceito de *supercoluna*, ou seja, colunas compostas por outras colunas, definindo uma estrutura complexa. Os BDs colunares mais populares são o HBase, Cassandra, Hypertable e simpleDB.

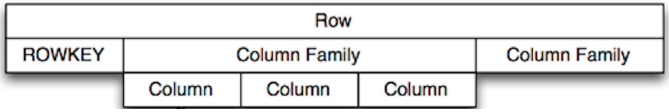
2.1.1.4.1 HBase

HBase é um BD NoSQL colunar, desenvolvido pela Apache, que executa sobre o sistema distribuído do *Hadoop*. Ele foi inspirado pela

publicação da *BigTable*, sendo um SGBD distribuído, de alto desempenho, escalável, com um esquema mais flexível e de código aberto (APACHE, 2015; SHRIPARV, 2010).

O modelo de dados é capaz de acomodar dados semiestruturados diversos, contanto que possam ser convertidos em um *array* de *bytes* para facilitar a distribuição dos dados (APACHE, 2015). Por converter os dados em *arrays* de *bytes*, é difícil definir os tipos de dados sem conhecimento prévio. Ainda que não apresente linguagem de consulta, pode ser acessado através de sua API Java ou com as ferramentas da Apache *Avro* e *Thrift*.

Figura 3 – Estrutura de uma Tabela no HBase



Fonte: Haines (2014)

O modelo de dados mantém uma estrutura bem definida com a qual é possível ter acesso aos dados. O BD, chamado de *namespace*, contém um conjunto de tabelas. Cada tabela tem um conjunto de linhas que segue a estrutura vista na Figura 3, na qual uma linha (*Row*) tem um identificador (*RowKey*) e uma coluna (*Column*) que sempre está vinculada a uma família de colunas (*Column Family*). Assim sendo, um dado no HBase é sempre representado pela seguinte estrutura hierárquica: *namespace*, tabela, linha, família de colunas e coluna.

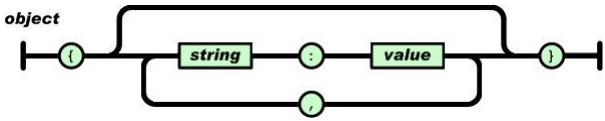
2.2 JSON

JSON (*JavaScript Object Notation*) é um formato de dados que segue um padrão de texto legível por humanos e máquinas, para representar dados semiestruturados. Essa notação ganhou destaque como formato de troca de dados por ser simples e legível (RUIZ; MORALES; MOLINA, 2015), além de ser um formato de serialização de dados independente de linguagem (IETF, 2017).

Na organização de um documento JSON podem ser encontradas basicamente duas estruturas de dados:

- Objetos: um conjunto não ordenado de pares do tipo chave-valor, limitado por chaves, como mostra a Figura 4.

Figura 4 – Estrutura de um objeto JSON



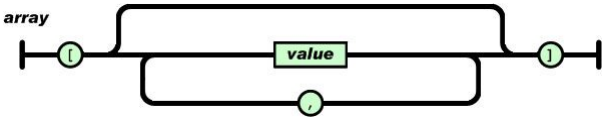
Fonte: JSON (2018)

- *Arrays*: um conjunto ordenado de valores separados por virgula e limitados por colchetes, como mostra a Figura 5.

Quanto ao valor, JSON aceita, além dos tipos primitivos (*string*, número, booleano), outros objetos e *arrays*, podendo assim ser aninhado, assim como o valor *null*, que indica a ausência de valor.

A Figura 6 mostra um exemplo hipotético de documento JSON. Ele exemplifica as estruturas de objeto e *array* descritas anteriormente,

Figura 5 – Estrutura de um *array* JSON



Fonte: JSON (2018)

assim como propriedades.

Figura 6 – Exemplo de Documento JSON

```
{
  'objeto': {
    'propriedadeAninhada': 'valor'
  },
  'array': [
    'prop1',
    'prop2',
    'prop3'
  ],
  'propSimples': 'valor'
}
```

Fonte: Autor

2.3 JSON SCHEMA

Com a popularidade do formato JSON, especialmente em *Web* APIs, é notável que diversos cenários podem se beneficiar de uma maneira de especificar esquemas para documentos JSON. Com um esquema, problemas de tipagem ou falta de requisitos não precisam ser tratados durante o desenvolvimento, pois a formatação correta já é previamente filtrada pelo esquema definido (PEZOA et al., 2016).

JSON *schema* é uma linguagem simples de esquemas, que per-

mite ao usuário descrever a estrutura de um documento JSON e provê um *framework* para verificar a integridade das requisições e respostas JSON. A definição ainda não é um padrão de fato, mas apresenta um crescimento de sua adoção junto às aplicações JSON (PEZOA et al., 2016).

Um JSON *Schema* é descrito como um documento JSON. Este documento apresenta basicamente duas partes: *definitions* e *properties*. A primeira parte permite definir estruturas que vão ser utilizadas no documento. A segunda parte descreve todos os itens do esquema que devem ser validados.

Um exemplo de documento JSON *Schema* pode ser visto na Figura 7. Ele descreve um objeto que pode ter as propriedades *description*, *maxLength*, *maxItems* e *maxProperties*, sendo a propriedade *description* obrigatória. Ele apresenta também uma definição que é reutilizada para descrever várias propriedades.

considerando então o esquema da figura 7, a figura 8 apresenta um JSON que satisfaz as limitações do esquema, sendo um objeto que obrigatoriamente tem a propriedade *description* do tipo *string* além de não apresentar propriedades inesperadas. A figura 9 entretanto não esta em conformidade com o esquema pois não tem a propriedade obrigatória *description* e contem uma propriedade inesperada.

Figura 7 – Exemplo de Documento JSON *Schema*

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://json-schema.org/draft-07/schema#",
  "title": "Core schema subset",
  "definitions": {
    "nonNegativeInteger": {
      "type": "integer",
      "minimum": 0
    }
  },
  "type": "object",
  "properties": {
    "description": {
      "type": "string"
    },
    "maxLength": {
      "$ref": "#/definitions/nonNegativeInteger"
    },
    "maxItems": {
      "$ref": "#/definitions/nonNegativeInteger"
    },
    "maxProperties": {
      "$ref": "#/definitions/nonNegativeInteger"
    },

  },
  "required": [
    "description"
  ]
}
```

Fonte: Autor

Figura 8 – Exemplo de JSON adequado ao esquema

```
{
  'description': "some text!",
  'maxLenght': 255
}
```

Fonte: Autor

Figura 9 – Exemplo de JSON inadequado ao esquema

```
{
  'name': "Eduardo",
  'maxLenght': 255
}
```

Fonte: Autor

3 TRABALHOS RELACIONADOS

Dois trabalhos que apresentam processos de inferência de esquema para BDs NoSQL colunares foram encontrados na literatura.

Foram realizadas buscas no Google Scholar[1] assim como na DBLP[2] considerando os arranjos sobre os três conjuntos a seguir

- *Schema*, NoSQL, NoSQL *Schema*
- *Extraction*, *Inference*, *discovery*, *generation*
- HBase, Cassandra, *columnar*, *column*, *column-family*, *column family*, *key-value*, *key value*, *datastore*

Destas consultas foram filtrados os trabalhos pertinentes a geração de esquemas em BDs NoSQL. Por fim, foi realizado um segundo filtro sobre trabalhos que indicavam um BD colunar. Os trabalhos são detalhados a seguir.

3.1 TRABALHO DE RUIZ ET AL.

Este trabalho levanta os problemas possíveis de manter esquemas implícitos e apresenta uma estratégia de engenharia reversa para inferir um esquema de um BDs NoSQL chave-valor, documento ou colunar (RUIZ; MORALES; MOLINA, 2015). O processo de inferência se dá através da aplicação da metodologia MDE (*Model Driven Engineering*).

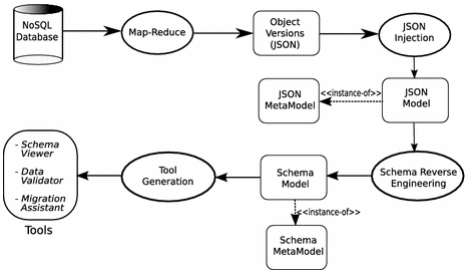
[1] scholar.google.com
[2] https://dblp.uni-trier.de/

Os autores consideram um BD NoSQL como uma coleção arbitrariamente grande de objetos JSON. Ele inclui uma forma de identificador único para cada objeto, chamado de *_id*, e um campo, chamado de *type*, que descreve o tipo da entidade. A proposta não se compromete em seguir algum formato específico de BD NoSQL, mas sim, um formato no qual BDs NoSQL orientados a agregados possam ser inseridos. Por isso a escolha do JSON como formato padrão.

O processo de inferência é composto de três estágios (ver Figura 10):

1. Extrair os objetos relevantes;
2. Definir o esquema JSON em um metamodelo intermediário;
3. Transformar o esquema JSON em um esquema NoSQL.

Figura 10 – Visão geral da arquitetura da proposta



Fonte: Ruiz, Morales e Molina (2015)

O primeiro passo para descobrir os esquemas é obtendo o "esquema bruto" (*raw schema*) de cada objeto que passou pelo *map-reduce*. Para obter o esquema bruto é necessário que a estrutura que descreve

o objeto seja mantida e cada valor primitivo seja substituído pelo seu tipo primitivo. Um exemplo é mostrado na Figura 11.

Figura 11 – Exemplo de Esquema bruto

JSON object	Raw Schema
{name:"Omega", city:"Barcelona"}	*{name:String, city:String}*
{title:"Writing and...", publisher_id:"928672", author:{name:"Bradley Holt", company:{country:"USA", name:"IBM Cloudant"} } }	*{title:String, publisher_id:String, author:{name:String, company:{country:String, name:String} } }*

Fonte: Ruiz, Morales e Molina (2015)

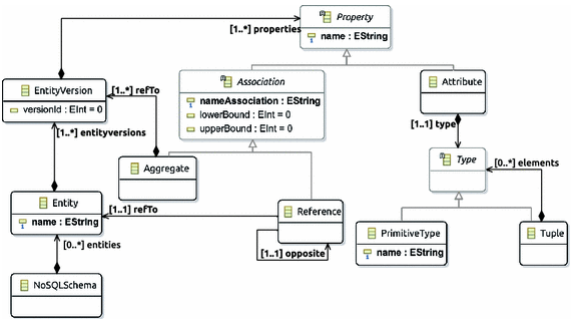
Ainda na primeira etapa, é gerado um par chave-valor para relacionar a versão do objeto. Após isso, para cada versão é selecionado um objeto que represente a arquitetura do grupo e este é adicionado à saída. Na segunda etapa, a coleção de objetos obtida na primeira etapa é convertida em um esquema JSON em conformidade com um metamodelo JSON.

Finalmente, na terceira etapa é feita uma transformação entre modelos, ou seja, do esquema JSON proveniente da segunda etapa para um esquema genérico NoSQL baseado em um modelo de agregados, considerando as entidades, suas versões e atributos. A Figura 12 apresenta o modelo proposto para tratar, além das associações e os atributos, também as referências a agregados e o versionamento.

3.2 TRABALHO DE KIRAN AT AL.

Este trabalho propõe uma abordagem para a integração de dados em BDs HBase expondo um esquema final no formato de ontologias OWL (*Ontology Web Language*) sobre o qual é possível fazer consultas

Figura 12 – Metamodelo de agregados para BDs NoSQL



Fonte: Ruiz, Morales e Molina (2015)

(KIRAN; VIJAYAKUMAR, 2014). O trabalho é dividido em três etapas: uma para inferência de esquemas NoSQL, seguida da conversão do esquema para ontologia e o alinhamento de ontologias.

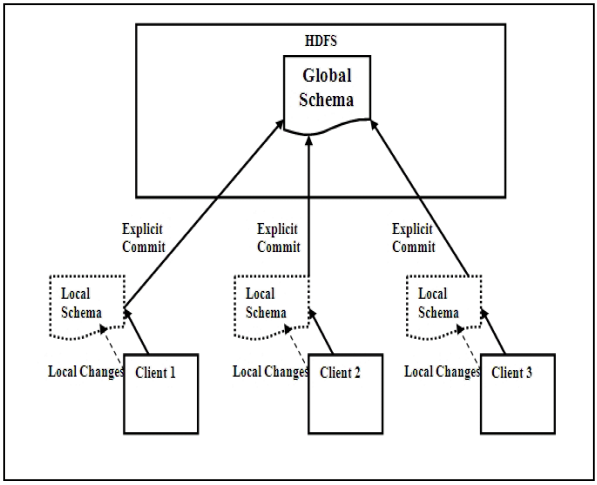
Na etapa de geração de esquema são considerados dois métodos:

- *Online*: o esquema é gerado enquanto dados são inseridos no BD;
- *Offline*: o esquema é gerado com o BD já populado.

Como pode ser visto na Figura 13, clientes locais utilizam a geração de esquema *online*. Posteriormente, alterações são transferidas e adicionadas ao esquema global de forma *offline*.

Para a geração do esquema NoSQL, inicialmente é necessário criar uma tabela de pesquisa, na qual todas as tabelas têm suas linhas mapeadas com base no identificador prévio da linha na tabela de origem. Este processo é dividido em duas etapas e adota uma estratégia *map-reduce*. A primeira etapa gera a tabela de pesquisa e a segunda mapeia as entradas nela.

Figura 13 – Manutenção dos esquemas locais e global



Fonte: Kiran e Vijayakumar (2014)

Após a construção da tabela de pesquisa, aplicam-se algoritmos genéticos para selecionar o registro mais adequado para representar o conjunto de dados. Uma vez definido esse registro, extrai-se o esquema dele. Após o esquema ser obtido, é necessário convertê-lo para OWL. Para realizar a conversão, as seguintes regras são aplicadas:

- O nome da tabela é convertido para uma classe OWL;
- A informação sobre a família de colunas é convertida para estar junto a propriedade da coluna na forma "família_coluna";
- As colunas são considerados instancias da classe da tabela que pertencem.

Após obter a OWL é realizado o alinhamento das ontologias para identificar os relacionamentos semânticos entre as entidades. Essa etapa

é executada com o auxílio de ferramentas de alinhamento já existentes, como o *COMA-Engine*.

3.3 COMPARATIVO

Observando os dois trabalhos na literatura, percebe-se que a inferência de esquemas em BDs colunares ainda não é um grande foco de pesquisa. A Tabela 1 apresenta um comparativo de algumas características observadas nos trabalhos relacionados, como a origem dos dados usada, o objetivo final do trabalho, a abordagem, o formato de saída do esquema gerado, se apresenta interface com o usuário e as etapas do processo.

Tabela 1 – Comparativo entre propostas para inferência de esquemas de BDs NoSQL colunares

	Ruiz, Morales e Molina (2015)	Kiran e Vijayakumar (2014)	Defreyn (2019)
Origem	MongoDB, CouchDB e HBase	HBase	HBase
Objetivo	Geração de esquema	Integração de dados	Geração de esquema
Abordagem	Transformação de modelos	Algoritmos genéticos	Transformação de modelos
Saída	Esquema de Agregados	OWL	JSON *Schema*
Interface	Linha de comando	SPARQL	Gráfica
Etapas	1) Extração da estrutura dos dados JSON (*raw schema*) 2) Transformação para um esquema JSON 3) Transformação para esquema	1) Geração do esquema 2) Conversão de esquema para ontologia 3) Alinhamento de ontologia	1) Inferência dos tipos de dados 2) Inferência da estrutura dos dados (*raw schema*) 3) Geração do JSON *Schema*

Fonte: Autor

A ferramenta proposta neste trabalho se diferencia dos demais trabalhos em diversos pontos. Ela se assemelha em objetivo ao trabalho de Ruiz, Morales e Molina (2015), porém, apresenta um processo mais simples de inferência de esquema que evita o uso de modelos intermediários durante o processo visando consumir menos recursos. Ela também apresenta uma interface gráfica que facilita a interação com o

usuário e sua saída é a recomendação JSON *Schema* ao invés de um esquema de dados em um modelo proprietário.

Em relação ao trabalho de Kiran e Vijayakumar (2014), a ferramenta proposta também foca no BD NoSQL HBase, porém, ela visa apenas a descoberta do esquema e não o processo de integração de dados. O formato de saída também é diferente, pois o propósito deste trabalho não é produzir um esquema conceitual ou semântico através da geração de uma ontologia e, ainda, este trabalho apresenta uma interface com o usuário.

Cabe ressaltar ainda que, diferente dos demais trabalhos, a ferramenta proposta apresenta uma interface gráfica simples e intuitiva ao usuário, além de manter o histórico dos esquemas gerados.

4 HBASI - HBASE SCHEMA INFERENCE TOOL

Este capítulo apresenta a ferramenta denominada *HBaSI - HBase Schema Inference tool*. Ela realiza a inferência de esquemas para o BD NoSQL HBase. As seções a seguir apresentam informações referentes ao projeto da HBaSI (seção 4.1), ao processo de inferência (seção 4.3) e à implementação da ferramenta (seção 4.4).

4.1 PROJETO

HBaSI - HBase Schema Inference tool é uma aplicação *desktop* que recebe como entrada um BD HBase e realiza diversas operações sobre ele para gerar um documento JSON *Schema* que representa a estrutura deste BD.

Ainda que o HBase não apresente esquema explícito, ele apresenta uma hierarquia estrutural fixa (ver Seção 2.1.1.4.1) e isso é explorado na ferramenta. Para manter todas as inferências disponíveis ao usuário é necessário guardar informações sobre a estrutura. A forma adotada para guardar a estrutura e não consumir muitos recursos foi criar novos *namespaces*, com uma estrutura similar ao *namespace* de origem, para guardar as informações de cada inferência. O novo *namespace* se diferencia do original por manter apenas uma linha onde são guardados os resultados.

Tendo o *namespace* para guardar os dados de cada coluna (nome

e tipo de dado) é iniciada a análise de cada coluna. Isso é realizado através de uma comparação de todos os seus valores com representações de tipos de dados em binário para tentar inferir o seu tipo, uma vez que os valores são mantidos em *arrays* de *bytes*, sendo essa uma característica do HBase. Em suma, é preciso definir o tipo de dado de cada valor para então compará-lo com os tipos dos demais valores na mesma coluna e verificar se há consenso.

Por fim, as informações coletadas sobre a estrutura e sobre os nomes e tipos de dados das colunas são convertidos para se adequarem a representação de um documento JSON *Schema*, que é a saída da ferramenta.

Seguindo as etapas do desenvolvimento de *software*, foi realizado o levantamento dos requisitos esperados da aplicação. Os requisitos funcionais (RF) são relativos ao comportamento esperado e os requisitos não funcionais (RNF) são relativos à qualidade do *software*. Informações detalhadas sobre esses requisitos estão presentes no Apêndice A.

Os RF identificados foram os seguintes:

- RF01 - Atualizar lista de BDs
- RF02 - Atualizar lista de esquemas
- RF03 - Gerar nova inferência
- RF04 - Deletar esquemas gerados
- RF05 - Visualizar JSON *Schema* Gerado

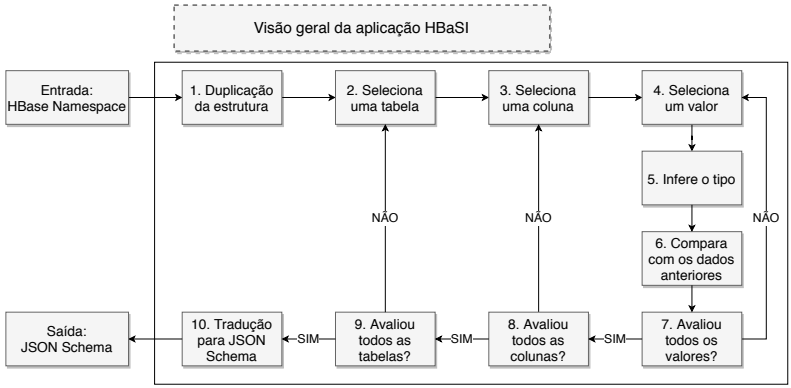
- RF06 - Exportar esquema
- RF07 - Listar esquemas de um BD

Os RNF identificados foram os seguintes:

- RNF01 - Arquitetura: MVC
- RNF02 - Banco de dados: HBase
- RNF03 - Linguagem de programação: Java
- RNF04 - permissão de acesso: Restrito a máquinas executando um nodo do BD

4.2 VISÃO GERAL

Figura 14 – Visão geral da ferramenta



Fonte: Autor

A figura 14 apresenta todo o processo da ferramenta. Inicialmente a ferramenta recebe um *namespace* e duplica sua estrutura, mas

não os dados, para um novo *namespace*. A seguir é feira a navegação sobre a entrada é selecionar uma coluna e sobre todos os valores nesta coluna é a inferência de tipo e a comparação com os demais valores. Esse processo é feito sobre todas as colunas de todas as tabelas e o resultado é armazenado na estrutura duplicada no passo 1.

Como a estrutura do HBase não é similar a de um JSON ela deve ser traduzida para gerar a saída.

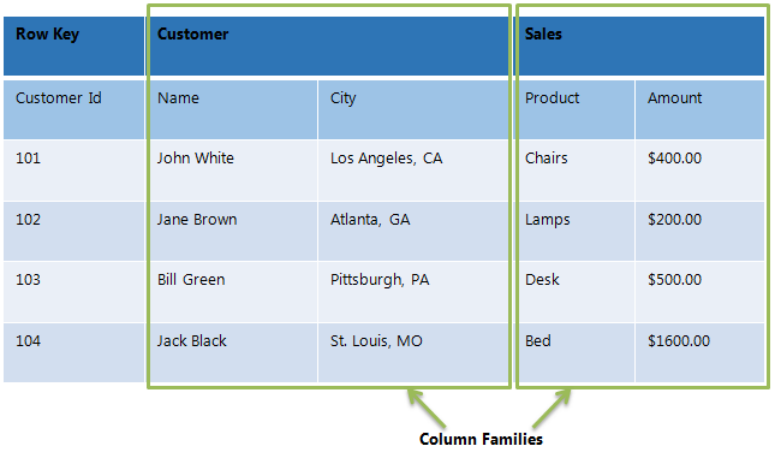
4.3 PROCESSO DE INFERÊNCIA

O processo de inferência adotado pela HBaSI deve se adequar a estrutura do BD HBase, para que possa gerar um esquema para os seus dados. Além disso, ele deve descobrir os tipos de dados que descrevem cada coluna. Para tanto, deve-se analisar todos os valores contidos em todas as colunas para identificar um esquema bruto (*rawSchema*) e então convertê-lo em um JSON *Schema*.

Um desafio para a ferramenta é a descoberta do tipo de dado de uma coluna, pois o HBase apenas opera com *arrays* de *bytes*, uma vez que a aplicação que o acessa é a responsável por saber os tipos de dados (SHRIPARV, 2010). Neste caso, foi necessário desenvolver um algoritmo para identificação de tipos de dados válidos a partir de conteúdos binários dos dados.

Para melhor entender o processo de inferência é importante entender a estrutura de representação de dados usada pelo BD HBase, que é descrita a seguir.

Figura 15 – Exemplo de tabela no HBase



Fonte: Sampagar (2018)

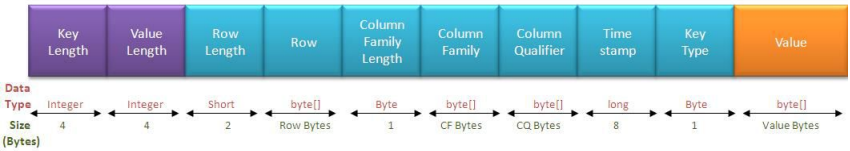
4.3.1 Estrutura de Representação de Dados do HBase

O HBase, por ser um BD colunar e orientado a agregados, utiliza uma estrutura hierárquica para representação dos seus dados. Inicialmente tem-se o conceito de *namespace*, que é um repositório de tabelas com um identificador único.

Uma *tabela* de um *namespace* é formada por um conjunto de linhas. Cada linha possui um identificador único (*Row Key*) e mantém um conjunto de *colunas*. Como pode ser observado no exemplo da Figura 15, uma coluna pertencente a uma tabela deve também pertencer a uma *família de colunas* desta tabela. É também possível que colunas de famílias diferentes tenham o mesmo nome.

A Figura 16 mostra a estrutura adotada pelo HBase para repre-

Figura 16 – Estrutura de um dado no HBase



Fonte: Prafull (2012)

sentar o valor de uma coluna. Os valores em roxo guardam informação sobre o numero de bytes da chave(linha-família-coluna) assim como do valor, os valores em azul guardam dados a estrutura da chave, com o valor da linha, família e coluna além de guardar a versão, por fim em laranja estão todas as informações relativas ao valor, no caso apenas os *bytes* do dado. Nota-se que não existe metadado com informação relativa ao tipo de dado, sendo o campo *key type* referente a chave e não valor, e todos os conteúdos estão armazenados como *bytes*.

Para melhor compreensão da estrutura pode ser considerado um valor de figura 15 por exemplo "John White"na chave "101-Customer-Name". Os campos roxos apenas guardam o numero de *bytes* da chave e do valor enquanto os azuis guardam informações mais detalhadas sobre a a linha (101) a família (Customer) e a coluna(Name), tudo em *bytes*, que são usadas para obter os *bytes* do valor("John White").

Uma vez que o HBase mantém os valores de itens de dados armazenados nesta representação de baixo nível, foi necessário desenvolver um algoritmo de inferência de tipos de dados. Este algoritmo é detalhado a seguir.

4.3.2 Algoritmo de Inferência de Tipo de Dado

A ferramenta HBaSI, para ser capaz de inferir um esquema bruto (*raw schema*) de um BD HBase, necessita analisar os valores de todas as colunas para determinar os seus tipos de dados. Porém, conforme explicado na seção anterior, o valor de uma coluna é apenas representado como um *array* de *bytes* e o HBase não guarda informação sobre o tipo dos dado de suas colunas.

Para lidar com este problema foi analisada a API nativa do HBase. Foi observado que a API apresenta métodos para auxiliar a interação entre as aplicações e o BD, fornecendo conversores de *array* de *bytes* para seguintes tipos: *byte*, *boolean*, *string*, *short*, *char*, *float*, *integer*, *double* e *long*. Ela também fornece métodos para passar esses tipos de dados para *array* de *bytes*. Foi adicionado mais um tipo de dado, denominado *blob*, que representa qualquer dado que não seja possível definir um tipo.

Foi observado também que todos os tipos de dados detêm restrições que podem ser usadas para fins de inferência em um *array* de *bytes*:

- ***byte*** : é representado por um *array* de tamanho um, aceitando qualquer valor;
- ***boolean*** : é representado por um *array* de tamanho um, apenas com os valores 0xFF ou 0x00;
- ***string*** : é representado por um *array* de tamanho variável, con-

tudo seguindo o padrão binário UTF8;

- ***short*** : é representado por um *array* de tamanho dois, aceitando qualquer valor;

- ***char*** : é representado por um *array* de tamanho quatro, tendo seus 2 *bytes* menos significativos representados em *unicode*;

- ***float*** : é representado por um *array* de tamanho quatro, sendo limitado pelos valores indicados no padrão IEEE 754[1], que define a representação dos números binários em ponto flutuante;

- ***integer*** : é representado por um *array* de tamanho quatro e pode utilizar todos os bits (32 bits) para representar um valor inteiro, em complemento de dois, sem limitações;

- ***double*** : é representado por um *array* de tamanho oito, sendo limitado pelos valores indicados no padrão IEEE 754 que define a representação dos números binários em ponto flutuante;

- ***long*** : é representado por um *array* de tamanho oito e pode utilizar todos os bits (64bits) para representar um valor inteiro, em complemento de dois, sem limitações;

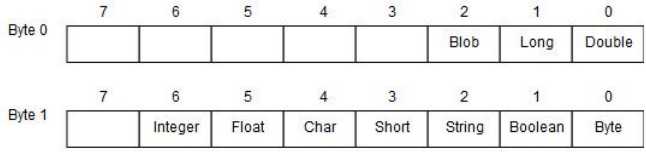
- ***blob*** : qualquer entrada que não se adeque a nenhum dos tipos anteriores.

Com base nessas restrições foi construído o Algoritmo 1. Ele retorna um *array* composto por 2 *bytes* com a indicação dos tipos de

[1] https://ieeexplore.ieee.org/document/4610935

dados possíveis para um determinado valor de coluna passado como entrada para o algoritmo. A Figura 17 mostra a estrutura do *array* de saída da inferência de tipos no formato de uma mascara onde cada tipo de dado é representado por um *bit* no decorrer da execução cada tipo possível tem seu *bit* setado como 1 e os tipos impossíveis tem seus *bits* setados como 0.

Figura 17 – Indicação dos tipos de dados no *array* de saída



Fonte: Autor

O Algoritmo 1 tenta inferir os possíveis tipos de dados para um determinado valor de coluna. Para tanto, ele verifica quais são os tipos de dados possíveis para cada valor, mantendo apenas os tipos que aparecem em algum valor de uma coluna.

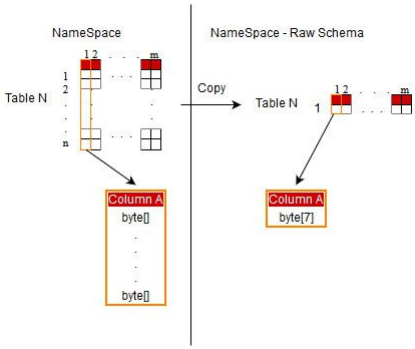
Para salvar os valores do esquema bruto foi feita a cópia da estrutura do *namespace* de origem para um novo *namespace*, conforme explicado na Seção 4.1. A Figura 18 representa esse processo de cópia da estrutura sem copiar os dados, apenas inserindo uma linha onde serão guardados os dados da inferência. Após a realização desta cópia, o Algoritmo 2 faz a análise de todos os valores de uma coluna, primeiramente de forma individual, depois comparando com os demais valores, para obter o *array* de *bytes* final, que é salvo no *namespace* de esquemas brutos da ferramenta.

Algorithm 1 Inferência de tipo de dado

INPUT: valor(*value* - *array de bytes*), tamanho do *array* (*len*)
OUTPUT: *byte*[2] com os possíveis tipos

```
1: function GetTypes(len, value)
2:     switch len do
3:         case 1
4:             if isUTF8Valid(value) then
5:                 output ← string + byte
6:             else if value is 0xFF or 0x00 then
7:                 output ← boolean + byte
8:             else
9:                 output ← byte
10:            end if
11:        case 2
12:            if isUTF8Valid(value) then
13:                output ← string + short
14:            else
15:                output ← short
16:            end if
17:        case 4
18:            if isUTF8Valid(value) then
19:                output ← string
20:            else if isChar(value) then
21:                output ← short
22:            end if
23:            if isFloat(value) then
24:                output ← float
25:            end if
26:            output ← integer
27:        case 8
28:            if isUTF8Valid(value) then
29:                output ← string
30:            end if
31:            if isDouble(value) then
32:                output ← double
33:            end if
34:            output ← long
35:        case outro
36:            if isUTF8Valid(value) then
37:                output ← string
38:            else
39:                output ← blob
40:            end if
       return output
41: end function
```

Figura 18 – Visão geral do processo de inferência do esquema bruto



Fonte: Autor

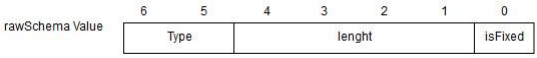
Algorithm 2 Avaliação de tipo de dado de uma coluna

INPUT: todos os valores de de uma determinada coluna
OUTPUT: *byte*[7] com os tipos filtrados e o tamanho da entrada

```
1:  function columnAnalysis(values)
2:      type ← 0x7, 0xFF
3:      fixedLen ← True
4:      for value in values do
5:          len ← value.lenght
6:          if len isNotFixed then
7:              fixedLen ← False
8:          end if
9:          thisValueType ← GetTypes(len, value)
10:         type ← type & thisValueType
11:     end for
12:     output[0] ← fixedLen
13:     output[1] ← len[0]
14:     output[2] ← len[1]
15:     output[3] ← len[2]
16:     output[4] ← len[3]
17:     output[5] ← type[0]
18:     output[6] ← type[1]
        return output
19: end function
```

O *array* de *bytes* gerado pelo Algoritmo 2 segue a organização mostrada na Figura 19. Essa é a organização de todos os valores dos esquemas brutos para guardar as informações obtidas da análise das colunas.

Figura 19 – Organização dos valores de colunas na tabela de esquemas brutos



Fonte: Autor

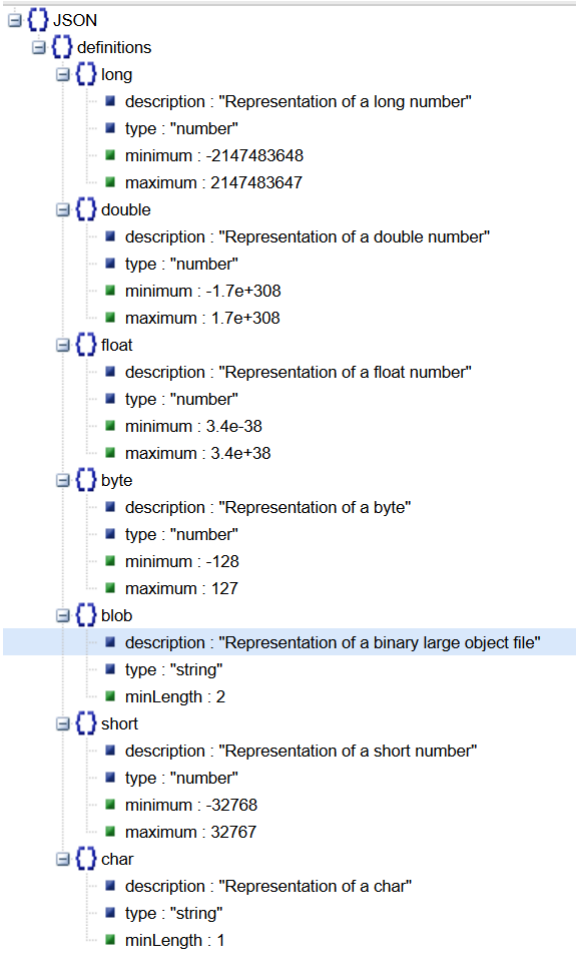
4.3.3 Geração do Esquema no Formato JSON Schema

Esta tarefa da ferramenta HBaSI visa representar a estrutura de um BD HBase, incluindo os tipos de dados inferidos para as colunas, no formato JSON *Schema*. Como os tipos de dados inferidos não são os mesmos tipos de dados nativos do JSON *Schema*, utiliza-se a seção *definitions* de um documento JSON *Schema* para descrever os tipos de dados, de modo que eles possam ser referenciados na definição do esquema. A Figura 20 mostra as definições que são fixas para todos os JSON *Schemas* gerados.

A seção *properties* do documento JSON *Schema* foi utilizada descrever a estrutura hierárquica de um BD HBase a partir de um *namespace*. A Figura 21 exemplifica como foi definida esta estrutura.

A conversão da estrutura HBase para uma estrutura JSON, realizada pela ferramenta, considera cada coluna como uma propriedade

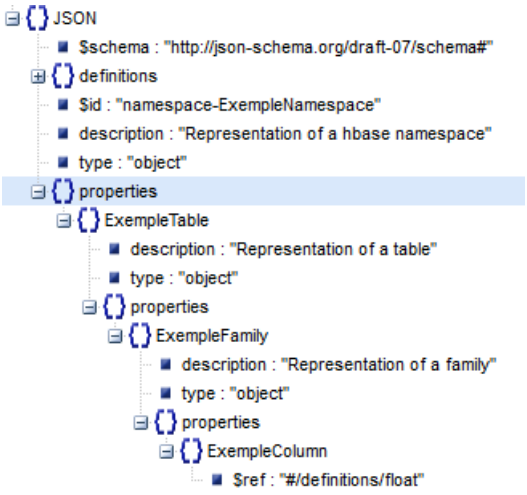
Figura 20 – Definições dos tipos de dados do HBase



Fonte: Autor

contida em um objeto que representa a família de colunas. Por sua vez, cada família é uma propriedade de um objeto que representa uma tabela e, por fim, cada tabela é uma propriedade do documento JSON

Figura 21 – Organização de um JSON *Schema* Gerado



Fonte: Autor

que representa um *namespace* HBase. Essa estrutura também pode ser observada na Figura 21.

A construção do JSON *Schema* é iniciada pelas colunas e vai subindo na hierarquia de representação de dados do HBase. O Algoritmo 3 é a base do esquema e gera o texto referente as colunas usando as informações obtidas do esquema bruto. Em seguida, os Algoritmos 4, 5 e 6 formatam os dados de mais alto nível para se adequarem ao padrão JSON *Schema*.

4.4 IMPLEMENTAÇÃO

Esta seção apresenta as tecnologias utilizadas na implementação da ferramenta, bem como as funcionalidades oferecidas através da sua

Algorithm 3 Tradução dos tipos de dados de cada coluna

INPUT: todas as colunas de uma família
OUTPUT: *string* representando as colunas de uma família

```
1: function columnToJson(columns)
2:     for column in columns do
3:         value ← column.value
4:         for each type in value do
5:             output ← formatted type reference
6:         end for
7:     end for
       return output
8: end function
```

Algorithm 4 Tradução de cada família de colunas

INPUT: Todas as famílias de uma Tabela
OUTPUT: *string* representando as familias de uma tabela

```
1: function familyToJson(families)
2:     for family in families do
3:         columns ← family.columns
4:         output ← formatted family info + columnToJson(Columns)
5:     end for
       return output
6: end function
```

Algorithm 5 Tradução de cada tabela

INPUT: Todas as tabelas de um *namespace*
OUTPUT: *string* representando as tabelas de um *namespace*

```
1: function tableToJson(tables)
2:     for table in tables do
3:         families ← table.families
4:         output ← formatted table info + familyToJson(families)
5:     end for
       return output
6: end function
```

Algorithm 6 Geração do *JSON Schema*

INPUT:O namespace do esquema bruto
OUTPUT: Json *Schema*

```
1: function RAWTOJSON(namespace)
2:     tables ← namespace.tables
3:     output ← definitions + formatted namespace info + tableToJson(tables)
       return output
4: end function
```

interface gráfica com o usuário.

4.4.1 Tecnologias e Ferramentas

A HBaSI foi totalmente codificada na linguagem *Java*. Assim sendo, todas as tecnologias utilizadas para a sua implementação são voltadas à essa linguagem. Elas são as seguintes:

- *Apache Maven*[2]: ferramenta de automação de compilação para projetos *Java* desenvolvida pela *fundação Apache*. Ela utiliza um arquivo XML para descrever a construção do projeto, suas dependências de módulos ou componentes externos, a ordem de compilação, os diretórios e qualquer *plugin* necessário. Além de facilitar o processo de compilação, o *Maven* também mantém um repositório com todos os *plugins* e bibliotecas *Java* e facilita a incorporação das mesmas ao projeto, assim como a incorporação de dependências locais ao repositório.

- *HBase API*[3]: biblioteca de operações sobre o HBase. Todas as

[2]https://maven.apache.org/
[3]https://www.tutorialspoint.com/hbase/hbase_client_api.htm

operações do HBase estão contidas na API *Hadoop*, sobre o qual o sistema de arquivos do BD é construído. Ainda, todas as operações exploram o paralelismo e distributividade do sistema de arquivos e se prendem fortemente à estrutura dele. Por isso, não existe sistema de autenticação explícito. O usuário de acesso é o informado pelo perfil logado.

- *Gson*[4]: biblioteca de codigo aberto, desenvolvida pela *Google*, para tratar de forma simples a conversão de objetos JSON em *string* JSON e vice-versa. Também tem por objetivo garantir o suporte a tipos genéricos do *Java* e converter objetos mesmo sem ter acesso ao seu código fonte.

- *Java Swing*: API que provê uma interface gráfica para projetos *Java*, sendo oficialmente suportada pelo *Java*. Ela fornece um conjunto de ferramentas mais sofisticado que a AWT *(Abstract Window Toolkit)*, que é o conjunto de ferramentas original do Java. Além de prover uma interface similar para diversas plataformas, ela também expande o uso da AWT. Em contrapartida, por ser uma API de mais alto nível e não interagir diretamente com o sistema operacional, ela tem uma degradação no desempenho e consome mais recursos.

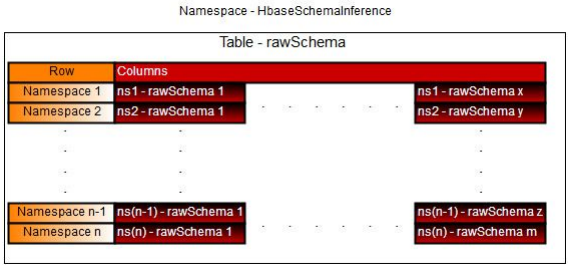
Levando em consideração o BD alvo, HBase, que é desenvolvido em *Java*, assim como o sistema de arquivos que o suporta, a HBaSI foi desenvolvida em *Java desktop*, usando o padrão de projetos MVC

[4]https://github.com/google/gson

(*Model, View, Controller*). Por ser tratar de uma aplicação local, a comunicação com o BD só acontece de forma local através da API *Java*.

A HBaSI mantém ainda um BD a respeito dos esquemas gerados para disponibilizar um histórico de inferências ao usuário. Este BD possui uma tabela para os esquemas brutos onde cada linha representa todos as tentativas de inferência de dados e tipos de dados para um determinado *namespace*. A estrutura desta tabela é mostrada na Figura 22.

Figura 22 – Tabela de esquemas brutos gerados



Fonte: Autor

4.4.2 Funcionalidades

Esta seção descreve as funcionalidades oferecidas pela ferramenta HBaSI para os seus usuários através da sua interface gráfica decorrentes dos requisitos funcionais da análise de requisitos, sendo eles:

- RF01 - Atualizar lista de BDs: Deve ser possível ao usuário atualizar a lista de namespaces, considerando as possíveis inconsis-

tências de um BD distribuído.

- RF02 - Atualizar lista de esquemas: Deve ser possível ao usuário atualizar a lista de esquemas gerados, considerando as possíveis inconsistências de um BD distribuído.

- RF03 - Gerar nova inferência: A qualquer momento o usuário pode gerar novas inferências dos BDs listados contanto que selecione um BD para realizar o processo de inferência.

- RF04 - Deletar esquemas gerados: Deve ser possível ao usuário deletar os esquemas gerados de modo que o mesmo seja permanentemente removido da aplicação.

- RF05 - Visualizar JSON *Schema* Gerado: A qualquer momento, e no final de uma inferência, deve ser possível ao usuário visualizar os JSON schemas gerados e presentes no BD.

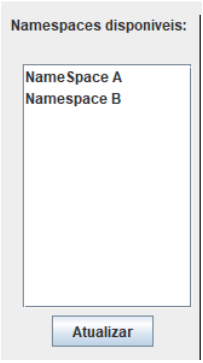
- RF06 - Exportar esquema: Deve ser possível exportar o JSON *Schema* gerado para um arquivo com extensão ".json"ou através do sistema operacional pelo clipboard.

- RF07 - Listar esquemas de um BD: quando selecionado um BD deve ser possível listar todos as inferências existentes do mesmo.

Primeiramente, considerando que o HBase explora a distributividade do seu sistema de arquivos, é necessário permitir que o usuário atualize a lista de *namespaces* sobre os quais pode ser feita inferência de esquemas. Por questões de simplicidade, foi adicionado um botão

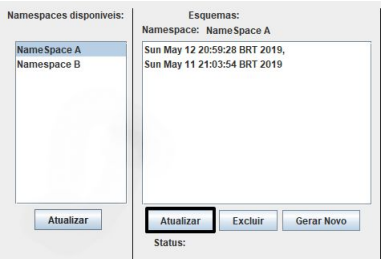
para efetuar a ação. Como pode ser visto na Figura 23, os *namespaces* são listados e é possível atualizá-los com o botão *Atualizar*.

Figura 23 – Atualizar *namespaces*



Fonte: Autor

Figura 24 – Opção de atualização da lista de esquemas

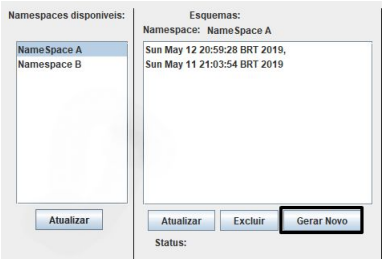


Fonte: Autor

Também é necessário permitir que o usuário atualize a lista de esquemas gerados para o *namespace* selecionado. Como pode ser visto na Figura 24, existe a possibilidade de atualizar a lista de esquemas de forma independente da lista de *namespaces* através do botão *Atualizar* abaixo da lista de esquemas.

Para realizar uma nova inferência de esquema, é necessário selecionar o *namespace* no qual se deseja realizar a ação e clicar no botão *Gerar Novo*, como mostra a Figura 25. Caso não haja *namespace* selecionado, essa opção não é habilitada. Uma vez solicitada uma nova inferência, o campo *Status* abaixo dos botões atualiza as informações sobre a execução, como pode ser visto na Figura 26 e, ao final, informa o tempo total gasto com o processamento da inferência.

Figura 25 – Geração de nova inferência



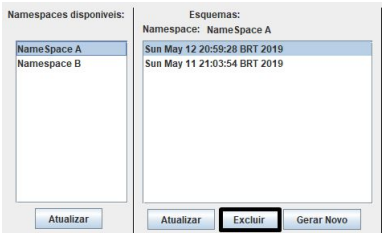
Fonte: Autor

Figura 26 – Atualização sobre a execução

Fonte: Autor

Para excluir um esquema gerado é apenas necessário selecionar previamente o *namespace* ao qual ele pertence, assim como o esquema gerado. Após isso, é necessário clicar no botão *Excluir*, como mostra a Figura 27. Caso não tenha sido selecionado nenhum *namespace*, esta função permanece inativa e, se não for selecionado um esquema, o campo *Status* informa que deve ser escolhido algum esquema.

Figura 27 – Exclusão de esquemas

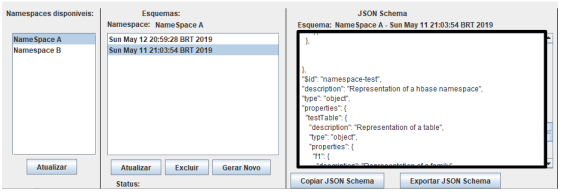


Fonte: Autor

A ferramenta também permite visualizar o JSON *Schema* gerado. Isso ocorre automaticamente sempre que um *namespace* e um esquema são selecionados. Na Figura 28 pode-se observar a visuali-

zação do *JSON Schema* para o esquema *Sun May 11 21:03:54 BRT 2019*.

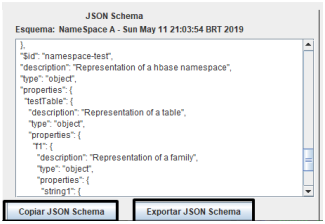
Figura 28 – Visualização de Esquemas



Fonte: Autor

A ferramenta HBaSI apresenta duas alternativas para exportação do JSON *Schema* gerado. Ambas necessitam, obrigatoriamente, que tenha sido previamente selecionado o *namespace* e o esquema para que a visualização dos esquema e sua exportação sejam possíveis. A Figura 29 destaca as duas possibilidades de exportação.

Figura 29 – Exportar JSON *Schema*



Fonte: Autor

É possível exportar para um arquivo JSON clicando no botão *Exportar* JSON *Schema*. Esta opção solicita o diretório de destino onde é gerado um arquivo com a extensão *.json* contendo o nome do *namespace* e do esquema. Também é possível copiar o texto do JSON *Schema*

diretamente para a área de transferência do sistema operacional com o botão *Copiar JSON Schema*.

5 AVALIAÇÃO DA FERRAMENTA

Este capítulo apresenta uma avaliação da qualidade dos dados inferidos pela ferramenta HBaSI. Analisa-se o desempenho da ferramenta em termos de tempo de processamento para inferir esquemas com volumes variados.

5.1 MATERIAIS E MÉTODOS

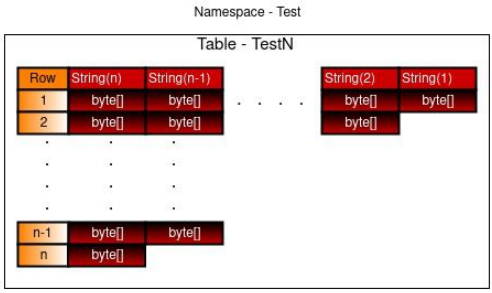
Conforme explicado na Seção 4.1, o HBase possui uma hierarquia estrutural fixa para cada um dos seus *namespaces*. Essa informação sobre a hierarquia *namespace-tabela-família de colunas* pode ser obtida através do acesso aos metadados do HBase. Portanto, a ferramenta HBaSI sempre obtém, com 100% de acurácia, essa hierarquia do esquema de cada BD. A principal dificuldade, portanto, é inferir os tipos de dados das colunas, que é uma informação disponível no formato de *array* de *bytes* nos metadados do HBase, conforme explicado na Seção 4.3.1. Assim sendo, foi considerado que a acurácia da inferência esta diretamente ligada a capacidade de obter metadados sobre os dados e pra avaliar foram construídos BDs.

Considerando os tipos de dados observados na ferramenta foram construídos geradores de dados para cada tipo usando valores pseudo aleatórios para tentar melhorar a efetividade da avaliação. Foram então construídas tabelas com números diferentes de linhas(dez, cem e mil)

que foram preenchidas com dados. Visando a amplitude da avaliação os dados foram distribuídos de modo uniforme na qual todos os tipos a mesma distribuição na tabela.

A Figura 30 mostra um exemplo da estrutura gerada para um tipo específico onde existe o mesmo numero de colunas, de cada tipo, para cada linha o numero de valores na coluna corresponde ao numero da coluna. Deste modo pode-se avaliar a inferência com desde uma coluna com penas um valor até uma coluna com todos os valores.

Figura 30 – Tabela de teste



Fonte: Autor

Para a avaliação da qualidade da inferência foram consideradas três possibilidades de resultado:

- o tipo de dado não condiz com o esperado (**incorreto**);
- o tipo de dado condiz com o esperado (**correto**);
- foram encontrados mais de um tipo de dado possível e o tipo esperado está entre eles (**parcial**).

A possibilidade **parcial** existe por considerar a diferença entre

uma avaliação assertiva de uma onde são apresentados mais de um tipo possível, normalmente nos tipos numéricos.

Sobre esse mesmo BD será realizada uma análise de tempo, apesar do HBase ser um BD para uso preferencialmente distribuído, este experimento foi limitado a uma instância monolítica do BD. Mesmo não sendo um ambiente de teste ideal, os resultados obtidos demonstram o desempenho da ferramenta. Os experimentos foram executados um uma máquina Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz com 6GB de RAM.

Para avaliar o desempenho foi considerado o número de colunas e seus conteúdos (denominados pares chave-valor - *pares C-V*) sendo os mesmos utilizados na inferência de tipos e por isso e possivel calcular o numero de pares C-V de cada tabela através da seguinte fórmula *pares* $C - V = qtd\_tipos * ((linhas/2) * (linhas + 1))$.

Para avaliar o desempenho foi considerada o numero pares avaliados por segundo além de comparar com o tempo da ferramenta *scan*, sendo ela linear em relação ao numero de bytes.

5.2 INFERÊNCIA DE TIPOS DE DADOS

A Tabela 2 mostra os resultados obtidos com o teste de inferência de tipos de dados utilizando a ferramenta HBaSI. É possível observar que os tipos de dados *string*, *boolean*, *byte*, *short* e *blob* são facilmente identificados pois, mesmo com poucos valores, o conjunto de regras que os define é restrito e não contém muitos intervalos em comum com os outros tipos de dados.

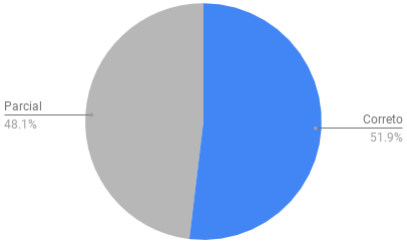
Tabela 2 – Resultado do Teste de Inferência de Tipo de Dado

Tipo de Dado	**Incorreto**	**Correto**	**Parcial**
string	0%	100%	0%
boolean	0%	100%	0%
byte	0%	99%	1%
short	0%	99%	1%
blob	0%	100%	0%
char	0%	0%	100%
integer	0%	17%	83%
float	0%	0%	100%
long	0%	11%	89%
double	0%	0%	100%

Com relação aos tipos de dados *char*, *integer*, *float*, *long* e *double*, ainda que tenha sido possível filtrar a entrada e obter o tipo esperado, não foi possível definir com exatidão o tipo de dado. Esse resultado ocorre por causa das limitações dos tipos de quatro *bytes* (*char*, *integer* e *float*) que não são exclusivas, assim como as dos tipos de oito *bytes* (*long* e *double*), que também não são. Isso acontece por que, ao se analisar os dados em binário, a representação de um tipo de dado está estritamente contido na representação de outro tipo de dado. Por exemplo um *char* sempre vai ter uma codificação binaria válida para representar um *integer* e *float*. Ainda, como existem representações estritamente contidas, alguns valores de *integer* não tem uma codificação binaria válida que represente também um *char* ou um *float*. Por isso, em alguns casos foi possível acertar com exatidão um valor como sendo do tipo *integer*. O mesmo ocoreu para alguns casos de valores do tipo *long*.

A Figura 31 mostra a distribuição de ocorrências dos tipos de

Figura 31 – Percentuais dos resultados obtidos na inferência de tipo



Fonte: Autor

resultado do teste. Percebe-se que não houve nenhuma inferência incorreta. Apenas o percentual de inferências corretas é levemente maior que o percentual de inferências parciais. Nos casos de inferência parcial, a ferramenta retorna o conjunto de todos os possíveis tipos de dados inferidos, o que já serve como indicador de restrição para uma determinada coluna, o que pode ser útil para processos de integração de dados, por exemplo.

5.3 TEMPO DE PROCESSAMENTO

Esta seção apresenta uma análise de tempo de processamento da ferramenta. . A Tabela 3 mostra os tempos de execução para cada tamanho de entrada. A penúltima coluna desta tabela mostra quantos tipos de dados de pares C-V foram processados e inferidos por segundo.

É possível observar que, com um baixo numero de linhas e pares C-V, o tempo necessário para iniciar as operações de geração do BD e posteriormente inferir e salvar os dados da inferência (coluna Tempo

Tabela 3 – Resultado do Experimento de Tempo de Processamento

Linhas	**Pares C-V**	**Tempo (segundos)**	**Pares C-V por Segundo**	**Tempo scan (segundos)**
10	550	4	137	4
100	50500	203	248	274
1000	5005000	23037	229	37581

na Tabela 3) impacta no desempenho final. Já quando o numero de pares C-V aumenta, mantém-se uma relação tempo versus número de pares C-V mais linear, ainda que com alguma perda, provavelmente por que existe um conjunto maior de pares C-V com dados que exigem um maior tempo de processamento, por exemplo, tipos de dados *blob*.

Vale destacar também que os tempos de execução observados na ferramenta HBaSI se mostraram iguais ou inferiores aos tempos da ferramenta de *shell* nativa do HBase quando executado o comando '*scan*', que lista todos o conteúdo de uma tabela.

Por se tratar de uma avaliação preliminar sobre uma instância monolítica, não é possível garantir sempre o bom desempenho da ferramenta HBaSI, pois o tempo de processamento, ainda que apresente um crescimento próximo ao linear, pode se degradar em um ambiente distribuído. Contudo, pode-se ter uma estimativa positiva considerando que o tempo para analisar todos os conteúdos de um *namespace* aparenta ser menor ou igual ao tempo para imprimir o mesmo conjunto de valores. Assim sendo, a ferramenta consegue tratar um volume de dados de forma viável na mesma escala que o HBase.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo principal desenvolver uma ferramenta *desktop* para a inferência de esquemas para BDs NoSQL colunares, com foco no SGBD HBase. Esta ferramenta, denominada HBaSI - HBase Schema Inference tool, se diferencia de trabalhos relacionados por implementar um processo mais simples de inferência que gera um JSON *Schema* representando o esquema do BD de origem, além de disponibilizar uma interface gráfica com o usuário.

Observa-se que todos os objetivos específicos foram atingidos com o desenvolvimento da ferramenta. A ferramenta, apesar de não apresentar acurácia de 100% para todos os tipos de dados inferidos, gera um esquema de resultado que pode ser usado como um indicador para que processos de integração, interoperabilidade ou busca de dados possam melhor entender os dados presentes em um BD HBase. Cabe observar que a HBaSI será útil em uma Tese de Doutorado em desenvolvimento no Programa de Pós-Graduação em Ciência da Computação da UFSC (PPGCC/UFSC) (FROZZA; MELLO, 2018). Esta Tese está desenvolvendo um processo de engenharia reversa de BDs NoSQL e uma de suas etapas é a extração de esquemas em um formato canônico em *JSON Schema*.

Quanto ao conhecimento adquirido, a implementação da ferramenta foi fundamental para o autor aprofundar seus conhecimentos a respeito de operações de baixo nível sobre *bits*. Além disso, houve

aprendizado sobre novas tecnologias de BD, que é o caso dos BDs NoSQL.

A primeira versão da HBaSI encontra-se funcional, atendendo aos objetivos propostos para esse trabalho. Mesmo assim, podem ser listados as seguintes trabalhos futuros:

- Aperfeiçoamento do algoritmo de inferência de tipos de dados;
- Análise de desempenho com grandes volumes de dados reais;
- Melhorias na usabilidade e internacionalização da ferramenta.

Todo o código gerado pode ser encontrado no *GitHub*, através do *link http://lisa.inf.ufsc.br/wiki/index.php/HBaSI*.

REFERENCIAS

APACHE. *Apache HBase ^{TM} Reference Guide*. 2015. <http://hbase.apache.org/book.html>. Acessado em 13 nov. 2018.

BREWER, E. A. Towards robust distributed systems (abstract). In: *nineteenth annual ACM symposium on Principles of distributed computing*. [s.n.], 2000. v. 19, p. 7. ISSN 01635700. <http://portal.acm.org/citation.cfm?doid=343477.343502>.

CATTELL, R. *Scalable SQL and NoSQL data stores*. v. 39, n. 4, 2011. 12-27 p.

CHANG, F. et al. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 2008. <https://doi.org/10.1145/1365815.1365816>.

FROZZA, A. A.; MELLO, R. dos S. A process for reverse engineering of aggregate-oriented nosql databases with emphasis on geographic data. In: *XXXIII Simpósio Brasileiro de Banco de Dados: Demos e WTDBD, SBBD 2018 Companion, Rio de Janeiro, RJ, Brazil, August 25-26, 2018*. [s.n.], 2018. p. 109–115. <http://sbbd.org.br/2018/wp-content/uploads/sites/5/2018/08/109-sbbd_2018_comp.pdf>.

HAINES, S. *Introduction to HBase, the NoSQL Database for Hadoop*. Outubro 2014. <http://www.informit.com/articles/article.aspx?p=2253412>. Acessado em 1 Jun. 2019.

HAN, J. et al. Survey on nosql database. In: *6th International Conference on Pervasive Computing and Applications*. [s.n.], 2011. p. 363–366. <https://doi.org/10.1109/ICPCA.2011.6106531>.

HEWITT, E. *Cassandra: The Definitive Guide*. [S.l.]: O'Reilly Media, 2010. 336 p. ISBN 9781449390419.

IETF. *The JavaScript Object Notation (JSON) Data Interchange Format*. Dezembro 2017. <https://tools.ietf.org/html/rfc8259>. Acessado em 8 nov. 2018.

JSON. *Introdução ao JSON*. 2018. <https://www.json.org/json-pt.html>. Acessado em 13 nov. 2018.

JSON-SCHEMA. *Specification Links*. 2018. <http://json-schema.org/specification-links.html>. Acessado em 8 nov. 2018.

KIRAN, V. K.; VIJAYAKUMAR, R. Ontology based data integration of nosql datastores. In: *9th Int. Conf. on Industrial and Information Systems*. [s.n.], 2014. p. 1–6. <https://ieeexplore.ieee.org/abstract/document/7036545>.

KLETTKE, M.; STöRL, U.; SCHERZINGER, S. Schema extraction and structural outlier detection for json-based nosql data stores. In: *BTW*. [S.l.: s.n.], 2015. v. 241, p. 425–444. ISBN 9783885796350. ISSN 16175468.

LYNCH, N.; GILBERT, S. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. In: *SIGACT News*. [s.n.], 2002. v. 33, n. 2, p. 51–59. <https://users.ece.cmu.edu/ adrian/731-sp04/readings/GL-cap.pdf>.

NEO4J. *Graph Databases for Beginners: A Tour of Aggregate Stores*. 2015. <https://neo4j.com/blog/aggregate-stores-tour/>. Acessado em 11 nov. 2018.

PEZOA, F. et al. Foundations of json schema. In: *25th International Conference on World Wide Web*. [s.n.], 2016. p. 263–273. ISBN 9781450341431. <https://dl.acm.org/citation.cfm?doid=2872427.2883029>.

PRAFULL. *How to calculate the record size of HBase?* 2012. <http://prafull-blog.blogspot.com/2012/06/how-to-calculate-record-size-of-hbase.html>. Acessado em 18 abr. 2019.

RUIZ, D. S.; MORALES, S. F.; MOLINA, J. G. *Inferring Versioned Schemas from NoSQL Databases and its Applications*. v. 9381, n. October, 2015. 467–480 p. <https://link.springer.com/chapter/10.1007%2F978-3-319-25264-3_35>.

SADALAGE, P. J.; FOWLER, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. [S.l.: s.n.], 2012. 192 p. ISSN 1098-. ISBN 9780321826626.

SAMPAGAR, V. *Read HBase Table using HBase shell get Command and Examples*. 2018. <https://www.netwoven.com/2013/10/10/hbase-overview-of-architecture-and-data-model/>. Acessado em 26 nov. 2018.

SCHERZINGER, S.; KLETTKE, M.; STöRL, U. Managing schema evolution in nosql data stores. *arXiv preprint arXiv:1308.0514*, 2013. <https://arxiv.org/abs/1308.0514>.

SHRIPARV, S. *Learning HBase*. [S.l.]: Packt Publishing, 2010. 326 p. ISBN 9781783985944.

TUDORICA, B. G.; BUCUR, C. A. A comparison between several nosql databases with comments and notes. In: *RoEduNet IEEE International Conference*. [s.n.], 2011. ISSN 20681038. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5993686>.

APÊNDICE A – Análise de Requisitos

A.1 PROJETO - HBASI - HBASE SCHEMA INFERENCE TOOL

A.2 INTRODUÇÃO

A.2.1 Objetivo

desenvolver uma ferramenta que possibilite a inferência de esquemas e BDs colunares HBase e visualização em JSON *Schema*.

A.2.2 Descrição da aplicação

HBaSI - HBase Schema Inference tool é uma aplicação que tem como entrada um BD em Hbase e após realizar a análise dos dados salva-os e gera seu JSON *Schema*, sendo que o *Schema* pode ser copiado diretamente ou salvo em arquivo.

A.3 VISÃO GERAL

A.3.1 Arquitetura da aplicação

Por se tratar de uma aplicação *desktop* ela deve ser desenvolvida usando o padrão MVC.

A.3.2 Premissas de desenvolvimento

- A aplicação deve provar uma interface gráfica onde o usuário pode interagir com a aplicação.

- A aplicação deve ser executada em uma maquina que executa uma instancia do banco de dados.

- Os dados gerados devem ser persistidos.

- A aplicação deve ser desenvolvida com a mesma linguagem que o banco de dados.

A.4 REQUISITOS DE SOFTWARE

A.4.1 Requisitos Funcionais (RF)

- RF01 - Atualizar lista de BDs: Deve ser possível ao usuário atualizar a lista de namespaces, considerando as possíveis inconsistências de um BD distribuído.

- RF02 - Atualizar lista de esquemas: Deve ser possível ao usuário atualizar a lista de esquemas gerados, considerando as possíveis inconsistências de um BD distribuído.

- RF03 - Gerar nova inferência: A qualquer momento o usuário pode gerar novas inferências dos BDs listados contanto que selecione um BD para realizar o processo de inferência.

- RF04 - Deletar esquemas gerados: Deve ser possível ao usuário deletar os esquemas gerados de modo que o mesmo seja permanentemente removido da aplicação.

- RF05 - Visualizar JSON *Schema* Gerado: A qualquer momento, e no final de uma inferência, deve ser possível ao usuário visualizar os JSON schemas gerados e presentes no BD.

- RF06 - Exportar esquema: Deve ser possível exportar o JSON *Schema* gerado para um arquivo com extensão ".json"ou através do sistema operacional pelo clipboard.

- RF07 - Listar esquemas de um BD: quando selecionado um BD deve ser possível listar todos as inferências existentes do mesmo.

A.4.2 Requisitos Não Funcionais (RNF)

- RNF01 - Arquitetura: Deve ser desenvolvido usando *MVC*.

- RNF02 - Banco de dados: O BD utilizado deve ser HBase, sem ferramentas adicionais.

- RNF03 - Linguagem de programação: Deve ser desenvolvido em *java desktop*, a mesma linguagem com a qual foi construído o HBase e com a API nativa.

- RNF04 - permissão de acesso: A execução deve ser feita em um maquina com acesso ao BD e o usuário deve ter permissão para acessar as bases de dados.

APÊNDICE B – codigo fonte

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package HbaseSchemaInference.control;

import HbaseSchemaInference.model.RawSchema;
import HbaseSchemaInference.view.MainView;
import java.io.IOException;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.Map.Entry;
import java.util.NavigableMap;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.ZooKeeperConnectionException;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.util.Bytes;

/**
 *
 * @author eduardo
 */
public class App {

    //hbase namespace, man table, man family, new namespace
ident
    static String[] manageNamespace = {"hBaseSchemaInference",
```

```
"rawSchemas", "data", "_rawSchema_"};
    static String inferenceNamespace2 = "tests2", inferenceNamespace
= "tests";
    static String inferenceTable = "testTable";
    private MainView view;
    private HbaseOperations ops;

    public static void main(String[] args) {
        new App();
    }

    public App() {
        ops = new HbaseOperations();
        ops.createNamespace(manageNamespace[0]);
        ops.createTable(manageNamespace[0], manageNamespace[1],
(new String[]{manageNamespace[2]}));
        view = new MainView(this);
        view.setVisible(true);
        //ops.deleteNamespace(namespace);
        //ops.deleteNamespace("tests_rawSchema_15551"+"11674769");
    }

    public String[] getNamespaces() {
        return ops.getNamespaces(true, manageNamespace);
    }

    public String[] getSchemes(String namespace) {
        return ops.getSchemes(manageNamespace[0], manageNamespace[1],
manageNamespace[2], namespace);
    }

    public String getSchema(String namespace) {
        String[] split = namespace.split("_");
        RawSchema rawSchema = new RawSchema(split[0], ops, namespace,
this);
        return rawSchema.rawToJSON();
    }

    public void newSchema(String namespace) {
        Date date = new Date();
        String newNamespace = namespace + manageNamespace[3]
```

```
+ date.getTime();
        ops.putData(manageNamespace[0], manageNamespace[1],
namespace, Bytes.toBytes(manageNamespace[2]), Bytes.toBytes(newName
Bytes.toBytes(newNamespace));
        RawSchema rawSchema = new RawSchema(namespace, ops,
newNamespace, this);
        new Thread() {

            @Override
            public void run() {
                long rawTime = rawSchema.getRawSchema();
                view.newSchema(date, rawTime);
            }
        }.start();

    }

    public void updateStatus(String text) {
        view.updateStatus(text);
    }

    public void deleteScheme(String selected) {
        ops.deleteNamespace(selected);
        String row = selected.split("_")[0];
        ops.deleteColumn(manageNamespace[0], manageNamespace[1],row
    }
}


/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
 /*
https://github.com/larsgeorge/hbase-book/tree/master/ch05/src/main/
code reference.

 */
package HbaseSchemaInference.control;
```

```
import HbaseSchemaInference.model.PutData;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.NamespaceDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Delete;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.filter.BinaryComparator;
import org.apache.hadoop.hbase.filter.CompareFilter;
import org.apache.hadoop.hbase.filter.RowFilter;
import org.apache.hadoop.hbase.security.User;
import org.apache.hadoop.hbase.util.Bytes;

/**
 *
 * @author eduardo
 */
public class HbaseOperations {

    private Connection connection;

    public User getUser() throws IOException {
        return User.getCurrent();
    }
```

```
    /*
        return a connection or null on erro.
     */
    public Connection connect() {
        if (connection != null) {
            return connection;
        }
        try {
            Configuration conf = HBaseConfiguration.create();
            connection = ConnectionFactory.createConnection();
            return connection;
        } catch (IOException ex) {
            Logger.getLogger(HbaseOperations.class.getName()).log(L
null, ex);
            return null;
        }
    }

    /*
        return
         1 if namespace already exists
         0 on sucess
        -1 on erro

     */
    public short createNamespace(String name) {
        Admin admin = null;
        Connection connection = connect();
        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {

            return -1;

        }

        try {
            admin.getNamespaceDescriptor(name);
```

```
            return 1;
        } catch (IOException ex) {
            try {
                NamespaceDescriptor namespace = NamespaceDescriptor.crea
                admin.createNamespace(namespace);
            } catch (IOException ex1) {
                Logger.getLogger(HbaseOperations.class.getName()).log(Le
null, ex1);
            }
        }

        return 0;

    }

    /*
        return
         1 if table already exists
         0 on sucess
        -1 on erro
        -2 namespace dont exists
        -3 error on creation

     */
    public short createTable(String namespace, String name,
String[] familyName) {
        Admin admin = null;

        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {

            return -1;

        }

        try {
            admin.getNamespaceDescriptor(namespace);

        } catch (IOException ex) {
```

```
            return -2;
        }

        TableName tableName = TableName.valueOf(namespace, name);

        try {
            admin.getTableDescriptor(tableName);

            return 1;
        } catch (IOException ex) {
        }

        HTableDescriptor desc = new HTableDescriptor(tableName);

        for (String family : familyName) {
            HColumnDescriptor coldef = new HColumnDescriptor(Bytes.
            desc.addFamily(coldef);
        }
        try {
            admin.createTable(desc);
        } catch (IOException ex) {

            return -3;
        }

        return 0;

    }

    /*
        return
         1 all families already exists
         0 on sucess
        -1 on erro
        -2 namespace dont exists
        -3 table dont exists
        -4 error on modify

     */
    public short alterFamilies(String namespace, String table,
```

```
String[] familyName) {
        Admin admin = null;
        ;
        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {
            Logger.getLogger(HbaseOperations.class.getName()).log(Level.
null, ex);

            return -1;

        }

        try {
            admin.getNamespaceDescriptor(namespace);

        } catch (IOException ex) {

            return -2;
        }

        TableName tableName = TableName.valueOf(namespace, table);
        HTableDescriptor desc = null;
        try {
            desc = admin.getTableDescriptor(tableName);
        } catch (IOException ex) {

            return -3;
        }
        boolean allFamilyExists = true;
        for (String family : familyName) {
            HColumnDescriptor coldef = new HColumnDescriptor(Bytes.toByt
            if (!desc.hasFamily(Bytes.toBytes(family))) {
                allFamilyExists = false;
                desc.addFamily(coldef);
            }
        }
        if (allFamilyExists) {

            return 1;
```

```
        }
        try {
            admin.modifyTable(tableName, desc);
        } catch (IOException ex) {

            return -4;
        }

        return 0;

    }

    /*
        return
        0 on sucess
        -1 table not founded
        -2 put error

     */
    public short putData(String namespace, String table, String
row, byte[] family, byte[] column, byte[] value) {

        TableName tableName = TableName.valueOf(namespace, table);
        byte[] rowName = Bytes.toBytes(row);
        Put p = new Put(rowName);
        Table tableClass = null;
        try {
            tableClass = connection.getTable(tableName);
        } catch (IOException ex) {

            return -1;
        }
        p.addColumn(family, column, value);
        try {
            tableClass.put(p);
        } catch (IOException ex) {

            return -2;
        }

        return 0;
```

```
    }

    /*
        return
        0 on sucess
        -1 table not founded
        -2 put error

     */
    public short putArrayOfData(String namespace, String table,
String row, ArrayList<PutData> data) {

        TableName tableName = TableName.valueOf(namespace, table);
        byte[] rowName = Bytes.toBytes(row);
        Put p = new Put(rowName);
        Table tableClass = null;
        try {
            tableClass = connection.getTable(tableName);
        } catch (IOException ex) {

            return -1;
        }
        for (PutData entry : data) {
            p.addColumn(entry.getFamily(), entry.getColumn(),
entry.getValue());
        }
        try {
            tableClass.put(p);
        } catch (IOException ex) {

            return -2;
        }

        return 0;

    }

    /*
        return
         0 on sucess
```

```
        -1 on erro
        -2 namespace dont exists
        -3 error on disable and delete

     */
    public short deleteTable(String namespace, String table)
{
        Admin admin = null;

        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {

            return -1;

        }

        try {
            admin.getNamespaceDescriptor(namespace);

        } catch (IOException ex) {

            return -2;
        }

        TableName tableName = TableName.valueOf(namespace, table);

        try {
            admin.disableTable(tableName);
            admin.deleteTable(tableName);
        } catch (IOException ex) {

            return -3;
        }

        return 0;
    }

    /*
        return
```

```
         0 on sucess
        -1 on erro
        -2 namespace dont exists
        -3 error on disable and delete

     */
    public short deleteNamespace(String namespace) {
        Admin admin = null;

        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {

            return -1;

        }
        NamespaceDescriptor namespaceDescriptor = null;
        try {
            namespaceDescriptor = admin.getNamespaceDescriptor(namespace

        } catch (IOException ex) {

            return -2;
        }
        String[] tables = this.getTables(namespace);
        for (String table : tables) {
            deleteTable(namespace, table);
        }

        try {
            admin.deleteNamespace(namespace);
        } catch (IOException ex) {

            return -3;
        }

        return 0;
    }
```

```
/*
    return the table names of a namespace
 */
public String[] getTables(String namespace) {
    Admin admin = null;

    try {
        admin = connection.getAdmin();

    } catch (IOException ex) {

        return null;

    }
    NamespaceDescriptor namespaceDescriptor;
    try {
        namespaceDescriptor = admin.getNamespaceDescriptor(name

    } catch (IOException ex) {

        return null;
    }

    HTableDescriptor[] tables = new HTableDescriptor[0];
    try {
        tables = admin.listTableDescriptorsByNamespace(namespac
    } catch (IOException ex) {

        return null;
    }
    String[] tableNames = new String[tables.length];
    for (int i = 0; i < tables.length; i++) {
        tableNames[i] = tables[i].getNameAsString().split(":")[

    }

    return tableNames;
}

/*
    return the table names of a namespace
```

```
     */
    public String[] getFamilies(String namespace, String table)
{
        Admin admin = null;

        try {
            admin = connection.getAdmin();

        } catch (IOException ex) {

            return null;

        }

        TableName tableName = TableName.valueOf(namespace, table);
        HTableDescriptor desc = null;
        try {
            desc = admin.getTableDescriptor(tableName);
        } catch (IOException ex) {

            return null;
        }

        HColumnDescriptor[] columnFamilies = desc.getColumnFamilies();
        String[] familyNames = new String[columnFamilies.length];
        for (int j = 0; j < columnFamilies.length; j++) {
            familyNames[j] = columnFamilies[j].getNameAsString();
        }

        return familyNames;
    }

    public String[] getColumns(String namespace, String table,
String family) {
        String[] columnNames = new String[0];

        TableName tableName = TableName.valueOf(namespace, table);

        Table table2 = null;
        try {
            table2 = connection.getTable(tableName);
```

```
        } catch (IOException ex) {

            return null;
        }
        Scan scan = new Scan();
        scan.addFamily(Bytes.toBytes(family));

        try {
            ResultScanner scanner = table2.getScanner(scan);
            for (Result result2 = scanner.next(); result2 !=
null; result2 = scanner.next()) {
                List<Cell> family_cells = result2.listCells();
                String[] tempNames = new String[family_cells.size()
                int i = 0;
                for (Cell family_cell : result2.listCells())
{
                    byte[] rowArray = family_cell.getRowArray();
                    byte[] column = Arrays.copyOfRange(rowArray,
family_cell.getQualifierOffset(), family_cell.getQualifierOffset()
+ family_cell.getQualifierLength());
                    tempNames[i] = Bytes.toString(column);
                    i++;
                }
                columnNames = combine(columnNames, tempNames);
            }
        } catch (IOException ex) {

            return null;
        }

        return columnNames;
    }

    public ResultScanner getValuesScan(String namespace, String
table, String family, String column) {

        TableName tableName = TableName.valueOf(namespace, table);

        Table table2 = null;
        try {
            table2 = connection.getTable(tableName);
```

```
        } catch (IOException ex) {

            return null;
        }
        Scan scan = new Scan();
        scan.addColumn(Bytes.toBytes(family), Bytes.toBytes(column));
        ResultScanner scanner = null;
        try {
            scanner = table2.getScanner(scan);

        } catch (IOException ex) {

            return null;
        }

        return scanner;
    }

    public ResultScanner getTableScan(String namespace, String
table) {

        TableName tableName = TableName.valueOf(namespace, table);

        Table table2 = null;
        try {
            table2 = connection.getTable(tableName);
        } catch (IOException ex) {

            return null;
        }
        Scan scan = new Scan();
        ResultScanner scanner = null;
        try {
            scanner = table2.getScanner(scan);

        } catch (IOException ex) {

            return null;
        }

        return scanner;
```

```
    }

    public short deleteColumn(String namespace, String table,
String row, String family, String column) {

        TableName tableName = TableName.valueOf(namespace, table);

        Table table2 = null;
        try {
            table2 = connection.getTable(tableName);
        } catch (IOException ex) {

            return -1;
        }
        Delete delete = new Delete(Bytes.toBytes(row));
        delete.addColumn(Bytes.toBytes(family), Bytes.toBytes(colum
        try {
            table2.delete(delete);
        } catch (IOException ex) {
            return -2;
        }
        return 0;
    }

    /*
        https://javarevisited.blogspot.com/2013/02/combine-integer-
     */
    public String[] combine(String[] a, String[] b) {
        int length = a.length + b.length;
        String[] result = new String[length];
        System.arraycopy(a, 0, result, 0, a.length);
        System.arraycopy(b, 0, result, a.length, b.length);
        return result;
    }

    public String[] getNamespaces(boolean clear, String[] managemen
{
        Admin admin = null;

        try {
            admin = connection.getAdmin();
```

```
        } catch (IOException ex) {

            return new String[0];

        }
        NamespaceDescriptor[] namespaceDescriptor;
        String[] namespaces;
        try {
            namespaceDescriptor = admin.listNamespaceDescriptors();
            namespaces = new String[namespaceDescriptor.length];
            int i = 0;
            for (NamespaceDescriptor namespace : namespaceDescriptor)
{
                String name = namespace.getName();
                if (!name.contains(management[3]) && !management[0].equal
{
                    namespaces[i] = namespace.getName();
                    i++;
                }
            }
            return namespaces;
        } catch (IOException ex) {

            return new String[0];
        }
    }

    String[] getSchemes(String managementNamespace, String managementTabl
String managementFamily, String namespace) {
        String[] namespaceNames = new String[0];

        TableName tableName = TableName.valueOf(managementNamespace,
managementTable);

        Table table2 = null;
        try {
            table2 = connection.getTable(tableName);
        } catch (IOException ex) {

            return null;
```

```
        }
        Scan scan = new Scan();
        scan.addFamily(Bytes.toBytes(managementFamily));
        scan.setFilter(new RowFilter(CompareFilter.CompareOp.EQUAL,
new BinaryComparator(Bytes.toBytes(namespace))));
        try {
            ResultScanner scanner = table2.getScanner(scan);
            for (Result result2 = scanner.next(); result2 !=
null; result2 = scanner.next()) {
                List<Cell> family_cells = result2.listCells();
                String[] tempNames = new String[family_cells.size()
                int i = 0;
                for (Cell family_cell : result2.listCells())
{
                    byte[] rowArray = family_cell.getRowArray();
                    byte[] column = Arrays.copyOfRange(rowArray,
family_cell.getQualifierOffset(), family_cell.getQualifierOffset()
+ family_cell.getQualifierLength());
                    tempNames[i] = Bytes.toString(column);
                    i++;
                }
                namespaceNames = combine(namespaceNames, tempNames)
            }
        } catch (IOException ex) {

            return null;
        }

        return namespaceNames;
    }

}


/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package HbaseSchemaInference.model;
```

```
import HbaseSchemaInference.control.HbaseOperations;
import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;
import org.apache.hadoop.hbase.util.Bytes;

/**
 *
 * @author eduardo
 */
public class TablePopulator {

    String[] types = {"byte", "boolean", "string", "short",
"char", "float", "integer", "double", "long", "blob"};
    String namespace, table;
    ArrayList<String> families, rows;

    public TablePopulator(String namespace, String table, int
numFamilies, int numRows) {
        this.namespace = namespace;
        this.table = table;
        families = new ArrayList<String>();
        rows = new ArrayList<String>();
        for (int i = 0; i < numFamilies; i++) {
            families.add("f" + i);
        }
        for (int i = 0; i < numRows; i++) {
            rows.add("r" + i);
        }
        HbaseOperations ops = new HbaseOperations();
        ops.createNamespace(namespace);
        String[] fam = families.toArray(new String[families.size()]);
        short tab = ops.createTable(namespace, table, fam);
        if(tab == 1){
            ops.alterFamilies(namespace, table, fam);
        }
        for (int i = numRows; i > 0; i--) {
            for (int j = 0; j < types.length; j++) {
                String family = families.get(ThreadLocalRandom.current()
families.size()));
                for (int k = 0; k < i; k++) {
```

```
                    byte[] value = null;
                    switch (types[j]) {
                        case "byte":
                            value = new byte[]{makeByte()};
                            break;
                        case "boolean":
                            value = new byte[]{makeBoolean()};
                            break;
                        case "string":
                            value = makeString(0);
                            break;
                        case "short":
                            value = makeShort();
                            break;
                        case "char":
                            value = makeChar();
                            break;
                        case "float":
                            value = makeFloat();
                            break;
                        case "integer":
                            value = makeInt();
                            break;
                        case "double":
                            value = makeDouble();
                            break;
                        case "long":
                            value = makeLong();
                            break;
                        case "blob":
                            value = makeBlob(0);
                            break;

                    }
                    ops.putData(namespace, table, rows.get(k),
Bytes.toBytes(family), Bytes.toBytes((types[j] + i)), value);
                }
            }
        }
    }
```

```
    public String getNamespace() {
        return namespace;
    }

    public String getTable() {
        return table;
    }

    public ArrayList<String> getFamilies() {
        return families;
    }

    public ArrayList<String> getRows() {
        return rows;
    }

    //data generators
    /* datatypes
        *byte
        *boolean
        *string
        *short
        *char
        *float
        *integer
        *double
        *long
        *blob

     */
    public byte makeByte() {
        byte output = (byte) ThreadLocalRandom.current().nextInt(0x0,
0xFF + 1);
        return output;
    }

    public byte makeBoolean() {
        byte output = (byte) 0xFF;
        int rand = ThreadLocalRandom.current().nextInt(0, 2);
        if (rand == 0) {
            output = (byte) 0x00;
```

```
        }
        return output;
    }

    public byte[] makeString(int lenght) {
        byte[] output = new byte[0];
        int[] cMin = new int[5], cMax = new int[5];
        cMin[0] = 0x80;
        cMax[0] = 0xBF;
        cMin[1] = 0x1;
        cMax[1] = 0x7E;
        cMin[2] = 0xC2;
        cMax[2] = 0xDF;
        cMin[3] = 0xE0;
        cMax[3] = 0xEF;
        cMin[4] = 0xF0;
        cMax[4] = 0xF5;

        if (lenght == 0) {
            lenght = getRandom(1, 3000);
        }
        int size = 0;

        while (size < lenght) {
            int rest = lenght - size;
            int byteLen = 0;

            switch (rest) {
                case 1:
                    byteLen = 1;
                    break;
                case 2:
                    byteLen = ThreadLocalRandom.current().nextInt(1
3);
                    break;
                case 3:
                    byteLen = ThreadLocalRandom.current().nextInt(1
4);
                    break;
                default:
                    byteLen = ThreadLocalRandom.current().nextInt(1
```

```
5);

            }
            byte[] atual = new byte[byteLen];
            atual[0] = (byte) ThreadLocalRandom.current().nextInt(cMin[b
cMax[byteLen] + 1);
            for (int i = 1; i < byteLen; i++) {
                atual[i] = (byte) ThreadLocalRandom.current().nextInt(cM
cMax[0] + 1);
            }
            output = combine(output, atual);
            size++;
        }
        return output;
    }

    public byte[] makeShort() {
        byte[] output = new byte[2];
        short rand = (short) ThreadLocalRandom.current().nextInt(0x0,
0xFFFF + 1);
        output = Bytes.toBytes(rand);
        return output;
    }

    public byte[] makeChar() {
        byte[] output = new byte[4];
        char rand = (char) ThreadLocalRandom.current().nextInt(0x1,
0xFF65 + 1);
        output = Bytes.toBytes(rand);
        return output;
    }

    public byte[] makeFloat() {
        byte[] output = new byte[2];
        float rand = ThreadLocalRandom.current().nextFloat()
* ThreadLocalRandom.current().nextInt(0x1, 100000) * (-ThreadLocalRandom
2));
        output = Bytes.toBytes(rand);
        return output;
    }
```

```
    public byte[] makeInt() {
        byte[] output = new byte[4];
        int rand = ThreadLocalRandom.current().nextInt();
        output = Bytes.toBytes(rand);
        return output;
    }

    public byte[] makeDouble() {
        byte[] output = new byte[8];
        double rand = ThreadLocalRandom.current().nextDouble(-2
^ 50, 2 ^ 50);
        output = Bytes.toBytes(rand);
        return output;
    }

    public byte[] makeLong() {
        byte[] output = new byte[8];
        long rand = ThreadLocalRandom.current().nextLong();
        output = Bytes.toBytes(rand);
        return output;
    }

    public byte[] makeBlob(int lenght) {

        if (lenght == 0) {
            lenght = getRandom(10, 100000);
        }

        byte[] output = new byte[lenght];
        for (int i = 1; i < lenght; i++) {
            output[i] = (byte) ThreadLocalRandom.current().nextInt(
0xFF + 1);
        }

        return output;
    }

    //end data generators
    private int getRandom(int min, int max) {
        return ThreadLocalRandom.current().nextInt(min, max
+ 1);
```

```
    }

    /*
        https://javarevisited.blogspot.com/2013/02/combine-integer-and-s
     */
    public byte[] combine(byte[] a, byte[] b) {
        int length = a.length + b.length;
        byte[] result = new byte[length];
        System.arraycopy(a, 0, result, 0, a.length);
        System.arraycopy(b, 0, result, a.length, b.length);
        return result;
    }
}

/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package HbaseSchemaInference.model;

import HbaseSchemaInference.control.App;
import HbaseSchemaInference.control.HbaseOperations;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.util.Bytes;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;
```

```
/**
 *
 * @author eduardo
 */
public class RawSchema {

    private String namespace, newNamespace;
    private final String rowName = "raw";
    private final HbaseOperations ops;
    private final int byteNum = 1, booleanNum = 2, stringNum
= 4, shortNum = 8,
            charNum = 16, floatNum = 32, integerNum = 64, doubleNum
= 1,
            longNum = 2, blobNum = 4;
    private final App app;

    public RawSchema(String namespace, HbaseOperations ops,
String newNamespace, App app) {
        this.namespace = namespace;
        this.ops = ops;
        this.newNamespace = newNamespace;
        this.app = app;
    }

    public long getRawSchema() {
        Date date = new Date();
        short createNamespace = ops.createNamespace(newNamespace);
        if (createNamespace < 0) {
            return -1;
        } else if (createNamespace == 1) {
            String[] tables = ops.getTables(newNamespace);
            for (String table : tables) {
                ops.deleteTable(namespace, table);
            }
        }
        ArrayList<PutData> data = new ArrayList<PutData>();
        String[] tables = ops.getTables(namespace);
        int tableCount = 0;
        String tableText = "";
        for (String table : tables) {
            tableText = "Tabela: "+table+" ["+(tableCount++)+"/"+ta
```

```
            String[] families = ops.getFamilies(namespace, table);
            ops.createTable(newNamespace, table, families);
            int familyCount = 0;
            String familyText = "";
            for (String family : families) {
                familyText = "Familia: "+family+" ["+(familyCount++)+"/"
                byte[] fam = Bytes.toBytes(family);
                String[] columns = ops.getColumns(namespace,
table, family);
                int columnCount = 0;
                String columnText = "";
                for (String column : columns) {
                    columnText = "colunas: ["+(columnCount++)+"/"+column

                    byte[] col = Bytes.toBytes(column);
                    byte[] value = columnAnalysis(namespace,
table, family, column);
                    //ops.putData(newNamespace, table, rowName,
fam, col, value);
                    app.updateStatus("<html>"+tableText+"<br/>"+familyTe
                    data.add(new PutData(fam, col, value));
                }
            }
            ops.putArrayOfData(newNamespace, table, rowName,
data);
        }
        Date date2 = new Date();
        return date2.getTime() - date.getTime();
    }

    private byte[] columnAnalysis(String namespace, String table,
String family, String column) {
        byte[] type = new byte[]{0x7, (byte) 0xFF};
        boolean fixedLenght = true;
        int lenght = 0;
        byte[] output = new byte[7];
        try {
            ResultScanner scanner = ops.getValuesScan(namespace,
table, family, column);

            for (Result result2 = scanner.next(); result2 !=
```

```
null; result2 = scanner.next()) {
                List<Cell> family_cells = result2.listCells();

                for (Cell family_cell : result2.listCells())
{
                    byte[] rowArray = family_cell.getRowArray();
                    int valueLen = family_cell.getValueLength();
                    if (lenght == 0) {
                        lenght = valueLen;
                    }
                    if (fixedLenght && lenght != valueLen) {
                        fixedLenght = false;
                    }

                    byte[] value = Arrays.copyOfRange(rowArray,
family_cell.getValueOffset(),
                            family_cell.getValueOffset() + family_c
                    byte[] types = getTypes(valueLen, value);
                    type[0] &= types[0];
                    type[1] &= types[1];
                }
            }

        } catch (IOException ex) {
            return null;
        }

        byte[] fLen = Bytes.toBytes(fixedLenght);
        output[0] = fLen[0];
        byte[] len = Bytes.toBytes(lenght);
        output[1] = len[0];
        output[2] = len[1];
        output[3] = len[2];
        output[4] = len[3];
        output[5] = type[0];
        output[6] = type[1];
        return output;
    }

    private byte[] getTypes(int len, byte[] value) {
        byte[] output = new byte[]{0x0, 0x0};
```

```
        switch (len) {
            case 1:
                if (isUtf8Valid(value)) {
                    //string or byte
                    output[1] |= (byteNum + stringNum);

                } else if (value[0] == -1 || value[0] == 0)
{
                    //boolean or byte
                    output[1] |= (byteNum + booleanNum);

                } else {
                    //byte
                    output[1] |= byteNum;
                }
                break;
            case 2:
                if (isUtf8Valid(value)) {
                    //string or short
                    output[1] |= (shortNum + stringNum);
                } else {
                    //short
                    output[1] |= shortNum;
                }
                break;
            case 3:
                //string
                if (isUtf8Valid(value)) {
                    output[1] |= stringNum;
                } else {
                    output[0] |= blobNum;
                }
                break;
            case 4:
                //char or float or integer or string

                if (isUtf8Valid(value)) {
                    output[1] |= stringNum;
                } else if (value[0] == 0 && value[1] == 0 &&
(value[2] != 0 || value[3] != 0)) {
                    output[1] |= charNum;
```

```
                }

                if ((value[0] != -1 && value[0] != 127) || value[1]
>= 0) {
                    output[1] |= floatNum;
                }

                output[1] |= integerNum;
                break;
            case 5:
            case 6:
            case 7:
                //string
                if (isUtf8Valid(value)) {
                    output[1] |= stringNum;
                } else {
                    output[0] |= blobNum;
                }
                break;
            case 8:
                //double or long or string
                if (isUtf8Valid(value)) {
                    output[1] |= stringNum;
                }
                //01111111 11110000
                if ((value[0] != -1 && value[0] != 127) || (value[1
< -16 || value[1] > -1)) {
                    output[0] |= doubleNum;
                }

                output[0] |= longNum;
                break;
            default:
                //string or blob
                if (isUtf8Valid(value)) {
                    output[1] |= stringNum;
                } else {
                    output[0] |= blobNum;
                }

        }
```

```
        return output;

    }

    private boolean isUtf8Valid(byte[] value) {
        int c1Min = 0x1, c1Max = 0x7E;
        int c2Min = 0xC2, c2Max = 0xDF;
        int c3Min = 0xE0, c3Max = 0xEF;
        int c4Min = 0xF0, c4Max = 0xF5;
        int min = 0x80, max = 0xBF;

        for (int i = 0; i < value.length; i++) {
            int val = byteToIntUnsigned(value[i]);

            if ((int) value[i] >= c1Min && val <= c1Max) {
                continue;
            } else if ((i + 1) < value.length && val >= c2Min
&& val <= c2Max) {
                int val1 = byteToIntUnsigned(value[i + 1]);
                if (val1 >= min && val1 <= max) {
                    i++;
                    continue;
                } else {
                    return false;
                }
            } else if ((i + 2) < value.length && val >= c3Min
&& val <= c3Max) {
                int val1 = byteToIntUnsigned(value[i + 1]);
                int val2 = byteToIntUnsigned(value[i + 2]);
                if (val1 >= min && val1 <= max
                        && val2 >= min && val2 <= max) {
                    i += 2;
                    continue;
                } else {
                    return false;
                }
            } else if ((i + 3) < value.length && val >= c4Min
&& val <= c4Max) {
                int val1 = byteToIntUnsigned(value[i + 1]);
                int val2 = byteToIntUnsigned(value[i + 2]);
                int val3 = byteToIntUnsigned(value[i + 3]);
```

```
                if (val1 >= min && val1 <= max
                        && val2 >= min && val2 <= max
                        && val3 >= min && val3 <= max) {
                    i += 3;
                    continue;
                } else {
                    return false;
                }
            } else {
                return false;
            }
        }
        return true;
    }

    public String rawToJSON() {
        String[] tables = ops.getTables(newNamespace);
        String schema = "\"\$schema\": \"http://json-schema.org/dra
                + "  \"definitions\": {"
                + "    \"long\": {"
                + "      \"description\": \"Representation of
a long number\","
                + "      \"type\": \"number\","
                + "      \"minimum\": -2147483648,"
                + "      \"maximum\": 2147483647"
                + "    },"
                + "    \"double\": {"
                + "      \"description\": \"Representation of
a double number\","
                + "      \"type\": \"number\","
                + "      \"minimum\": -1.7E+308,"
                + "      \"maximum\": 1.7E+308"
                + "    },"
                + "    \"float\": {"
                + "      \"description\": \"Representation of
a float number\","
                + "      \"type\": \"number\","
                + "      \"minimum\": 3.4E-38,"
                + "      \"maximum\": 3.4E+38"
                + "    },"
                + "    \"byte\": {"
```

```
                + "      \"description\": \"Representation of
a byte\","
                + "      \"type\": \"number\","
                + "      \"minimum\": -128,"
                + "      \"maximum\": 127"
                + "    },"
                + "    \"blob\": {"
                + "      \"description\": \"Representation of
a binary large object file\","
                + "      \"type\": \"string\","
                + "      \"minLength\": 2"
                + "    },"
                + "    \"short\": {"
                + "      \"description\": \"Representation of
a short number\","
                + "      \"type\": \"number\","
                + "      \"minimum\": -32768,"
                + "      \"maximum\": 32767"
                + "    },"
                + "      \"char\": {"
                + "      \"description\": \"Representation of
a char\","
                + "      \"type\": \"string\","
                + "      \"minLength\": 1,"
                + "      \"minLength\": 1"
                + "    },"
                + "  },"
                + ""
                + "  \"\$id\": \"namespace-" + namespace + "\","
                + "  \"description\": \"Representation of a
hbase namespace\","
                + "  \"type\": \"object\","
                + "\"properties\": {";
        if (tables != null) {
            for (String table : tables) {
                schema += " \"" + table + "\": {"
                        + "     \"description\": \"Representation
of a hbase table\","
                        + "     \"type\": \"object\","
                        + "     \"properties\": {";
                try {
```

```
                    ResultScanner scanner = ops.getTableScan(newNam
table);
                    for (Result result2 = scanner.next(); result2
!= null; result2 = scanner.next()) {
                        List<Cell> family_cells = result2.listCells
                        String family = "";
                        for (Cell family_cell : result2.listCells()
{

                            byte[] rowArray = family_cell.getRowArr
                            byte[] fam = Arrays.copyOfRange(rowArra
family_cell.getFamilyOffset(),
                                    family_cell.getFamilyOffset()
+ family_cell.getFamilyLength());
                            String temp = Bytes.toString(fam);
                            if (!temp.equals(family)) {
                                if (family != "") {
                                    schema += "}"
                                            + "}, ";
                                }
                                family = temp;
                                schema += "        \"" + family
+ "\": {"
                                        + "          \"description\
\"Representation of a hbase family\","
                                        + "          \"type\":
\"object\","
                                        + "          \"properties\"
{";
                            }
                            byte[] col = Arrays.copyOfRange(rowArra
family_cell.getQualifierOffset(),
                                    family_cell.getQualifierOffset(
+ family_cell.getQualifierLength());
                            String column = Bytes.toString(col);
                            schema += "        \"" + column
+ "\": ";

                            ArrayList<String> types = new ArrayList
                            String ref = "\"\$ref\": \"#/definition
```

```
                            byte[] type = Arrays.copyOfRange(rowArray,
family_cell.getValueOffset() + 5,
                                    family_cell.getValueOffset()
+ family_cell.getValueLength());
                            if ((type[1] & booleanNum) != 0)
{
                                types.add("\"type\": \"boolean\"");
                            } else if ((type[1] & byteNum) !=
0) {
                                types.add(ref + "byte\"");
                            }
                            if ((type[1] & stringNum) != 0)
{
                                types.add("\"type\": \"string\"");
                            }
                            if ((type[1] & shortNum) != 0) {
                                types.add(ref + "short\"");
                            }
                            if ((type[1] & charNum) != 0) {
                                types.add(ref + "char\"");
                            }
                            if ((type[1] & floatNum) != 0) {
                                types.add(ref + "float\"");
                            }
                            if ((type[1] & integerNum) != 0)
{
                                types.add("\"type\": \"integer\"");
                            }
                            if ((type[0] & doubleNum) != 0)
{
                                types.add(ref + "double\"");
                            }
                            if ((type[0] & longNum) != 0) {
                                types.add(ref + "long\"");
                            }
                            if ((type[0] & blobNum) != 0) {
                                types.add(ref + "blob\"");
                            }
                            if (types.size() == 1) {
                                schema += " {" + types.get(0)
+ " },";
```

```
                            } else {
                                String choices = "";
                                for (int i = 0; i < types.size();
i++) {

                                    choices += "{" + types.get(i)
+ "},";

                                }
                                schema += " {\"anyOf\":[" +
choices + " ]},";
                            }
                        }
                    }

                } catch (IOException ex) {
                    Logger.getLogger(RawSchema.class.getName()).log
null, ex);
                }
                schema += "\}"
                        + "\},"
                        + "},";
            }
             schema += "} ";
        }
        schema += "} ";
        schema = "{" + schema + "}";
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        JSONParser parser = new JSONParser();
        JSONObject json = null;
        try {
            json = (JSONObject) parser.parse(schema);
        } catch (ParseException ex) {
            Logger.getLogger(RawSchema.class.getName()).log(Level.S
null, ex);
        }

        return gson.toJson(json);

    }

    private int byteToIntUnsigned(byte value) {
        byte[] result = new byte[]{0x0, 0x0, 0x0, 0x0};
```

```
        result[3] |= value;
        return Bytes.toInt(result);
    }

}

/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package HbaseSchemaInference.model;

/**
 *
 * @author eduardo
 */
public class PutData {

    byte[] family, column, value;

    public PutData(byte[] family, byte[] column, byte[] value)
{
        this.family = family;
        this.column = column;
        this.value = value;
    }

    public byte[] getFamily() {
        return family;
    }

    public byte[] getColumn() {
        return column;
    }

    public byte[] getValue() {
        return value;
    }
```

```
}


/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package HbaseSchemaInference.view;

import HbaseSchemaInference.control.App;
import java.awt.Toolkit;
import java.awt.datatransfer.StringSelection;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.Date;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JList;

/**
 *
 * @author eduardo
 */
public class MainView extends javax.swing.JFrame {

    private App app;
    private String[] schemas;

    public MainView() {
        initComponents();
    }
```

```
    public MainView(App main) {
        app = main;

        initComponents();
        getNamespaces();

    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this
method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        namespacesScroll = new javax.swing.JScrollPane();
        avaliableNamespaces = new javax.swing.JList<>();
        jLabel1 = new javax.swing.JLabel();
        refreshNamespaces = new javax.swing.JButton();
        filler1 = new javax.swing.Box.Filler(new java.awt.Dimension(0,
24), new java.awt.Dimension(0, 24), new java.awt.Dimension(32767,
24));
        jSeparator1 = new javax.swing.JSeparator();
        refreshSchemes = new javax.swing.JButton();
        jSeparator2 = new javax.swing.JSeparator();
        jLabel2 = new javax.swing.JLabel();
        schemasScroll = new javax.swing.JScrollPane();
        AvaliableSchemes = new javax.swing.JList<>();
        jLabel3 = new javax.swing.JLabel();
        namespaceLabel = new javax.swing.JLabel();
        newScheme = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        schemaArea = new javax.swing.JTextArea();
        jLabel4 = new javax.swing.JLabel();
        schemaLabel = new javax.swing.JLabel();
```

```
        jLabel5 = new javax.swing.JLabel();
        exportScheme = new javax.swing.JButton();
        jSeparator3 = new javax.swing.JSeparator();
        copyScheme = new javax.swing.JButton();
        statusLabel = new javax.swing.JLabel();
        deleteScheme = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_O

        avaliableNamespaces.setModel(new javax.swing.AbstractListMo
{
            String[] strings = { "testes" };
            public int getSize() { return strings.length; }
            public String getElementAt(int i) { return strings[i];
}
        });
        avaliableNamespaces.setSelectionMode(javax.swing.ListSelect
        namespacesScroll.setViewportView(avaliableNamespaces);

        jLabel1.setText("Namespaces disponiveis:");

        refreshNamespaces.setText("Atualizar");
        refreshNamespaces.addMouseListener(new java.awt.event.Mouse
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                refreshNamespacesMouseReleased(evt);
            }
        });
        refreshNamespaces.addActionListener(new java.awt.event.Acti
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                refreshNamespacesActionPerformed(evt);
            }
        });

        jSeparator1.setForeground(new java.awt.Color(0, 0, 0));
        jSeparator1.setOrientation(javax.swing.SwingConstants.VERTI
        jSeparator1.setFont(new java.awt.Font("Dialog", 0, 24));
// NOI18N
```

```
        refreshSchemes.setText("Atualizar");
        refreshSchemes.addMouseListener(new java.awt.event.MouseAdapter(
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                refreshSchemesMouseReleased(evt);
            }
        });

        jSeparator2.setForeground(new java.awt.Color(0, 0, 0));
        jSeparator2.setOrientation(javax.swing.SwingConstants.VERTICAL);
        jSeparator2.setFont(new java.awt.Font("Dialog", 0, 24));
// NOI18N

        jLabel2.setText("Esquemas:");

        AvaliableSchemes.setSelectionMode(javax.swing.ListSelectionModel
        schemasScroll.setViewportView(AvaliableSchemes);

        jLabel3.setText("Namespace: ");

        namespaceLabel.setText("N");

        newScheme.setText("Gerar Novo");
        newScheme.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                newSchemeMouseReleased(evt);
            }
        });
        newScheme.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                newSchemeActionPerformed(evt);
            }
        });

        schemaArea.setEditable(false);
```

```
        schemaArea.setColumns(20);
        schemaArea.setRows(5);
        jScrollPane1.setViewportView(schemaArea);

        jLabel4.setText("JSON Schema");

        schemaLabel.setText("N");

        jLabel5.setText("Esquema:");

        exportScheme.setText("Exportar JSON Schema");
        exportScheme.addMouseListener(new java.awt.event.MouseAdapt
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                exportSchemeMouseReleased(evt);
            }
        });
        exportScheme.addActionListener(new java.awt.event.ActionLis
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                exportSchemeActionPerformed(evt);
            }
        });

        jSeparator3.setForeground(new java.awt.Color(0, 0, 0));
        jSeparator3.setOrientation(javax.swing.SwingConstants.VERTI
        jSeparator3.setFont(new java.awt.Font("Dialog", 0, 24));
// NOI18N

        copyScheme.setText("Copiar JSON Schema");
        copyScheme.addMouseListener(new java.awt.event.MouseAdapter
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                copySchemeMouseReleased(evt);
            }
        });
        copyScheme.addActionListener(new java.awt.event.ActionListe
{
```

```
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                copySchemeActionPerformed(evt);
            }
        });

        statusLabel.setText("Status:");

        deleteScheme.setText("Excluir");
        deleteScheme.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseReleased(java.awt.event.MouseEvent
evt) {
                deleteSchemeMouseReleased(evt);
            }
        });
        deleteScheme.addActionListener(new java.awt.event.ActionListener
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                deleteSchemeActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(get
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel1)
                        .addGroup(layout.createParallelGroup(javax.swing
                            .addGroup(layout.createSequentialGroup()
                                .addGap(26, 26, 26)
                                .addComponent(jLabel3)
                                .addPreferredGap(javax.swing.LayoutStyle
                                .addComponent(namespaceLabel))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(81, 81, 81)
```

```
                                    .addComponent(jLabel2)
                                    .addGap(296, 296, 296)
                                    .addComponent(jLabel4)))
                            .addPreferredGap(javax.swing.LayoutStyle.Co
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.
                                .addGroup(layout.createSequentialGroup(
                                    .addGap(20, 20, 20)
                                    .addComponent(namespacesScroll,
javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayou
                                    .addPreferredGap(javax.swing.Layout
                                .addGroup(javax.swing.GroupLayout.Align
layout.createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(refreshNamespaces)
                                    .addGap(38, 38, 38)))
                            .addComponent(jSeparator1, javax.swing.Grou
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFE
                            .addGroup(layout.createParallelGroup(javax.
                                .addGroup(layout.createSequentialGroup(
                                    .addGap(24, 24, 24)
                                    .addComponent(statusLabel, javax.sw
210, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(layout.createSequentialGroup(
                                    .addPreferredGap(javax.swing.Layout
                                    .addGroup(layout.createParallelGrou
                                        .addGroup(layout.createSequenti
                                            .addGap(6, 6, 6)
                                            .addComponent(refreshScheme
                                            .addPreferredGap(javax.swin
                                            .addComponent(deleteScheme)
                                            .addPreferredGap(javax.swin
                                            .addComponent(newScheme))
                                        .addComponent(schemasScroll,
javax.swing.GroupLayout.PREFERRED_SIZE, 287, javax.swing.GroupLayou
                            .addPreferredGap(javax.swing.LayoutStyle.Co
15, Short.MAX_VALUE)
                            .addComponent(jSeparator2, javax.swing.Grou
2, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createParallelGroup(javax.
```

```
                                .addGroup(layout.createSequentialGroup()
                                    .addPreferredGap(javax.swing.LayoutStyle
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(copyScheme)
                                    .addGap(26, 26, 26)
                                    .addComponent(exportScheme)
                                    .addGap(94, 94, 94))
                                .addGroup(layout.createSequentialGroup()
                                    .addGap(18, 18, 18)
                                    .addGroup(layout.createParallelGroup(jav
                                        .addGroup(layout.createSequentialGro
                                            .addComponent(jLabel5)
                                            .addPreferredGap(javax.swing.Lay
                                            .addComponent(schemaLabel)
                                            .addGap(0, 0, Short.MAX_VALUE))
                                        .addComponent(jScrollPane1))
                                    .addPreferredGap(javax.swing.LayoutStyle
                            .addComponent(jSeparator3, javax.swing.GroupLayo
12, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(22, 22, 22)))
                    .addComponent(filler1, javax.swing.GroupLayout.PREFERRED
122, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(156, 156, 156))
            );
            layout.setVerticalGroup(
                layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                .addGroup(layout.createSequentialGroup()
                    .addGap(51, 51, 51)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLa
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
layout.createSequentialGroup()
                            .addComponent(filler1, javax.swing.GroupLayout.P
110, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(51, 51, 51))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
layout.createSequentialGroup()
                            .addComponent(namespacesScroll, javax.swing.Grou
203, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
                            .addComponent(refreshNamespaces)
                            .addContainerGap(43, Short.MAX_VALUE))))
```

```
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(layout.createParallelGroup(javax.swing.Gr
                        .addComponent(jSeparator3, javax.swing.GroupLay
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel4)
                            .addGap(3, 3, 3)
                            .addGroup(layout.createParallelGroup(javax.
                                .addComponent(jLabel5)
                                .addComponent(schemaLabel))
                            .addPreferredGap(javax.swing.LayoutStyle.Co
                            .addComponent(jScrollPane1)
                            .addPreferredGap(javax.swing.LayoutStyle.Co
                            .addGroup(layout.createParallelGroup(javax.
                                .addComponent(exportScheme)
                                .addComponent(copyScheme)))
                        .addComponent(jSeparator1)
                        .addComponent(jSeparator2)
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.
                                .addComponent(jLabel1)
                                .addComponent(jLabel2))
                            .addGap(3, 3, 3)
                            .addGroup(layout.createParallelGroup(javax.
                                .addComponent(jLabel3)
                                .addComponent(namespaceLabel))
                            .addPreferredGap(javax.swing.LayoutStyle.Co
                            .addComponent(schemasScroll, javax.swing.Gr
204, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.Co
                            .addGroup(layout.createParallelGroup(javax.
                                .addComponent(refreshSchemes)
                                .addComponent(newScheme)
                                .addComponent(deleteScheme))
                            .addPreferredGap(javax.swing.LayoutStyle.Co
                            .addComponent(statusLabel)
                            .addGap(0, 0, Short.MAX_VALUE)))
                    .addContainerGap())
        );

        pack();
```

```
    }// </editor-fold>//GEN-END:initComponents

    private void refreshNamespacesMouseReleased(java.awt.event.MouseEven
evt) {//GEN-FIRST:event_refreshNamespacesMouseReleased
        getNamespaces();
        statusLabel.setText("");
    }//GEN-LAST:event_refreshNamespacesMouseReleased

    private void refreshSchemesMouseReleased(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_refreshSchemesMouseReleased
        if (refreshSchemes.isEnabled()) {
            selectNamespace();
            statusLabel.setText("");
        }
    }//GEN-LAST:event_refreshSchemesMouseReleased

    private void refreshNamespacesActionPerformed(java.awt.event.ActionE
evt) {//GEN-FIRST:event_refreshNamespacesActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_refreshNamespacesActionPerformed

    private void newSchemeMouseReleased(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_newSchemeMouseReleased
        if (newScheme.isEnabled()) {
            newScheme.setEnabled(false);
            statusLabel.setText("Iniciando a inferência");
            String namespace = namespaceLabel.getText();
            app.newSchema(namespace);
        }
    }//GEN-LAST:event_newSchemeMouseReleased

    private void newSchemeActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_newSchemeActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_newSchemeActionPerformed

    private void exportSchemeMouseReleased(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_exportSchemeMouseReleased
        if (copyScheme.isEnabled()) {
            JFileChooser fc = new JFileChooser();
            File yourFolder = new java.io.File(".");
```

```
            fc.setCurrentDirectory(yourFolder); // start at
application current directory
            fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            int returnVal = fc.showSaveDialog(this);

            if (returnVal == JFileChooser.APPROVE_OPTION) {
                yourFolder = fc.getSelectedFile();
            }
            BufferedWriter writer = null;
            try {
                writer = new BufferedWriter(new FileWriter(yourFold
+ "/" + schemaLabel.getText() + ".json", true));
                writer.append(schemaArea.getText());
                writer.close();
            } catch (IOException ex) {
                Logger.getLogger(MainView.class.getName()).log(Leve
null, ex);
            } finally {
                try {
                    writer.close();
                } catch (IOException ex) {
                    Logger.getLogger(MainView.class.getName()).log(
null, ex);
                }
            }
        }
    }//GEN-LAST:event_exportSchemeMouseReleased

    private void exportSchemeActionPerformed(java.awt.event.ActionE
evt) {//GEN-FIRST:event_exportSchemeActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_exportSchemeActionPerformed

    private void copySchemeMouseReleased(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_copySchemeMouseReleased
        if (copyScheme.isEnabled()) {
            Toolkit.getDefaultToolkit()
                    .getSystemClipboard()
                    .setContents(
                            new StringSelection(schemaArea.getText(
                            null
```

```
                    );
        }
    }//GEN-LAST:event_copySchemeMouseReleased

    private void copySchemeActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_copySchemeActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_copySchemeActionPerformed

    private void deleteSchemeMouseReleased(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_deleteSchemeMouseReleased
        if (deleteScheme.isEnabled()) {
            if (AvaliableSchemes.getSelectedIndex() >= 0) {
            String selected = schemas[AvaliableSchemes.getSelectedIndex(
            app.deleteScheme(selected);
            selectNamespace();
            statusLabel.setText("");
        }else{
            statusLabel.setText("selecione um esquema para excluir");
            }

        }
    }//GEN-LAST:event_deleteSchemeMouseReleased

    private void deleteSchemeActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_deleteSchemeActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_deleteSchemeActionPerformed

    public void updateStatus(String text) {
        statusLabel.setText(text);
    }

    public void newSchema(Date date, long totalTime) {

        String dateF = String.format("%02d:%02d:%02d",
                        TimeUnit.MILLISECONDS.toHours(totalTime),
                        TimeUnit.MILLISECONDS.toMinutes(totalTime),
                        TimeUnit.MILLISECONDS.toSeconds(totalTime));
        updateStatus("Tempo de inferência: "+dateF);
        selectNamespace();
```

```
        AvaliableSchemes.setSelectedValue(date.toString(), true);
        selectScheme();
        newScheme.setEnabled(true);
    }

    private void selectNamespace() {
        if (avaliableNamespaces.getSelectedIndex() >= 0) {
            String selected = avaliableNamespaces.getSelectedValue(
            namespaceLabel.setText(selected);
            getSchemesList(selected);
        }
    }

    private void selectScheme() {
        if (AvaliableSchemes.getSelectedIndex() >= 0) {
            String selected = schemas[AvaliableSchemes.getSelectedI
            schemaLabel.setText(namespaceLabel.getText() + "
- " + AvaliableSchemes.getSelectedValue());
            getSchema(selected);
        }
    }

    private void getSchema(String namespace) {
        String scheme = app.getSchema(namespace);
        schemaArea.setText(scheme);
        exportScheme.setEnabled(true);
        copyScheme.setEnabled(true);

    }

    private void getSchemesList(String namespace) {
        String[] namespaces = app.getSchemes(namespace);

        schemas = Arrays.copyOfRange(namespaces, 0, namespaces.leng
        for (int i = 0; i < namespaces.length; i++) {
            String[] split = namespaces[i].split("_");
            long time = Long.parseLong(split[split.length -
1]);
            Date date = new Date(time);
            namespaces[i] = date.toString();
        }
```

```
        AvaliableSchemes = new JList(namespaces);
        AvaliableSchemes.addMouseListener(new MouseAdapter()
{
            public void mouseClicked(MouseEvent e) {
                selectScheme();
            }
        });
        AvaliableSchemes.setSelectedIndex(-1);
        schemasScroll.setViewportView(AvaliableSchemes);
        refreshSchemes.setEnabled(true);
        deleteScheme.setEnabled(true);
        newScheme.setEnabled(true);
        clearSchemeView();
    }

    private void getNamespaces() {
        String[] namespaces = app.getNamespaces();
        avaliableNamespaces = new JList(namespaces);
        avaliableNamespaces.addMouseListener(new MouseAdapter()
{
            public void mouseClicked(MouseEvent e) {
                selectNamespace();
            }
        });
        avaliableNamespaces.setSelectedIndex(-1);
        namespacesScroll.setViewportView(avaliableNamespaces);
        clearSchemesList();

    }

    private void clearSchemeView() {
        schemaArea.setText("");
        schemaLabel.setText("");
        exportScheme.setEnabled(false);
        copyScheme.setEnabled(false);
    }

    private void clearSchemesList() {
        AvaliableSchemes = new JList();
        AvaliableSchemes.addMouseListener(new MouseAdapter()
{
```

```
            public void mouseClicked(MouseEvent e) {
                selectScheme();
            }
        });
        AvaliableSchemes.setSelectedIndex(-1);
        schemasScroll.setViewportView(AvaliableSchemes);
        namespaceLabel.setText("");
        refreshSchemes.setEnabled(false);
        deleteScheme.setEnabled(false);
        newScheme.setEnabled(false);
        statusLabel.setText("");
        schemas = new String[0];
        clearSchemeView();

    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JList<String> AvaliableSchemes;
    private javax.swing.JList<String> avaliableNamespaces;
    private javax.swing.JButton copyScheme;
    private javax.swing.JButton deleteScheme;
    private javax.swing.JButton exportScheme;
    private javax.swing.Box.Filler filler1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JLabel namespaceLabel;
    private javax.swing.JScrollPane namespacesScroll;
    private javax.swing.JButton newScheme;
    private javax.swing.JButton refreshNamespaces;
    private javax.swing.JButton refreshSchemes;
    private javax.swing.JTextArea schemaArea;
    private javax.swing.JLabel schemaLabel;
    private javax.swing.JScrollPane schemasScroll;
```

```
    private javax.swing.JLabel statusLabel;
    // End of variables declaration//GEN-END:variables
}
```

MainView.form

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.3" maxVersion="1.9" type="org.netbeans.modules.form.for
  <Properties>
    <Property name="defaultCloseOperation" type="int" value="3"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
    <SyntheticProperty name="generateCenter" type="boolean"
value="false"/>
  </SyntheticProperties>
  <AuxValues>
    <AuxValue name="FormSettings_autoResourcing" type="java.lang.Integer
value="0"/>
    <AuxValue name="FormSettings_autoSetComponentName" type="java.lang.B
value="false"/>
    <AuxValue name="FormSettings_generateFQN" type="java.lang.Boolean"
value="true"/>
    <AuxValue name="FormSettings_generateMnemonicsCode" type="java.lang.
value="false"/>
    <AuxValue name="FormSettings_i18nAutoMode" type="java.lang.Boolean"
value="false"/>
    <AuxValue name="FormSettings_layoutCodeTarget" type="java.lang.Integ
value="1"/>
    <AuxValue name="FormSettings_listenerGenerationStyle" type="java.lan
value="0"/>
    <AuxValue name="FormSettings_variablesLocal" type="java.lang.Boolean
value="false"/>
    <AuxValue name="FormSettings_variablesModifier" type="java.lang.Inte
value="2"/>
  </AuxValues>

  <Layout>
    <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0" attributes="0">
```

```
          <Group type="102" attributes="0">
              <Group type="103" groupAlignment="0" attributes="0">
                  <Group type="102" alignment="0" attributes="0">
                      <EmptySpace max="-2" attributes="0"/>
                      <Component id="jLabel1" min="-2" max="-2"
attributes="0"/>
                      <Group type="103" groupAlignment="0" attribut
                          <Group type="102" alignment="0" attribute
                              <EmptySpace min="-2" pref="26"
max="-2" attributes="0"/>
                              <Component id="jLabel3" min="-2"
max="-2" attributes="0"/>
                              <EmptySpace max="-2" attributes="0"/>
                              <Component id="namespaceLabel"
min="-2" max="-2" attributes="0"/>
                          </Group>
                          <Group type="102" alignment="0" attribute
                              <EmptySpace min="-2" pref="81"
max="-2" attributes="0"/>
                              <Component id="jLabel2" min="-2"
max="-2" attributes="0"/>
                              <EmptySpace min="-2" pref="296"
max="-2" attributes="0"/>
                              <Component id="jLabel4" min="-2"
max="-2" attributes="0"/>
                          </Group>
                      </Group>
                      <EmptySpace max="32767" attributes="0"/>
                  </Group>
                  <Group type="102" alignment="0" attributes="0">
                      <Group type="103" groupAlignment="0" attribut
                          <Group type="102" attributes="0">
                              <EmptySpace min="-2" pref="20"
max="-2" attributes="0"/>
                              <Component id="namespacesScroll"
min="-2" pref="140" max="-2" attributes="0"/>
                              <EmptySpace max="-2" attributes="0"/>
                          </Group>
                          <Group type="102" alignment="1" attribute
                              <EmptySpace max="-2" attributes="0"/>
                              <Component id="refreshNamespaces"
```

```
min="-2" max="-2" attributes="0"/>
                                <EmptySpace min="-2" pref="38"
max="-2" attributes="0"/>
                            </Group>
                        </Group>
                        <Component id="jSeparator1" min="-2" max="-2"
attributes="0"/>
                        <Group type="103" groupAlignment="0" attributes="0
                            <Group type="102" attributes="0">
                                <EmptySpace min="-2" pref="24"
max="-2" attributes="0"/>
                                <Component id="statusLabel" min="-2"
pref="210" max="-2" attributes="0"/>
                            </Group>
                            <Group type="102" alignment="0" attributes="0"
                                <EmptySpace type="unrelated" max="-2"
attributes="0"/>
                                <Group type="103" groupAlignment="0"
attributes="0">
                                    <Group type="102" attributes="0">
                                        <EmptySpace min="-2" pref="6"
max="-2" attributes="0"/>
                                        <Component id="refreshSchemes"
min="-2" max="-2" attributes="0"/>
                                        <EmptySpace max="-2" attributes="0
                                        <Component id="deleteScheme"
min="-2" max="-2" attributes="0"/>
                                        <EmptySpace max="-2" attributes="0
                                        <Component id="newScheme"
min="-2" max="-2" attributes="0"/>
                                    </Group>
                                    <Component id="schemasScroll"
min="-2" pref="287" max="-2" attributes="0"/>
                                </Group>
                            </Group>
                        </Group>
                        <EmptySpace pref="15" max="32767" attributes="0"/>
                        <Component id="jSeparator2" min="-2" pref="2"
max="-2" attributes="0"/>
                        <Group type="103" groupAlignment="0" attributes="0
                            <Group type="102" attributes="0">
```

```
                                <EmptySpace max="32767" attributes="0
                                <Component id="copyScheme" min="-2"
max="-2" attributes="0"/>
                                <EmptySpace min="-2" pref="26"
max="-2" attributes="0"/>
                                <Component id="exportScheme" min="-2"
max="-2" attributes="0"/>
                                <EmptySpace min="-2" pref="94"
max="-2" attributes="0"/>
                            </Group>
                            <Group type="102" alignment="0" attribute
                                <EmptySpace min="-2" pref="18"
max="-2" attributes="0"/>
                                <Group type="103" groupAlignment="0"
attributes="0">
                                    <Group type="102" attributes="0">
                                        <Component id="jLabel5"
min="-2" max="-2" attributes="0"/>
                                        <EmptySpace max="-2" attribut
                                        <Component id="schemaLabel"
min="-2" max="-2" attributes="0"/>
                                        <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                                    </Group>
                                    <Component id="jScrollPane1"
max="32767" attributes="0"/>
                                </Group>
                                <EmptySpace max="-2" attributes="0"/>
                            </Group>
                        </Group>
                        <Component id="jSeparator3" min="-2" pref="12
max="-2" attributes="0"/>
                        <EmptySpace min="-2" pref="22" max="-2"
attributes="0"/>
                    </Group>
                </Group>
                <Component id="filler1" min="-2" pref="122" max="-2"
attributes="0"/>
                <EmptySpace min="-2" pref="156" max="-2" attributes="
            </Group>
        </Group>
```

```
    </DimensionLayout>
    <DimensionLayout dim="1">
      <Group type="103" groupAlignment="0" attributes="0">
          <Group type="102" attributes="0">
              <EmptySpace min="-2" pref="51" max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0" attributes="0">
                  <Group type="102" alignment="1" attributes="0">
                      <Component id="filler1" min="-2" pref="110"
max="-2" attributes="0"/>
                      <EmptySpace min="-2" pref="51" max="-2"
attributes="0"/>
                  </Group>
                  <Group type="102" alignment="1" attributes="0">
                      <Component id="namespacesScroll" min="-2"
pref="203" max="-2" attributes="0"/>
                      <EmptySpace max="-2" attributes="0"/>
                      <Component id="refreshNamespaces" min="-2"
max="-2" attributes="0"/>
                      <EmptySpace pref="43" max="32767" attributes="0"/>
                  </Group>
              </Group>
          </Group>
          <Group type="102" attributes="0">
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0" attributes="0">
                  <Component id="jSeparator3" alignment="1"
max="32767" attributes="0"/>
                  <Group type="102" alignment="0" attributes="0">
                      <Component id="jLabel4" min="-2" max="-2"
attributes="0"/>
                      <EmptySpace min="-2" pref="3" max="-2"
attributes="0"/>
                      <Group type="103" groupAlignment="3" attributes="0
                          <Component id="jLabel5" alignment="3"
min="-2" max="-2" attributes="0"/>
                          <Component id="schemaLabel" alignment="3"
min="-2" max="-2" attributes="0"/>
                      </Group>
                      <EmptySpace max="-2" attributes="0"/>
                      <Component id="jScrollPane1" max="32767"
attributes="0"/>
```

```
                    <EmptySpace max="-2" attributes="0"/>
                    <Group type="103" groupAlignment="3" attribut
                        <Component id="exportScheme" alignment="3
min="-2" max="-2" attributes="0"/>
                        <Component id="copyScheme" alignment="3"
min="-2" max="-2" attributes="0"/>
                    </Group>
                </Group>
                <Component id="jSeparator1" alignment="0"
max="32767" attributes="0"/>
                <Component id="jSeparator2" alignment="0"
max="32767" attributes="0"/>
                <Group type="102" alignment="0" attributes="0">
                    <Group type="103" groupAlignment="3" attribut
                        <Component id="jLabel1" alignment="3"
min="-2" max="-2" attributes="0"/>
                        <Component id="jLabel2" alignment="3"
min="-2" max="-2" attributes="0"/>
                    </Group>
                    <EmptySpace min="-2" pref="3" max="-2"
attributes="0"/>
                    <Group type="103" groupAlignment="3" attribut
                        <Component id="jLabel3" alignment="3"
min="-2" max="-2" attributes="0"/>
                        <Component id="namespaceLabel" alignment=
min="-2" max="-2" attributes="0"/>
                    </Group>
                    <EmptySpace max="-2" attributes="0"/>
                    <Component id="schemasScroll" min="-2"
pref="204" max="-2" attributes="0"/>
                    <EmptySpace max="-2" attributes="0"/>
                    <Group type="103" groupAlignment="3" attribut
                        <Component id="refreshSchemes" alignment=
min="-2" max="-2" attributes="0"/>
                        <Component id="newScheme" alignment="3"
min="-2" max="-2" attributes="0"/>
                        <Component id="deleteScheme" alignment="3
min="-2" max="-2" attributes="0"/>
                    </Group>
                    <EmptySpace max="-2" attributes="0"/>
                    <Component id="statusLabel" min="-2" max="-2"
```

```
attributes="0"/>
                        <EmptySpace min="0" pref="0" max="32767"
attributes="0"/>
                    </Group>
                </Group>
                <EmptySpace max="-2" attributes="0"/>
            </Group>
        </Group>
    </DimensionLayout>
  </Layout>
  <SubComponents>
    <Container class="javax.swing.JScrollPane" name="namespacesScroll">
      <AuxValues>
        <AuxValue name="autoScrollPane" type="java.lang.Boolean"
value="true"/>
      </AuxValues>

      <Layout class="org.netbeans.modules.form.compat2.layouts.support.J
      <SubComponents>
        <Component class="javax.swing.JList" name="avaliableNamespaces">
          <Properties>
            <Property name="model" type="javax.swing.ListModel"
editor="org.netbeans.modules.form.editors2.ListModelEditor">
              <StringArray count="1">
                <StringItem index="0" value="testes"/>
              </StringArray>
            </Property>
            <Property name="selectionMode" type="int" value="0"/>
          </Properties>
          <AuxValues>
            <AuxValue name="JavaCodeGenerator_TypeParameters"
type="java.lang.String" value="&lt;String&gt;"/>
          </AuxValues>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JLabel" name="jLabel1">
      <Properties>
        <Property name="text" type="java.lang.String" value="Namespaces
disponiveis:"/>
      </Properties>
```

```
    </Component>
    <Component class="javax.swing.JButton" name="refreshNamespaces"
      <Properties>
        <Property name="text" type="java.lang.String" value="Atuali
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.even
parameters="java.awt.event.MouseEvent" handler="refreshNamespacesMo
        <EventHandler event="actionPerformed" listener="java.awt.ev
parameters="java.awt.event.ActionEvent" handler="refreshNamespacesA
      </Events>
    </Component>
    <Component class="javax.swing.Box\$Filler" name="filler1">
      <Properties>
        <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[32767, 24]"/>
        </Property>
        <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[0, 24]"/>
        </Property>
        <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[0, 24]"/>
        </Property>
      </Properties>
      <AuxValues>
        <AuxValue name="classDetails" type="java.lang.String"
value="Box.Filler.VerticalStrut"/>
      </AuxValues>
    </Component>
    <Component class="javax.swing.JSeparator" name="jSeparator1">
      <Properties>
        <Property name="foreground" type="java.awt.Color" editor="o
          <Color blue="0" green="0" red="0" type="rgb"/>
        </Property>
        <Property name="orientation" type="int" value="1"/>
        <Property name="font" type="java.awt.Font" editor="org.netb
          <Font name="Dialog" size="24" style="0"/>
        </Property>
```

```
      </Properties>
    </Component>
    <Component class="javax.swing.JButton" name="refreshSchemes">
      <Properties>
        <Property name="text" type="java.lang.String" value="Atualizar"/
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.event.Mou
parameters="java.awt.event.MouseEvent" handler="refreshSchemesMouseRelea
      </Events>
    </Component>
    <Component class="javax.swing.JSeparator" name="jSeparator2">
      <Properties>
        <Property name="foreground" type="java.awt.Color" editor="org.ne
          <Color blue="0" green="0" red="0" type="rgb"/>
        </Property>
        <Property name="orientation" type="int" value="1"/>
        <Property name="font" type="java.awt.Font" editor="org.netbeans.
          <Font name="Dialog" size="24" style="0"/>
        </Property>
      </Properties>
    </Component>
    <Component class="javax.swing.JLabel" name="jLabel2">
      <Properties>
        <Property name="text" type="java.lang.String" value="Esquemas:"/
      </Properties>
    </Component>
    <Container class="javax.swing.JScrollPane" name="schemasScroll">
      <AuxValues>
        <AuxValue name="autoScrollPane" type="java.lang.Boolean"
value="true"/>
      </AuxValues>

      <Layout class="org.netbeans.modules.form.compat2.layouts.support.J
      <SubComponents>
        <Component class="javax.swing.JList" name="AvaliableSchemes">
          <Properties>
            <Property name="model" type="javax.swing.ListModel"
editor="org.netbeans.modules.form.editors2.ListModelEditor">
              <StringArray count="0"/>
            </Property>
```

```
            <Property name="selectionMode" type="int" value="0"/>
          </Properties>
          <AuxValues>
            <AuxValue name="JavaCodeGenerator_TypeParameters"
type="java.lang.String" value="&lt;String&gt;"/>
          </AuxValues>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JLabel" name="jLabel3">
      <Properties>
        <Property name="text" type="java.lang.String" value="Namesp
"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JLabel" name="namespaceLabel">
      <Properties>
        <Property name="text" type="java.lang.String" value="N"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JButton" name="newScheme">
      <Properties>
        <Property name="text" type="java.lang.String" value="Gerar
Novo"/>
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.even
parameters="java.awt.event.MouseEvent" handler="newSchemeMouseRelea
        <EventHandler event="actionPerformed" listener="java.awt.ev
parameters="java.awt.event.ActionEvent" handler="newSchemeActionPer
      </Events>
    </Component>
    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
      <AuxValues>
        <AuxValue name="autoScrollPane" type="java.lang.Boolean"
value="true"/>
      </AuxValues>

      <Layout class="org.netbeans.modules.form.compat2.layouts.supp
      <SubComponents>
        <Component class="javax.swing.JTextArea" name="schemaArea">
```

```
          <Properties>
            <Property name="editable" type="boolean" value="false"/>
            <Property name="columns" type="int" value="20"/>
            <Property name="rows" type="int" value="5"/>
          </Properties>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JLabel" name="jLabel4">
      <Properties>
        <Property name="text" type="java.lang.String" value="JSON
Schema"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JLabel" name="schemaLabel">
      <Properties>
        <Property name="text" type="java.lang.String" value="N"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JLabel" name="jLabel5">
      <Properties>
        <Property name="text" type="java.lang.String" value="Esquema:"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JButton" name="exportScheme">
      <Properties>
        <Property name="text" type="java.lang.String" value="Exportar
JSON Schema"/>
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.event.Mou
parameters="java.awt.event.MouseEvent" handler="exportSchemeMouseReleased
        <EventHandler event="actionPerformed" listener="java.awt.event.A
parameters="java.awt.event.ActionEvent" handler="exportSchemeActionPerfo
      </Events>
    </Component>
    <Component class="javax.swing.JSeparator" name="jSeparator3">
      <Properties>
        <Property name="foreground" type="java.awt.Color" editor="org.ne
          <Color blue="0" green="0" red="0" type="rgb"/>
        </Property>
```

```
        <Property name="orientation" type="int" value="1"/>
        <Property name="font" type="java.awt.Font" editor="org.netb
          <Font name="Dialog" size="24" style="0"/>
        </Property>
      </Properties>
    </Component>
    <Component class="javax.swing.JButton" name="copyScheme">
      <Properties>
        <Property name="text" type="java.lang.String" value="Copiar
JSON Schema"/>
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.even
parameters="java.awt.event.MouseEvent" handler="copySchemeMouseRele
        <EventHandler event="actionPerformed" listener="java.awt.ev
parameters="java.awt.event.ActionEvent" handler="copySchemeActionPe
      </Events>
    </Component>
    <Component class="javax.swing.JLabel" name="statusLabel">
      <Properties>
        <Property name="text" type="java.lang.String" value="Status
      </Properties>
    </Component>
    <Component class="javax.swing.JButton" name="deleteScheme">
      <Properties>
        <Property name="text" type="java.lang.String" value="Exclui
      </Properties>
      <Events>
        <EventHandler event="mouseReleased" listener="java.awt.even
parameters="java.awt.event.MouseEvent" handler="deleteSchemeMouseRe
        <EventHandler event="actionPerformed" listener="java.awt.ev
parameters="java.awt.event.ActionEvent" handler="deleteSchemeAction
      </Events>
    </Component>
  </SubComponents>
</Form>
```