Schemas and Types for JSON Data: from Theory to Practice

Mohamed-Amine Baazizi
Sorbonne Université, LIP6 UMR 7606
France
baazizi@ia.lip6.fr

Dario Colazzo
Université Paris-Dauphine, PSL Research University
France
dario.colazzo@dauphine.fr

Giorgio Ghelli
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
ghelli@di.unipi.it

Carlo Sartiani
DIMIE
Università della Basilicata
Potenza, Italy
sartiani@gmail.com

ABSTRACT

The last few years have seen the fast and ubiquitous diffusion of JSON as one of the most widely used formats for publishing and interchanging data, as it combines the flexibility of semistructured data models with well-known data structures like records and arrays. The user willing to effectively manage JSON data collections can rely on several schema languages, like JSON Schema, JSound, and Joi, as well as on the type abstractions offered by modern programming and scripting languages like Swift or TypeScript.

The main aim of this tutorial is to provide the audience (both researchers and practitioners) with the basic notions for enjoying all the benefits that schema and types can offer while processing and manipulating JSON data. This tutorial focuses on four main aspects of the relation between JSON and schemas: (1) we survey existing schema language proposals and discuss their prominent features; (2) we analyze tools that can infer schemas from data, or that exploit schema information for improving data parsing and management; and (3) we discuss some open research challenges and opportunities related to JSON data.

CCS CONCEPTS

• **Information systems** → **Semi-structured data**; • **Theory of computation** → *Type theory*.

KEYWORDS

JSON, schemas, schema inference, parsing, schema libraries

ACM Reference Format:
Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2019. Schemas and Types for JSON Data: from Theory to Practice. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3299869.3314032

1 INTRODUCTION

The last two decades have seen a dramatic change in the data processing landscape. While at the end of the last century data were usually very structured and managed inside relational DBMSs, nowadays they have very different characteristics: they are big, usually semistructured or even unstructured, without a rigid and predefined schema, and hosted and produced in data processing platforms that do not embrace the relational model. In this new scenario, where data come without a schema, and multiple data models coexist, JSON is affirming as a useful format for publishing and exchanging data, as it combines the flexibility of XML with well-known data structures like records and arrays. JSON is currently employed for publishing and sharing data in many application fields and for many different purposes: for instance, JSON is used as the result format for many web site APIs (e.g., Twitter, New York Times), as a common format for the remote interaction of modern web applications (e.g., Facebook's GraphQL is entirely based on JSON), and as a common format for exchanging scientific data as well as public open data (e.g., the U.S. Government's open data platform: `https://www.data.gov`).

Given the wide diffusion of JSON and its use in scientific as well as mainstream applications, the need to directly manipulate JSON data inside applications rapidly emerged. To this aim, a schema language, specifically designed for JSON, has been introduced, but its adoption is not growing at a fast pace, since its specification is somewhat complex and many modern programming and scripting languages, like Swift and TypeScript, directly support JSON data through their own, simple type systems; furthermore, Walmart Labs endowed JavaScript, which is inherently untyped, with a powerful schema language for JSON objects by means of JavaScript function calls, and MongoDB, a very popular NoSQL system capable of directly processing JSON data, defined its own schema language.

In this tutorial proposal, we will present and discuss existing schema and type languages for JSON data, and compare their features; we will also discuss several schema-related tools, with a particular focus on approaches for schema inference. The main aim of this tutorial is to provide the audience, comprising researchers, practitioners, and developers, with the basic notions for enjoying all the benefits that schema and types can offer while processing and manipulating JSON data.

Outline. This *1.5-hour* tutorial is split into four main parts:

(1) **JSON primer**. In this very introductory part of the tutorial, we review the basic notions about JSON together with its JavaScript legacy. We also present a few examples, coming from publicly available datasets, that we will use throughout the remaining parts of the tutorial.
(2) **Schema languages**. In this part of the tutorial we focus on existing schema languages for JSON data collections and discuss their most prominent features. We review JSON Schema [12], Walmart Labs Joi [8], and MongoDB Mongoose [7].
(3) **Schema Tools**. In this part of the tutorial we analyze tools that exploit schema information for improving JSON data processing. We focus on the problem of inferring a meaningful schema for schemaless JSON collections, as well as on the exploitation of schema information for improving data parsing and management. We also review a few libraries supporting schemas for JSON data.
(4) **Future Opportunities**. Finally, we outline open research problems as potential directions for new research in this area.

The preferred duration of this tutorial is 3 hours, but it can also fit a smaller slot of 1.5 hours. In what follows we describe at a very high level the technical content covered in each of the last four aforementioned parts, using the green color to mark the content that will presented in the 3-hour format only.

2 SCHEMA LANGUAGES

In this part of the tutorial we will focus our attention on several schema languages for JSON data, with particular emphasis on JSON Schema [6], Walmart Labs Joi [8], and MongoDB Mongoose [7].

JSON Schema emerged in the academic community and has been developed without a specific programming or scripting language in mind. JSON Schema allows the programmer to specify a schema for any kind of JSON values, and supports traditional type constructors, like union and concatenation, as well as very powerful constructors like negation types.

JSON Schema has already been studied. Indeed, in [22], motivated by the need of laying the formal foundations for the JSON Schema language [6], Pezoa et al. present the formal semantics of that language, as well as a theoretical study of its expressive power and validation problem. Along the lines of [22], Bourhis et al. [18] have recently laid the foundations for a logical characterization and formal study for JSON schema and query languages.

On the contrary, Joi has been developed by Walmart as a tool for (i) creating schemas for JSON objects and (ii) ensuring the validation of objects inside an untyped scripting language like JavaScript; furthermore, to the best of our knowledge, Joi has not been studied so far. Joi only allows the designer to describe the schema for JSON objects, but it still provides the ability to specify co-occurrence and mutual exclusion constraints on fields, as well as union and value-dependent types.

Mongoose [7] has been developed inside the MongoDB system for describing the structure and the behavior of objects stored inside MongoDB, and it features somewhat limited capabilities wrt JSON Schema and Joi, but it supports a form of *any* type.

We will analyze the most prominent features of these languages, and compare their capabilities in a few scenarios.

3 SCHEMA TOOLS

In this part of the tutorial we will present several schema-related tools for JSON data. We will first discuss existing approaches for inferring a schema starting from a dataset and then move to parsing tools that are able to exploit dynamic type information to speed-up data parsing. We will conclude this part with a brief review of existing schema libraries for JSON data.

3.1 Schema Inference

Several schema inference approaches for JSON data collections have been proposed in the past. In [13–15] authors of this tutorial describe a distributed, parametric schema inference approach capable of inferring schemas at different levels of abstraction. In the context of Spark, the Spark Dataframe schema extraction [11] is a very interesting tool for the automated extraction of a schema from JSON datasets; this tool performs schemas in a distributed fashion, but, unlike the technique described in [13, 14], its inference approach is quite imprecise, since the type language lacks union types, and the inference algorithm resorts to `Str` on strongly heterogeneous collections of data. Another Spark-based inference system is Schema-Guru [9], now part of the SnowPlow analytics suite. The peculiar feature of Schema-Guru is its ability to infer a JSON Schema, hence avoiding the use of proprietary languages. Other systems, like Jaql [16], exploit schema information for inferring the output schema of a query, but still require an externally supplied schema for input data, and perform output schema inference only locally on a single machine.

There are also a few inference tools for data stored in NoSQL systems and RDBMSs. Indeed, in the context of NoSQL systems (e.g. MongoDB), recent efforts have been dedicated to the problem of implementing tools for JSON schema inference. A JavaScript library for JSON, called mongodb-schema, is presented in [23]. This tool analyzes JSON objects pulled from MongoDB, and processes them in a streaming fashion; it is able to return quite concise schemas, but it cannot infer information describing field correlation. In [24], a python-based tool is described, called Skinfer, which infers JSON Schemas from a collection of JSON objects. Skinfer exploits two different functions for inferring a schema from an object and for merging two schemas; schema merging is limited to record types only, and cannot be recursively applied to objects nested inside arrays. Finally, Couchbase is endowed with a schema discovery module which classifies the objects of a JSON collection based on both structural and semantic information [4]. This module is meant to facilitate query formulation and select relevant indexes for optimizing query workloads.

3.2 Parsing

There are a few novel parsing tools for JSON data that take into account dynamic type information for improving the efficiency of the applications relying on them.

In a recent work [21], Li et al. describe the Mison JSON parser. Mison creates *on-the-fly* a structural index for each JSON object being parse, with the aim of quickly locating object fields, and performs some *speculative* guesses on the logical position of the fields being queried; the construction of the structural index heavily exploits SIMD vectorization and bitwise parallelism, while Mison speculative guesses rely on a tree structure summarizing frequent patterns in the data. Through these techniques Mison is able to prune parts of the data that are not needed by a given analytics task.

In [17] Bonetta and Brantner present FAD.JS, a *speculative*, JIT-based JSON encoder and decoder designed for the Oracle Graal.js JavaScript runtime. It exploits data access patterns to optimize both encoding and decoding: indeed, FAD.JS relies on the assumption that most applications never use all the fields of input objects, and, for instance, skips unneeded object fields during JSON object parsing.

3.3 Schema Libraries

In the previous parts of the tutorial we illustrated several schema languages for JSON data. These languages are supported by a few libraries targeting different host languages and different backends. In this final section devoted to schema tools we will briefly cover libraries like ajv [1], Everit [5], and schm [10]. ajv is a JavaScript JSON Schema validator based on the `Node.js` framework, while Everit is a Java library targeting JSON Schema. schm, finally, is JavaScript library supporting Mongoose that can be used together with Joi and even other schema libraries.

4 FUTURE OPPORTUNITIES

We finally discuss several open challenges and opportunities related to JSON schemas.

Schema Inference and ML. While all schema inference approaches covered in the previous part of the tutorial are based on traditional techniques, a recent work by Gallinucci et al. [19] shows the potential benefits of ML approaches in schema inference; furthermore, ML-based inference techniques have already been used for non-JSON data, like in [20]. Hence, a promising research direction is to understand how these methods can be efficiently applied to large collections of data and whether they can overcome some limitations of previous approaches.

Schema-Based Data Translation. While JSON is very frequently used for exchanging and publishing data, it is hardly used as internal data format in Big Data management tools, that, instead, rely on formats like Avro [2] and Parquet [3]. When input datasets are heterogeneous, schemas can improve the efficiency and the effectiveness of data format conversion. Therefore, a major opportunity is to design schema-aware data translation algorithms that are driven by schema information and use it to improve the quality of the translation.

5 INTENDED AUDIENCE AND COVERAGE

Our goal is to make this tutorial accessible to all participants in SIGMOD and PODS interested in understanding the foundations and applications of schemas and types for JSON data processing. Therefore, we will not assume any background in JSON schema languages, but will introduce them starting from the roots, giving broad coverage of many of the key ideas, making it appropriate for developers, graduate students seeking new areas to study, and researchers active in the field alike.

6 DIFFERENCES WITH OTHER VERSIONS OF THE TUTORIAL

A preliminary, 1.5-hour version of this tutorial has been presented at EDBT 2019. Wrt that proposal, here we widened the scope of the proposal, by covering more topics and more approaches, but also slightly shifted the focus from theory to practice, with particular emphasis on schema tools and libraries.

7 BIOGRAPHICAL SKETCHES

Mohamed-Amine Baazizi (Ph.D.) is an assistant professor at Sorbonne Université. He received his PhD from Université of Paris-Sud and completed his postdoctoral studies in Télécom ParisTech. His research focuses on exploiting schema information for optimizing the processing of semi-structured data.

Dario Colazzo (Ph.D.) is Full Professor in Computer Science at LAMSADE - Université Paris-Dauphine. He received his PhD from Università di Pisa, and he completed his postdoctoral studies at Università di Venezia and Université Paris Sud. His main research activities focus on static analysis techniques for large-scale data management and analysis.

Giorgio Ghelli (Ph.D.) is Full Professor in Computer Science, at Università di Pisa. He was Visiting Professor at École Normale Supérieure Paris, at Microsoft Research Center, Cambridge (UK), and at Microsoft Co. (Redmond, USA), member of the W3C XML Query Working Group, member of the board of the EAPLS. He worked on database programming languages and type systems for these languages, especially in the fields of object oriented and XML data models.

Carlo Sartiani (Ph.D.) is an assistant professor at Università della Basilicata. He received his PhD from Università di Pisa, and he completed his postdoctoral studies at Università di Pisa. He worked on database programming languages and data integration systems, with particular emphasis on XML query languages and database systems. His current research activity mostly focuses on big and semistructured data.

REFERENCES

[1] ajv. https://github.com/epoberezkin/ajv.

[2] Apache Avro. https://avro.apache.org.

[3] Apache Parquet. https://parquet.apache.org.

[4] Couchbase auto-schema discovery. https://blog.couchbase.com/auto-schema-discovery/.

[5] Everit. https://github.com/everit-org/json-schema.

[6] JSON Schema language. http://json-schema.org.

[7] Mongoose. https://mongoosejs.com.

[8] Object schema description language and validator for JavaScript objects. https://github.com/hapijs/joi.

[9] Schema-Guru. https://github.com/snowplow/schema-guru.

[10] schm. https://github.com/diegohaz/schm.

[11] Spark Dataframe. https://spark.apache.org/docs/latest/sql-programming-guide.html.

[12] The JSON Schema Language. http://json-schema.org/.

[13] Mohamed Amine Baazizi, Houssem Ben Lahmar, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2017. Schema Inference for Massive JSON Datasets. In *EDBT '17*.

[14] Mohamed Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2017. Counting types for massive JSON datasets. In *Proceedings of The 16th International Symposium on Database Programming Languages, DBPL 2017, Munich, Germany, September 1, 2017*. 9:1–9:12.

[15] Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2019. Parametric schema inference for massive JSON datasets. *The VLDB Journal* (2019). https://doi.org/10.1007/s00778-018-0532-7

[16] Kevin S. Beyer, Vuk Ercegovac, Rainer Gemulla, Andrey Balmin, Mohamed Y. Eltabakh, Carl-Christian Kanne, Fatma Özcan, and Eugene J. Shekita. 2011. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. *PVLDB* 4, 12 (2011), 1272–1283.

[17] Daniele Bonetta and Matthias Brantner. 2017. FAD.js: Fast JSON Data Access Using JIT-based Speculative Optimizations. *PVLDB* 10, 12 (2017), 1778–1789. http://www.vldb.org/pvldb/vol10/p1778-bonetta.pdf

[18] Pierre Bourhis, Juan L. Reutter, Fernando Suárez, and Domagoj Vrgoc. 2017. JSON: Data model, Query languages and Schema specification. In *PODS '17*. 123–135.

[19] Enrico Gallinucci, Matteo Golfarelli, and Stefano Rizzi. 2018. Schema profiling of document-oriented databases. *Inf. Syst.* 75 (2018), 13–25.

[20] Alon Y. Halevy, Flip Korn, Natalya Fridman Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. 2016. Goods: Organizing Google's Datasets. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. 795–806.

[21] Yinan Li, Nikos R. Katsipoulakis, Badrish Chandramouli, Jonathan Goldstein, and Donald Kossmann. 2017. Mison: A Fast JSON Parser for Data Analytics. *PVLDB* 10, 10 (2017), 1118–1129.

[22] Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. 2016. Foundations of JSON Schema. In *WWW '16*. 263–273.

[23] Peter Schmidt. 2017. mongodb-schema. Available at https://github.com/mongodb-js/mongodb-schema.

[24] scrapinghub. 2015. Skinfer. Available at https://github.com/scrapinghub/skinfer.