Schema-Based JSON Data Stores in Relational Databases

Lubna Irshad, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Li Yan, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Zongmin Ma, Nanjing University of Aeronautics and Astronautics, Nanjing, China
https://orcid.org/0000-0001-7780-6473

ABSTRACT

JSON is a simple, compact and light weighted data exchange format to communicate between web services and client applications. NoSQL document stores evolve with the popularity of JSON, which can support JSON schema-less storage, reduce cost, and facilitate quick development. However, NoSQL still lacks standard query language and supports eventually consistent BASE transaction model rather than the ACID transaction model. This is very challenging and a burden on the developer. The relational database management systems (RDBMS) support JSON in binary format with SQL functions (also known as SQL/JSON). However, these functions are not standardized yet and vary across vendors along with different limitations and complexities. More importantly, complex searches, partial updates, composite queries, and analyses are cumbersome and time consuming in SQL/JSON compared to standard SQL operations. It is essential to integrate JSON into databases that use standard SQL features, support ACID transactional models, and has the capability of managing and organizing data efficiently. In this article, we empower JSON to use relational databases for analysis and complex queries. The authors reveal that the descriptive nature of the JSON schema can be utilized to create a relational schema for the storage of the JSON document. Then, the powerful SQL features can be used to gain consistency and ACID compatibility for querying JSON instances from the relational schema. This approach will open a gateway to combine the best features of both worlds: the fast development of JSON, consistency of relational model, and efficiency of SQL.

KEYWORDS

JSON, JSON Schema, NoSQL, Relational Database Systems, Schema-Less

INTRODUCTION

Web services are the means to exchange information on a mobile, search engine, enterprise application and many more, thought formats like XML (Extensible Markup Language) and JSON (Java Script Object Notation). For serialization, data interchange format plays a great role in terms of the rate of data transfer and performance (Helland, 2017). Structure of JSON is similar to the type model of many programming languages, essentially Java Scripting (JS) that makes it flexible, easy to use and
independent text format. It is simple and requires no prior knowledge to acquire and practice. JSON is a format that fills a particular niche to integrate multiple services across many platforms. XML and JSON, both being semi-structured and hierarchal data model for data exchange, are popular serialization techniques in Web development. XML as compared to JSON is heavy, complex and requires additional libraries to exchange data. Its model consists of hierarchal complex tags and requires more bytes for data transfer even for a small task. JSON is light weighted, compact and close to many programming languages. It is one of its kind of emerging data-interchange format as compared to XML and many others like Atom, RDF (Ma, Lin, Yan, Zhao, 2018; Ma, Jia, Cheng & Angryk, 2016), REBOL, Gellish, YAML and so on.

JSON document consists of objects that constitute of attributes/keys (string type), value (String, Number, Boolean and NULL), arrays and objects. An object is a collection of pairs (attributes & values) and a pair can again be a JSON object (Bourhis, Reutter, Suárez & Vrgoč, 2017). JSON document can have hierarchical data like nested objects and arrays. Being simple, easy and light weighted, JSON has become a format of choice for most Web services. JSON especially adores for storing temporary data (like filling the form on Web) and exchanging information between servers and clients. Note that it cannot be used for permanent storage, data analysis, and processing of complex queries\(^1\). So, database support for storing and querying JSON came into existence (Liu, 2019; Irshad, Ma, & Yan, 2019; Junkkari et al., 2016; Hu & Dessloch, 2015). Although JSON can work standalone, its database support provides secure, easy and fast processing of information. This situation just likes XML for databases (Fong & Shiu, 2012). In addition, it is difficult to manage data sharing by multiple users in the document but it is easy in databases. Two types of database models support JSON data management, which are relational database management system (RDBMS) and NoSQL (Not only SQL) (Liu, 2019; Irshad, Ma, & Yan, 2019). We categorize them on the basis of their data structure, model, data organization, and transaction and querying mechanism.

NoSQL database encompasses document databases, key-value pair stores, columnar stores and graph formats stores (Ma, Capretz, & Yan, 2016). JSON document stores belong to the category of NoSQL document databases that evolve with the popularity of JSON. NoSQL data stores are gaining popularity but still lack powerful standard query language. Although, as compared to many document stores, the query language of MongoDB is easy and close to SQL, it requires MapReduce for aggregation and complex queries. CouchDB provides better consistency but has complex query language. Learning a new language every time as per the need of application/project is a cumbersome and time taking task. Standardization of query language for all NoSQL document stores is the biggest demand at present. NoSQL document stores, CouchDB and MongoDB claim to provide ACID (Atomicity, Consistency, Isolation, Durability) transaction model in a coming version. First, the ACID model is implemented on a single document level only and secondly, it is not fully tested and available yet. BASE transaction model is not suitable for all applications, where things will be eventually consistent as compared to the ACID model.

For traditional RDBMS, it is compulsory to design schema carefully before storing/adding data in the relational tables, creating relations among tables and then querying data (Liu, Hammerschmidt, & McMahon, 2014). It enforces relationships among tables using entity-relationship model (Chen, 2002) by breaking down entities into tables and relations (Liu, Hammerschmidt, & McMahon, 2014). Good schema design helps for fast and efficient querying. Advocates of NoSQL document stores claim that schema maintenance costs a lot for rapidly growing data while proponents of RDBMS argue that schema-less development results in long-terms issues (Tahara, Diamond, & Abadi, 2014). NoSQL databases facilitate semi-structured and unstructured data that may fluctuate time to time while relational databases deal with structured data. NoSQL does not have a strong powerful and generalized query language while RDBMS have a native strong SQL language for fast efficient querying. NoSQL follows “schema later or never” approach while RDBMS follows the relational data model (Melton, 2003) based on “schema earlier, data afterward “approach (Liu et al., 2016). The relational database strongly follows the ACID model while the NoSQL databases follow BASE principles (Chandra,
Related Content

Interactive Indexing of Documents with a Multilingual Thesaurus
www.igi-global.com/chapter/interactive-indexing-documents-multilingual-thesaurus/9203?camid=4v1a

Informational and Computational Equivalence in Comparing Information Modeling Methods
Keng Siau (2004). Journal of Database Management (pp. 73-86).
www.igi-global.com/article/informational-computational-equivalence-comparing-information/3306?camid=4v1a
An Alternative Fit through Problem Representation in Cognitive Fit Theory
Hock Chuan Chan, Suparna Goswami and Hee-Woong Kim (2012). Journal of Database Management (pp. 22-43).
www.igi-global.com/article/alternative-fit-through-problem-representation/65540?camid=4v1a

Web Services, Service-Oriented Computing, and Service-Oriented Architecture: Separating Hype from Reality
www.igi-global.com/article/web-services-service-oriented-computing/3390?camid=4v1a