|  |
| --- |
| University of Western Ontario, CS4411B FW16 |
| Academic Paper Review |
| Closing the Functional and Performance Gap Between SQL and NoSQL |

|  |
| --- |
| David Cosman  2-6-2017 |

Review: Closing the functional and Performance Gap between SQL & NoSQL

Organizations typically have to choose between the rigid but powerful relational model that SQL operates on or the agile and quick data insertion NoSQL paradigm that trades flexibility for reliability. No solution is perfect for every application, instead each offering strengths over each other. The core strengths of NoSQL database languages are their accommodating nature; because there are no keys to specify, and because the system has no defined rules, many types of data can be concatenated and inserted as strings without having to deal with the rigid structure of SQL databases based on the Relational model. On the other hand, a schema identifies a concise structure that allows for quicker data access over NoSQL databases. Data solutions of the 21st century, such as MongoDB, have began doing a mix of both to bring in the best possible sides into one package. This review paper focuses on the JSON DataGuide presented by Liu et al through OSON [1], a self-contained query compatible binary format to encode JSON documents, similar to the BSON format used by MongoDB [2].

The authors believe that the flexibility provided by the NoSQL data model is important in addressing a diverse amount of data sources. This can include video files, embedded documents, and unstructured text. However, the relational schema still proves an important role in database systems, particularly the use of SQL to deliver quick data analysis through declarative language. As a result, the DataGuide model derives a schema from collections, similar to the MongoDB querying philosophy. The authors call this a Flexible Schema Data Management (FSDM) paradigm, where “data is storable, indexable and queryable without upfront schema definition” [1]. The concept of DataGuide writes without a schema present while emulating the traversal of queries as if they were going through a schema, thanks to OSON binary format encoding. OSON is essentially a compact representation of the data, organized in trees, vectors (or arrays), and values (or scalars). The implementation is well explained referencing previous work as the build up to the DataGuide; for example, the way in which NoSQL accesses JSON documents to grab data, which the authors point to the lengthy consumption of resources string matching algorithms can take. It takes assumed knowledge from the reader to be familiar with such procedures, as a comparison to the relational model is not given until section 6 figure 3 of the paper, three sections later. Despite the well written explanations, the missing references to figures or tables that would avoid implied previous knowledge from the reader detract from the reader’s attention, and can cause page flipping to find what the authors refer to. Despite this, the related work is laid out as the groundwork that built up to engineering a flexible binary formatting system to bridge Relational and NoSQL queries.

Even with some small implications imposed on the reader, the paper follows a cohesive outline. The authors present the JSON DataGuide conceptually, explaining the tree structure implemented and outlining an example table with a PurchaseOrder set of data in JSON document format. The introduction of different fields further complicates the data to show how the model transforms the attributes into a tree like structure, filled with root nodes that connect to children containing either arrays of values (expanding the database tree to grow deeper) or a new hierarchy that becomes a sibling of an existing node (expanding the database tree to grow wider). The tree forms the hierarchy that is used to emulate a schema access, which enables relational query languages such as SQL to be capable of indexing through the data, while the data is stored in binary format to create a flat document similar to that used by NoSQL models. The DataGuide is presented as a powerful model that brings compatibility between JSON and SQL while stabilizing the amount of time taken to perform queries. This is another one of the key strengths of the paper, where the structure of the flat document is directly tied to a tree model to represent both a schema and a quick accessible document.

The OSON model is very similar to that of BSON, storing data as byte ranges to represent and access it quicker. But OSON differs in the fact that it doesn’t need to read expensive fields of text, instead enabling the reader to jump inexpensively to possible locations in the document. This is done through the tree structure shown in figure 2 that hierarchically defines the next child nodes and by keeping a byte with a hash\_id for each child node, allowing the reader can then narrow down where to begin string matching the document. One disadvantage is that this takes slightly more data, because the files are now keeping one extra value per record. But the difference in performance is better because the tree can now be iterated by using a hash function, once the JSON document has been parsed and divided, and applying the given query. The OSON model is also flexible, providing the option to load on-disk or from memory. The reason is to make it accessible across devices, although it still cuts into the performance time when loaded on disk, and the load on memory can be significantly expensive if the dataset is large. To the authors benefit, however, they are very clear that the purpose is not to surpass the relational model, but to bridge the functional and performance gap between both database methodologies. Exceptions also exist; only minimal OSON is loaded in memory in order to run any SQL or JSON query faster. The performance evaluation is better on average in nearly every case. A remarkable amount of test cases were pushed through to generate results and comparsions. The NOBENCH examples in particular are impressive, using around ten thousand records to run test queries and returning them in less than 10 seconds as shown in Figure 5. The storage size is comparable to that of JSON, and less costly than BSON since it avoids interpretive byte non-terminal characters to index types of data. However, the queries being used become confusing when the test cases reference Q10 and so on. Two tables are in fact being referenced here; Table 9 and Table 13, but both which start at Q1 with their queries.

This is a fantastic paper showing that data can Some opinion of your own concerning the paper. It addresses the criticism that MongoDB chair directed at Oracle for abandoning the “heart and soul” of their consumers [3]. The OSON model is further more efficient in terms of storage than BSON used by MongoDB, while still maintaining flexibility in data types and even capable of being faster in some instances. This system will be successful when the future work is addressed and the system can tolerate object instances with different inherent types. In particular, the compatibility between the relational query model and the agile environment can greatly enhance the performance of systems that rely on static queries and potentially replace outdated legacy databases.

# References

|  |  |
| --- | --- |
| [1] | Z. H. Lui, B. Hammerschmidt, D. McMahon, Y. Lu and H. J. Chang, "Closing the Functional and Performance Gap Between SQL and NoSQL," in *SIGMOD*, San Francisco, 2016. |
| [2] | MongoDB, "MongoDB Architecture Guide," MongoDB, 2016. |
| [3] | M. Flamm, "Asked & Answered: Dev Ittycheria, MongoDB," *Crain's New York Business,* vol. 33, no. 1, p. 8, 2017. |