RelationalJSON, An Enriched Method to Store and Query JSON Records

Ankit Bharthan
Compro Technologies Pvt Ltd
E/88, Ashok Vihar,
Phase-1, Delhi, India

Devesh Bharathan
PayU India
E/88, Ashok Vihar,
Phase-1, Delhi, India

ABSTRACT
Storing JSON documents in a relational database is a favorable solution because relational database are advanced and scale very well and they have the advantage that in a relational database management system (RDBMS) database and organized data can coexist making it possible to build application that involves both kind of data with little effort. In this paper, we propose an algorithm schema named RelationalJSON that translates JSON documents to relational database according to anticipated storing structure. The steps and algorithm are giving in details to describe how to use the storing structure to storage and query JSON documents in RDBMS. Then we report our experimental results on a real database to show the performance of our method in some sorts.

General Terms
Efficient data retrieval.

Keywords
JSON, Relational Database, RDBMS, SQL.

1. INTRODUCTION
Today's data exchange between organizations has becoming perplexing because of the differences in data format and semantics of the meta-data which used to describe the data. Now a days' JSON emerged as a major standard for representing data on World Wide Web while the dominant storage mechanism for structured data is the relational databases, which has been an efficient tool for storing, searching, retrieving data from different collection of data. The ability to map JSON data in relational databases is difficult mission and challenging in the world of all IT organization so there is a need to develop an interfaces and tools for mapping and storing JSON data in relational databases.

1.1 JSON
Short for JavaScript Object Notation, JSON is a lightweight data-interchange format that is easy for humans to read and write, and for machines to parse and generate. JSON is based on the object notation of the JavaScript language. However, it does not require JavaScript to read or write because it is a text format that is language independent. JSON notation contains these basic elements:
Objects: Objects begin and end with curly braces ({}).
Object Members: Members consist of strings and values, separated by colon (:). Members are separated by commas.
Arrays: Arrays begin and end with braces and contain values. Values are separated by commas.
Values: A value can be a string, a number, an object, an array, or the literals true, false or null.
Strings: Strings are surrounded by double quotes and contain Unicode characters or common backslash escapes.

1.2 Relational Databases
Today, the dominant storage mechanism for structured enterprise data is the relational database, which has proven itself an efficient tool for storing, searching for, and retrieving information from massive collections of data. Relational databases specialize in relating individual data records grouped by type in tables. Developers can join records together as needed using SQL (Structured Query Language) and present one or more records to end-users as meaningful information. The relational database model revolutionized enterprise data storage with its simplicity, efficiency, and cost effectiveness. Relational databases have been prevalent in large corporations since the 1980s, and they will likely remain the dominant storage mechanism for enterprise data in the predictable future. Despite these strengths, relational databases lack the flexibility to impeccably integrate with other systems, since this was not historically a requirement of the database model (Reed, D. 2008). In addition, although relational databases share many similarities, there are enough differences between the major viable implementations to make developing applications to integrate multiple products difficult. Among the challenges are differences in data types, varying levels of conformance to the SQL standard, exclusive extensions to SQL, and so on.

2. PROBLEM DESCRIPTION
For the storage of JSON document, the key issue is transmuting the tree structure of a JSON document into tuples in relational tables. Nowadays, there are more and more data presented as JSON document, the need of storing them persistently in a database has increased rapidly while the native–JSON databases usually have limited support for relational databases. In recent years, with the popularity of relational databases (RDB), approaches based on RDB to store and manipulate JSON data as relational tables but still there is need to manage JSON data and relational data seamlessly with similar storage and retrieval efficiencies simultaneously. JSON and Relational databases cannot be kept separately because JSON is becoming the universal standard data format for the representation and exchanging the information whereas most existing data lies in RDBMS and their power of data proficiencies cannot be degraded so the solution to this problem a new efficient methods for storing JSON documents in relational database is required.. A new efficient method for storing JSON document in relational database is recommended in this paper to face these problems.

3. THE PROPOSED PROCESS

3.1 JSON Data

JSON is built on two structures:
First, A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
Second, An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are general data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.
In JSON, they take on these forms:
An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).
It is an instance of JSON document contains information about students as follows:

```
{
 "Personal": {
  "Student": [
   {
    "type": "Online",
    "Name": "Ankit",
    "Id": "21357",
    "Age": "22"
   },
   {
    "type": "Full Time",
    "Name": "Devesh",
    "Id": "22134",
    "Age": "24"
   },
   {
    "type": "Full Time",
    "Name": "Bharthan",
    "Id": "87688",
    "Age": "28"
   }
  ]
 }
}
```

Fig 1: JSON Document

3.2 The Tree Structure of JSON Document

In this section we represented the tree structure of JSON document in Fig 2 with labeling

3.3 Relational JSON Structure

3.3.1 RelationalJSON Structure

Each and every JSON can be describing as a JSON tree. In this figure the squares are the elements and the ovals are the attributes of the elements. A generated JSON tree has been shown in the figure. Every element or attributes are identified by a moniker (number).

3.3.2 Algorithm

JSON document can be stored in relational database, in this paper, MYSQL by use of above two tables. In this paper algorithms are proposed to store JSON document into relational database.

3.3.3 Example 1

In this structure when an element or type associates with its signature it also represents its parent element. We add document name in association with the id to be able to add multiple JSON file in the storage. Figure 2 represents the storage of the JSON file associated with its signature. For every element there will have a signature associated with it and there will also have a parent's signature associated with it. In table 1: fieldName represents the name of the node; id represents the id of the node which is the PK. And finally parentID represents the parent id of the node. As document name don't have any parent id so the id of the document name and parent id of the document name is same that has been shown in the table 2.

Table 1. Field Structure

fieldName	id	parentID
Personnel.JSON	1	1
Personnel	2	1
Student	3	2
Type	4	3
Name	5	3
Id	6	3
Age	7	3
Student	8	2
Type	9	8
Name	10	8
Id	11	8
Age	12	8
Student	13	2
Type	14	13
Name	14	13
Id	15	13
Age	16	13
Student	17	13

Table 2 Field Value

fieldID	Value	type
4	Online	A
5	Ankit	E
6	21357	E
7	22	E
9	Full Time	A
10	Devesh	E
11	22134	E
12	24	E
14	Full Time	A
15	Bharthan	E
16	87688	E
17	28	E

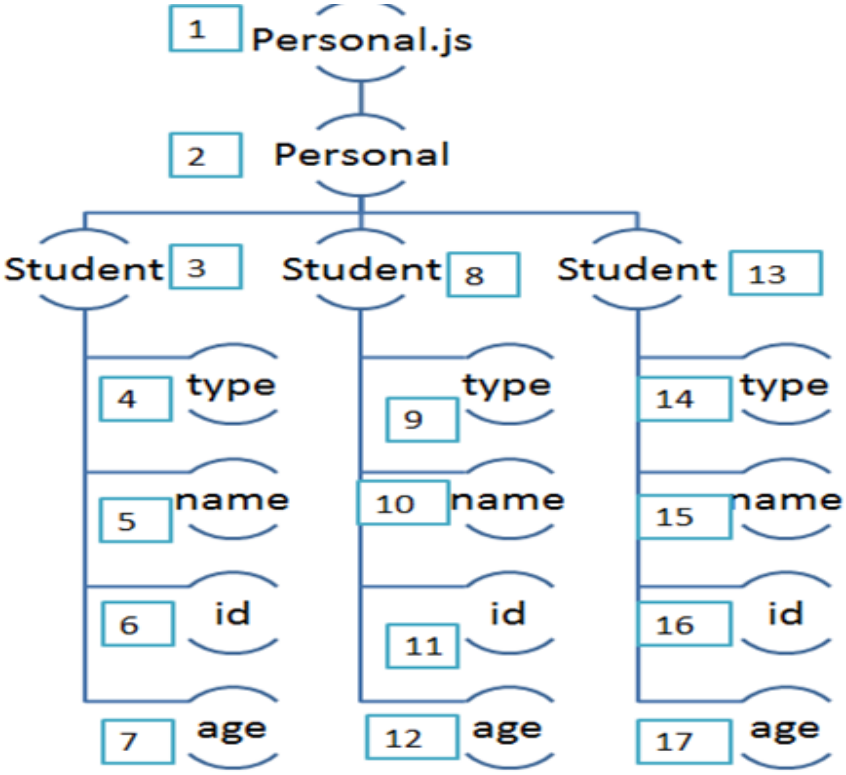

Fig 2: JSON Tree Structure with fieldIDs

In table 2, we represent the value associated with the elements or type. In RelationalJSON structure there is no need to store the path value or path structure as it will be determine recursively by its parent id. In Table 1: fieldName is the name of the tag, where Id is the parent key. In Table 2: fieldID presents the Table 1 id thus fieldIDis the foreign key. In Table 2 fieldID only represents the elements which contain a value and the value represents on the value column. And the type 'A' denoted to the attribute and 'E' denoted to the element.

4. THE ANALYSIS OF RESEARCH

This section discusses the experimental results of storage, parsing and query performance of JSON document using RelationalJSON method. All the experiments were conducted on a Pentium Intel i5 CPU
3.00 GHz with 2 GB RAM 320GB hard disk. We used Windows XP SP2, java 1.6 SDK and MYSQL 5.1 as the DBMS for storing and retrieving JSON document using RelationalJSON method as structure independent mapping approaches. We have implemented data loader for RelationalJSON using SAX and DOM parsers. From the results, the proposed method can be used as an efficient way for storing and queering JSON data in relational database.

4.1 Database Size:

The database size for JSON document in Fig. 1 using RelationalJSON method is given in show that by this storage method we can reduce not only the size of database requirement of the labelling of node, but also the number of tables.

4.2 Parsing Time:

The time of parsing the JSON document using RelationalJSON method is faster because it uses Document Object Model (DOM) parsing technique. Using DOM it read the nodes and corresponding child node.

4.3 Insertion Time

The insertion time of JSON document in Fig.1 using RelationalJSON method is given in table 3. As seen in the table 3, the RelationalJSON is fast and the reason could be that the data is stored in only two tables in this method.

Table 3. Performance Analysis

Insertion Time	1.177 Seconds
Database Size	0.001663 MB
Parse Time	0.053 Seconds

4.4 JSON Validation

RelationalJSON method also included JSON validation package. Before parsing whole JSON document, it checks if the JSON file has valid structure and grammar or not. It also shows the error on the JSON file with specific line number.

5. INFERENCE

RelationalJSON, a general storage method for JSON document using relational database is proposed in this paper. RelationalJSON adopts the model-mapping method to store

JSON document in relational database, to decompose the tree structure into nodes and store all information of nodes in relational database according to the node types by recursive way. It can deal with any documents no matter whether it has fixed outline or not. By using this method we can reduce the database size require to store the JSON document into relational database. The storing algorithm of JSON document into relational database was also given in the paper, and examined the accuracy of it by using the JSON document in performance section. Exploiting the actual JSON document evaluated the performance of storing JSON document into relational database by using our method.

6. REFERENCES

[1] Florescu, D., D. Kossman, 1999. Storing and querying JSON data using an RDBMS, IEEE Data Engineering Bulletin.

[2] Hasan Zafari, Keramat Hasami, M . Ebrahim Shiri, 2010. Xlight, an Efficient Relational Schema to Store and Query JSON Data. In proceeding of the IEEE International conference in Data Store and Data Engineering, pp: 254-257.

[3] Tatarinov, I., S. Viglas, K. Beyer, et al., 2002. Storing and querying ordered JSON using a relational database system, in Proceedings of the ACM SIGMOD.

[4] Tian, F., D. DeWitt, J. Chen, C. Zhang, 2002.The design and performance evaluation of alternative JSON storage strategies, ACM Sigmod Record

[5] Florescu, D., D. Kossman, 1999. Storing and querying JSON data using an RDBMS, IEEE Data Engineering Bulletin.

[6] M.A. Kashem, Abu Sayed Chowdhury, Rupam Deb, and Moslema Jahan, Query Optimization on Relational Databases for Supporting Top-k Query Processing Techniques 2010 JCIT, ISSN 2078-5828 (PRINT), ISSN 2218-5224 (ONLINE), VOLUME 01, ISSUE 01 2013

[7] Majid Khan and M. N. A. Khan, Exploring Query Optimization Techniques in Relational Databases, International Journal of Database Theory and Application Vol. 6, No. 3, June, 2013

[8] Zhang Yu, Research of Conversion Method of Entity Object and JSON Data, The 2nd International Conference on Computer Application and System Modeling (2012)