DATABASE FRAMEWORK

Galmulle Gamaralalage Uthpala Damindu Bandara

IT13118778

Dissertation submitted in partial fulfilment of the requirements for the B.Sc. Special Honors Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

September 2016

**DECLARATION**

“I declare that this is my own work and this dissertation does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belied it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.”

Also, I hereby grant to Sri Lanka Institute of Information Technology the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”

Signature: Date: 09/01/2016

**ABSTRACT**

Sudden requirement changes in software products while development process at the run is a major issue nowadays. Changes in single line of code might lead to unexpected catastrophic events. In every data centric products, database plays a major role as a persistence unit. Considering a project, with relational database at the start of the project with sudden requirement change to some other NoSQL database is the main area we address.

In this research, we propose Benzene, a practical framework for accessing document databases. Benzene is a universal database access layer that acts between the client application and the database framework, using a native library for each specific supported programming language. These libraries contain methods to read, write and update data which take conditions and data as parameters and are transferred to the proposed application in JSON format. The query converter takes these JSON formatted data as inputs and converts it in to SQL queries. Then it executes the query and retrieves data. The retrieved data is converted back to JSON format by Data Converter and send it to the client application. Client application can use this JSON data directly as it is or can convert it to a preferred native format (data type). Complex queries can be handled by custom methods. Users can create custom methods using the GUI provided by the database framework. The schema manager can be used to create the database schema. Schema can be designed using the Schema designer. When designing the schema data types are given using generic types. This schema is stored as a XML file locally in the server. The schema is converted to SQL queries by the Schema Converter when applying the designed schema to the actual database. When converting the schema, generic types are converted in to specific data types of the selected database engine.

**ACKNOWLEDGEMENT**

First of all, I would like to thank the supervisor of Benzene database framework for being with the team for almost one year of time.

**Table of Contents**

**List of Tables**

[Table 1: Differences of Proposed System vs. Existing System xiv](#_Toc460447960)

[Table 2: Similarities in Proposed System vs. Existing System xiv](#_Toc460447961)

**List of Figures**

[Figure 1: Object structure of JSON xi](#_Toc460447901)

[Figure 2: Array structure of JSON xi](#_Toc460447902)

[Figure 3: Value structure of JSON xii](#_Toc460447903)

[Figure 4: String structure of JSON xii](#_Toc460447904)

[Figure 5: Number structure of JSON xiii](#_Toc460447905)

**List of Abbreviations**

**INTRODUCTION**

* 1. **Background Context**

A NoSQL (originally referring to "non SQL" or "non-relational") database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but did not obtain the "NoSQL" moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages. [1]

There are so many plus points a developer or group of developers would select a NoSQL database over a Relational database. Motivations for this NoSQL approach include, simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables. [1] The problem with relational model is that it has some scalability issues that is performance degrades rapidly as data volumes increases. This led to the development of a new data model like NoSQL. Though the concept of NoSQL was developed a long time ago, it was after the introduction of database as a service that it gained a prominent recognition. Because of the high scalability provided by NoSQL, it was seen as a major competitor to the relational database model. Unlike RDBMS, NoSQL databases are designed to easily scale out as and when they grow. Most NoSQL systems have removed the multi-platform support and some extra unnecessary features of RDBMS, making them much more lightweight and efficient than their RDMS counterparts. The NOSQL data model does not guarantee ACID properties (Atomicity, Consistency, Isolation and Durability) but instead it guarantees BASE properties (Basically Available, Soft state, Eventual consistency).It is in compliance with the CAP (Consistency, Availability, Partition tolerance) theorem. [2] Listed below are the key extraction of the identified advantages of NoSQL over RDBMS’s.

* Data model domain is wide compared to RDBMS’s.
* Scalable.
* Efficient and flexible.
* Contribution to the research and development of NoSQL is getting higher with time.
* Has a strong community.

Apart from the advantages of NoSQL over RDBMS, still there are several disadvantages that might lead client application developers to choose RDBMS or ORDBMS. Listed below are the disadvantages of NoSQL.

* There’s no standard query language.
* Difficulties in maintaining.
* Does not guarantee ACID properties.
* Does not have a well-defined structure.
* Security vulnerabilities because of the document structures.

Considering the NoSQL nature, furthermore we can categorize it to some classes. There are 5 classes in existence identified currently. [2]

* Key-Value store databases.
* Column-Oriented databases.
* Document store databases.
* Graph databases.
* Object oriented databases.

The most commonly used words when talking about database management systems are DDL and DML. However Data Definition Language (DDL) is completely out of the scope in the NoSQL domain. A data manipulation language (DML) is a family of syntax elements similar to a computer programming language used for selecting, inserting, deleting and updating data in a database. Performing read-only queries of data is sometimes also considered a component of DML. A popular data manipulation language is that of Structured Query Language (SQL), which is used to retrieve and manipulate data in a relational database. [3] Data manipulation language comprises the SQL data change statements, which modify stored data but not the schema or database objects. Manipulation of persistent database objects, e.g., tables or stored procedures, via the SQL schema statements, rather than the data stored within them, is considered to be part of a separate data definition language. In SQL these two categories are similar in their detailed syntax, data types, expressions etc., but distinct in their overall function. [3]

NoSQL does not use SQL (Structured Query Language) which is the most commonly used query language by relational databases as its query language. Also NoSQL does not have a standard query language. [2] The need of querying is still there but the nature of JSON (JavaScript Object Notation) structure as the data storing structure helps NoSQL in a tremendous way.

JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute –value pairs. It is the most common data format used for asynchronous browser/server communication (AJAJ), largely replacing XML (eXtensible Markup Language) which is used by AJAX. JSON is language-independent data format since it derives from JavaScript, but as of 2016, code to generate and parse JSON-format data is available in many programming languages. The official Internet media type for JSON is application/json. [4]

JSON is built on two structures

* 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. [5]
* An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence. [5]

These are universal 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. [5] Listed below are the forms that JSON takes.

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). [5]

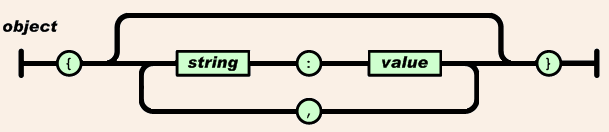


Figure 1: Object structure of JSON

Source: <http://www.json.org/>

An array is an ordered collection of values. An array begins with [ (left bracket) and ends with ] (right bracket). Values are separated by , (comma). [5]

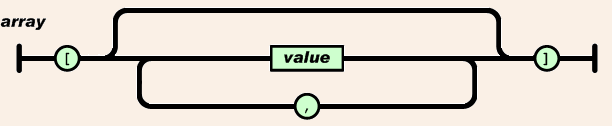


Figure 2: Array structure of JSON

Source: <http://www.json.org/>

A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested. [5]

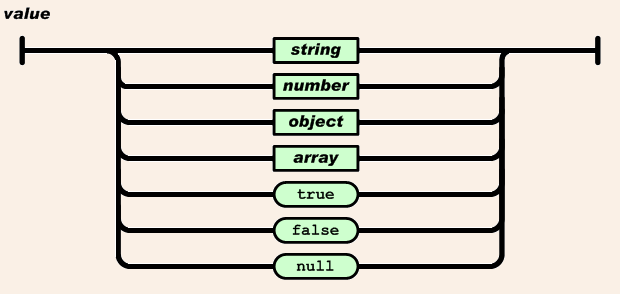


Figure 3: Value structure of JSON

Source: <http://www.json.org/>

A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string. [5]

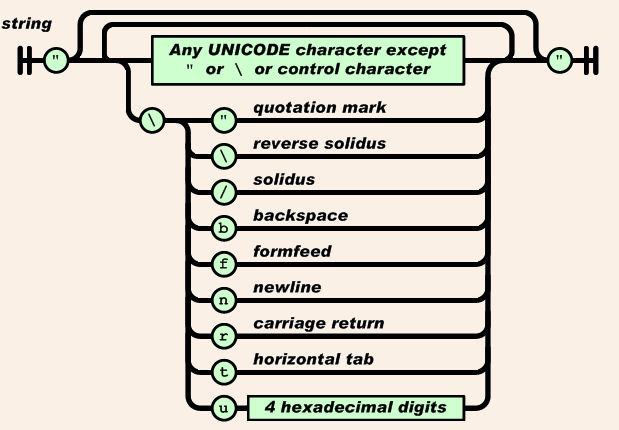


Figure 4: String structure of JSON

Source: <http://www.json.org/>

A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used. [5]

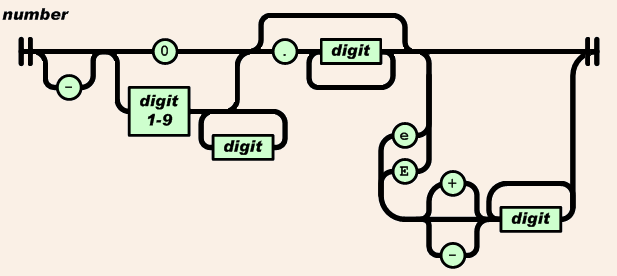


Figure 5: Number structure of JSON

Source: <http://www.json.org/>

Because of the JSON object nature, a client application developer can perform CRUD operations he/she wishes using a third party application. Object Relational Mapping (ORM) comes to rescue when there’s a situation involving a relational database as a persistence provider.

Object-relational mapping in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to construct their own ORM tools. [6]

However, many popular database products such as SQL database management systems (DBMS) can only store and manipulate scalar values such as integers and strings organized within tables. The programmer must either convert the object values into groups of simpler values for storage in the database (or convert them back upon retrieval), or only use simple scalar values within the program. Object-relational mapping implements the first approach. The heart of the problem involves translating the logical representation of the objects into an atomized form that is capable of being stored in the database, while preserving the properties of the objects and their relationships so that they can be reloaded as objects when needed. If this storage and retrieval functionality is implemented, the objects are said to be persistent. [6] Indeed there are solutions currently in the market for mapping between relational and object.

* 1. **Research Gap**

When developing enterprise applications or any other data centric applications, NoSQL databases act a tremendous role. Different vendors work on different types of NoSQL database systems and release multiple versions. These vendors implement and present their own way of querying data from/to their database management systems. This comes in handy when a client application developer master one specific NoSQL database system with one specific programming language. Currently existing NoSQL database management systems provide interfaces to perform CRUD operations facilities with different programming domains. Along with these systems, there are third party applications that fill the void and support the application developer. Most of these applications have GUIs to support the developers.

Table 1: Differences of Proposed System vs. Existing System

|  |  |
| --- | --- |
| **Properties of existing systems** | **Properties of proposed system** |
| Support one specific programming language. | Supports Java and C# languages. |
| Application code needs to be modified when there’s any change in database management system. | Application code doesn’t necessarily need to be modify if the modification is either of two NoSQL database systems we support. |
| Support custom function generations in some vendor’s products for limited language domains. | Supports custom function generation which is language or database independent for proposed language and database domain |
| Support complex database operations. | Supports limited database operations. |
| Doesn’t map database schema to XML schema. | Map database schema to XML schema. |
| Support one specific NoSQL database management system. | Supports MongoDB and OrientDB. |
| Doesn’t support altering database schema. | Supports altering the database schema according to the changes in XML schema. |
| Processed database results which contains objects cannot further processed in the Same instance. | Final JSON file can be organized with details of objects up to a pre-configured level. |

Table 2: Similarities in Proposed System vs. Existing System

|  |  |
| --- | --- |
| **Properties of existing systems** | **Properties of proposed system** |
| Map database schema to XSD. | Map database schema to XSD. |

* 1. **Research Problem**

NoSQL as a JSON based database type did an immense impact on, then most popular relational databases and object relational databases even when NoSQL was a concept. With the advancements of JavaScript and JavaScript based technologies such as AJAX, JSON and MEAN stack, NoSQL emerged quickly. Research and development projects on NoSQL and strong worldwide community is the main reason of this now popular NoSQL database management systems.

Data getting added to persistence units day by day, it was a

* 1. **Research Objectives**