Indonesian Journal of Electrical Engineering and Computer Science
Vol. 4, No. 1, October 2016, pp. 203 ~ 214
DOI: 10.11591/ijeecs.v4.i1.pp203-214

Efficiency of JSON approach for Data Extraction and Query Retrieval

Mohd Kamir Yusof*, Mustafa Man
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, 22200 Besut, Terengganu, Malaysia
*Corresponding author, e-mail: mohdkamir@unisza.edu.my

Abstract

Students' Information System (SIS) in Universiti Sultan Zainal Abidin (UniSZA) handles thousands of records on the information of students, subject registration, etc. Efficiency of storage and query retrieval of these records is the matter of database management especially involving with huge data. However, the execution time for storing and retrieving these data are still considerably inefficient due to several factors. In this contribution, two database approaches namely Extensible Markup Language (XML) and JavaScript Object Notation (JSON) were investigated to evaluate their suitability for handling thousands records in SIS. The results showed JSON is the best choice for storage and query speed. These are essential to cope with the characteristics of students' data. Whilst, XML and JSON technologies are relatively new to date in comparison to the relational database. Indeed, JSON technology demonstrates greater potential to become a key database technology for handling huge data due to an increase of data annually.

Keywords*: JSON, Query Retrieval, Relational Database, XML*

1. Introduction

Students' Information System (SIS) is currently being used by Universiti Sultan Zainal Abidin (UniSZA), Terengganu, Malaysia in handling and managing the students' records, registration, etc. Students' record in (UniSZA), has shown a dramatic increase annually due to the students' intakes that take place twice a year in some Malaysian higher institutions including UniSZA. Currently, (UniSZA) houses for more than 25,000 students records and is expected to grow steadily in the range of 20%-30% annually. Traditional current approach that have been widely practiced is by implementing the work on how to handle and manage these data. The traditional approach is called relational database. In the relational database, each record is stored in the table format. Relational database have proven records for providing efficient, correctness of data, eliminate inconsistency of data and cost-effective management of structured data [1-4]. The SQL query language is also used predominantly in a huge number of applications. SQL query provides a simple tool for data retrieval [5]. However, the major drawback of relational database is pre-design the extract field structures of data which is needed to ensure the consistency of data through process of database normalization [6].

Relational database model also is not practical for certain forms of data that require a lot of fields in handling different types of data involved. Whereas most of the data field are indeed left unused due to the nature of the data. Once this problem happens, the performance storage and retrieval become poor and inefficient. A suitable database approach is required to deal with this issue. In computer science, manipulating data through programming is not enough but native database system should work best, needs efficient storage, management and retrieval of data [3], [8-9]. Currently, XML approach is used for handling huge data as alternative approach compared to relation database approach. Based on previous researchers, XML can be used for standard data representation in information storage and exchange [10, 11]. The performance of query retrieval based on experiments from previous researchers shows XML is faster compared with other approaches [6, 12]. However, because of data growing, the researchers are still looking for better approach compared to XML and relational.

In this research, JSON is proposed for handling huge data. Dataset from SIS is used for experimental purposes. The performance of JSON approach with compared to XML approach. The comparisons are made from the following aspects: data extraction performance, query

Received May 23, 2016; Revised August 28, 2016; Accepted September 19, 2016

performance, scalability, flexibility and extensibility. The rest of this contribution is organized as follows: Section II gives an overview of relational database, XML and JSON. Section III describes the data structure using relational database, XML and JSON. Section IV discusses the two database approaches concerned based on experimental results and our experience in the development. Finally, a conclusion is given in Section V.

2. Background

In the past few years, there have been a dramatic increase of students' data at UniSZA. This institution is facing with problem to store and manage huge data storage. The data increase yearly due to the students' intake which take place twice a year in some higher institutions in Malaysia. SIS is used to handle and manage these records by using relation database approach. This problem is well known to be called big data issue. Big data is large and complex datasets collected from digital and conventional sources [13]. In traditional approach, data is stored and managed in relational database approach. In relational database, the data is represented in a database as a set of tables [14]. Each table has a name and contains a special top row and a finite number of data rows. Number of data depends on the number of rows in the table. Process of gathering, analyzing and reporting educational big data becomes more difficult and complicated when involve with huge data. Storing, querying and retrieving process are also seen to be considerably inefficient when involved with this huge data. A lot of research lately are focusing on the developing methodologies or approaches as an alternative in storing and manage the educational data especially when involved with huge data. Other approaches for data storing and retrieving must be more efficient and extensible compared to relational database approach.

XML is one of alternative approaches for storing and managing the data especially when involves with huge data. XML is widely been accepted as the relevant standardization for representing and exchange the data on the Web [15, 16]. In XML document is often built based on their given schemas such as Document Type-Definitions (DTDs) or XML schemas in exchange the data. LibSyD [17], IGPIP [18], AX-InCoDa [19], COVAX [20], EC-XAMAS [21], TIScover [22], ERP System [23], Novel Approach [24], Chemical Data Integration [25] and Web Services [26] are all examples of the system that are using XML approach. XML provides the functionality to access data from different data sources efficiently. XML also provides reliability, scalability, high performance indices, concurrency control and other advanced functionalities [15].

JSON is our proposed approach for data storing and retrieving especially involved with huge data. JSON schema is introduced in this research in mapping process to different data sources. JSON is designed to be a data exchange language which is human readable and easy for computer to parse and use [27]. It is alternative technique compared to XML due to its relative simplicity and compactness [28]. JSON provides significant performance over XML, which requires extra libraries to retrieve data from Document Object Model (DOM) objects. JSON is estimated to parse up to one hundred times faster than XML in modern browser.

JSON is hopefully able to become as an alternative approach for database system instead of relational database approach and XML approach. The performance for data extraction and query retrieval between XML and JSON will be tested in order to evaluate which approach is better.

3. Method

In this section, three types of different database approaches are thoroughly discussed. There are relational database, XML and JSON. The limitations of the relational database is determined in order to come out with an alternative method. The second and third approaches as the alternative method for relational database are XML and JSON.

3.1. Ralational Database for SIS

In traditional approach, SIS is designed and developed based on relational database. In this approach, the data is organized and managed in tables. Database modelling for SMS can be defined as a following definition:

Definition 1: Let, SIS database, DB = {D_1, D_2, D3…D_n}, where D_1 until D_n is number of tables in database of SMS. D1 until Dn is subset for DB.

Example:

DB ∈ {student, faculty, program, student, state}.

Definition 2: Let A_1, A_2….A_n be attributes name with associated domain D_1, D_2….Dn then R (A_1: D_1, A_2: D_2, A_n: D_n) in relation schema. A relation this is a set of n-tuples (d_1, d_2…d_n) where $d_i \in D_i$.

Example:

Given the sets

StudentID = {012017, 010528, 010529, 010530}
ProgramCode = {611, 213, 212, 412}
FacultyCode = {03, 04, 04, 01}

Then r = {(012017, 611, 03), (010528, 213, 04), (010529, 212, 04), (010530, 412, 01)} is a relation over StudentID x ProgramCode x FacultyCode.

Definition 3: An element of t ∈ r(R) is called a tuple (or row). Table 1, Table 2 and Table 3 shown how the data is stored in tables and their relation.

Table 1. Student's Records

Student ID	Name	ProgramCode	Faculty Code	Nationality	Gender	State	Result
012017	MOHD ASSHAARI BIN SAMSUDIN	611	03	01	L	10	3.29
010528	ROSMA BINTI JASNEY	213	04	01	P	05	2.46
010529	RAFEAH BINTI HAMIL	212	04	01	P	03	3.11
010530	RABIHAH SHAHIZAN BINTI ROSLAN	412	01	01	P	11	2.85

Table 2. Program

programCode	programName
112	Diploma in Islamic Studies (*Usuluddin*)
212	Diploma in Marketing
611	Diploma in English Language Teaching
711	Diploma in Arabic Language Education
411	Diploma in Information Technology
412	Diploma in Information Technology (Multimedia)
213	Diploma in Finance

Table 3. Faculty

facultyCode	facultyName
01	Faculty of Informatics
02	Faculty of Bioinformatics
03	Faculty of Language
04	Faculty of Management, Business and Accounting

In order to retrieve these data, selection operation is needed. The purpose of selection operation is to search and retrieve the data based on queries.

Definition 4: Select operation can be defined as σp (r) := {t | t ∈ r and P(t)} where r is a relation and P is a formula in propositional calculus.

By executing all operations in Table 3, user is able to view the results after searching and retrieving process is done. The major drawback in relational database is that the number of rows can grow considerably even for modest number of records. In our case study, UniSZA exhibits 10, 000 records per year and this record definitely will increase yearly. These records involve both undergraduate and postgraduate students. By increasing the data, the queries performance is considerably inefficient in term of time because of challenging tasks such as to

handle complicated query and data extraction. That way, this contribution evaluates the performance of two approaches as an alternative to relational database. The performance concerns the level of efficiency, scalability and usability.

Table 3. Selection operations

Query	Statement	Operation
I	To retrieve results which is gender equal to "Male"	σGender = 'M'(StudentRecords)
II	To retrieve results which is gender equal to "Female" and GPA > "3.00"	πName (σGender='M'(StudentRecords)) – πName (σGPA>'3.00'(StudentRecords))
III	To retrieve results which is result between 2.5 and 3.5, and state equal to Terengganu	πName (σGPA >'2.5'(StudentRecords)) – πName (σGPA >'3.5'(StudentRecords)) – πName (σState='03'(StudentRecords))
IV	To retrieve result which is program equal to "Diploma in Information Technology" or "Diploma in Information Technology (Multimedia)" or program equal to "Diploma in Marketing" or "Diploma in Finance", and GPA equal or more than "2.5"	πName (σStudentRecords.ProgramCode=Program.programCode (σProgramCode='411'(Program))) ¬ πName (σStudentRecords.ProgramCode=Program.programCode (σProgramCode='412'(Program))) ¬ πName (σStudentRecords.ProgramCode=Program.programCode (σProgramCode='212'(Program))) ¬ (σStudentRecords.ProgramCode=Program.programCode (σProgramCode='213'(Program))) – πName (σGPA>'2.5'(StudentRecords))

3.2. XML approach for modelling SIS

The first step in XML approach is to create a schema. XML is needed to create before extracting the data from Database Management System (DBMS) (i.e. MySQL). Figure 1 shows the standard syntax in XML. XML syntax is needed to write based on type of data prior extraction from DBMS.

```
<root>
<child>
        <subchild>…….</subchild>
</child>
</root>
```

Figure 1. XML Syntax

Figure 2 shown the XML schema before extracting the data from DBMS.

```
<StudentRecords>
<Item>
        <StudentID>…</StudentID>
        <Name>…</Name>
        <Nationality>…</Nationality>
        <Gender>…</Gander>
        <State>…</State>
        <Result>…</Result>
</Item>
</ StudentRecords >
```

Figure 2. XML schema

In order to extract the data from DBMS, algorithm is required to come out in XML file. Figure 3 shows the algorithm for extracting the data.

Input : A = {StudentRecords, Item},
B = {StudentID, Name, Nationality, Gender, State, CGPA},
C = Array ('A', 'B', 'C','D'),
i = 0
Output : XML document (XML)

Steps

1. Read set A
2. Create element StudentRecords
3. Read i
4. IF (i < count (C))
5. Create element Item
6. Read set B
7. Create element StudentID
8. StudentID → A_i; close element StudentID
9. Create element Name
10. Name → B_i; close element Name
11. Create element Nationality
12. Nationality → C_i; close element Nationality
13. Create element Gender
14. Gender → D_i; close element Gander
15. Create element State
16. State → E_i; close element State
17. Create element Result
18. CGPA → B_i; close element Result
19. Close element Item
20. XML.=XML
21. Write i=i+1; Go to step 4
22. Else {go to step 24; }
23. Display XML
24. Exit

Figure 3. Algorithm XML to extract the data

```
<?xml version="1.0" encoding="utf-8"?>
<StudentRecords>
<Item>
<StudentID>012017</StudentID>
<Name>MOHD ASSHAARI BIN SAMSUDIN</Name>
<Nationality>01</ Nationality >
<Gender>L</ Gender >
<State>10</State>
< Result >3.29</ Result >
</Item>
<Item>
<StudentID>010528</StudentID>
<Name>ROSMA BINTI JASNEY</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>05</State>
< Result >2.46</ Result >
</Item>
<Item>
<StudentID>010529</StudentID>
<Name>RAFEAH BINTI HAMIL</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>03</State>
< Result >3.11</ Result >
</Item>
<Item>
<StudentID>010530</StudentID>
<Name>RABIHAH SHAHIZAN BINTI ROSLAN</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>11</State>
< Result >2.85</Result>
</Item>
:
:
:
:
<Item>
<StudentID>010533</StudentID>
<Name>HASSANUL ABIDDIN BIN YUSOF</Name>
<Nationality>01</ Nationality >
<Gender>L</Gender>
<State>14</State>
<Result>2.60</Result>
</Item>
</StudentRecords>
```

Figure 4. XML document (StudentRecords.xml)

Figure 4 shows the XML document after implementation of algorithm in Figure 3.

After XML document (StudentRecords.xml) is created, this document can be accessed by any application because XML is independent platform. Figure 5 shows the algorithm to read the XML document. Four types of different queries are executed and the results are discussed in experimental studies and discussions sections.

```
<?xml version="1.0" encoding="utf-8"?>
<StudentRecords>
<Item>
<StudentID>012017</StudentID>
<Name>MOHD ASSHAARI BIN SAMSUDIN</Name>
<Nationality>01</ Nationality >
<Gender>L</ Gender >
<State>10</State>
< Result >3.29</ Result >
</Item>
<Item>
<StudentID>010528</StudentID>
<Name>ROSMA BINTI JASNEY</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>05</State>
< Result >2.46</ Result >
</Item>
<Item>
<StudentID>010529</StudentID>
<Name>RAFEAH BINTI HAMIL</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>03</State>
< Result >3.11</ Result >
</Item>
<Item>
<StudentID>010530</StudentID>
<Name>RABIHAH SHAHIZAN BINTI ROSLAN</Name>
< Nationality >01</ Nationality >
< Gender >P</ Gender >
<State>11</State>
< Result >2.85</Result>
</Item>
:
:
:
<Item>
<StudentID>010533</StudentID>
<Name>HASSANUL ABIDDIN BIN YUSOF</Name>
<Nationality>01</ Nationality >
<Gender>L</Gender>
<State>14</State>
<Result>2.60</Result>
</Item>
</StudentRecords>
```

Input : StudentRecords.xml (XML), Query (Q), Item, Key
Output : Date set (X)

Steps

1. Read Q
2. Load XML
3. For each (XML → Item & Key as Value)
4. Read Value
5. X.=X
6. Repeat step 3 & 4
7. Display X
8. Exit

Figure 5. Algorithm to read the XML document (StudentRecords.xml)

The efficiency and scalability of XML approach will discuss in next section.

3.3. JSON approach for modelling SIS

JSON file has a standard format or schema. Figure 1 shows the standard JSON file. Three main elements involved in the JSON file. There are title, type, and properties. Properties is similar with attributes. The properties can be one or more than one. Based Figure 6, a_1 until a_n is attribute for object O.

Definition 5: Let, JSON file, A = {S, O, X}, where S is represent title, O is represent object name, and X is represent properties or attributes.

Definition 6: Each O can has more than one object, which is a_i until a_n is number of object in JSON file.

$$F(G) = O \leftarrow \{a_1, a_2, a_3 \ldots a_n\}$$

Definition 7: Each X can has one or more properties, which is a_i until n_i is properties of X.

$$F(H) = X \leftarrow \{b_1, b_2, b_3 \ldots b_n\}$$

```
{
        “Title”: “Schema”
        “type”: “O”
        “properties”: {
                “b1”: {
                        “type”: “string”
                },
                “b2”: {
                        “type”: “string”
                },
                :
                :
                “bn”: {
                        “type”: “string”
                }
        },
}
```

Figure 6. JSON schema

After the JSON schema is created, data from DBMS is loaded and converted into JSON format. Figure 7 shows the algorithm to load and convert the data into the JSON format.

Input : i = 0; M = Array (O, X)
Output : JSON format **(**K)

Steps

1. Read i
2. IF (i < count(M))
3. Read O_i
4. Read X_i
5. $K = K.Q_i.X_i$
6. i=i+1; go to Step 2
7. ELSE {go to Step 7}
8. Display (K)
9. Exit

Figure 7. Algorithm XML to extract the data

After the data is extracted from DBMS using the algorithm in Figure 7, the data should display in JSON format. Figure 8 shows the data in JSON format.

[{"StudentID":"012017","Name":"MOHD ASSHAARI BIN SAMSUDIN","Nationality":"01","Gender":"L","State":"10","Result":"3.29"},{"StudentID":"010528","Name":"ROSMA BINTI JASNEY","Nationality":"01","Gender":"P","State":"05","Result":"2.46"},{"StudentID":"010529","Name":"RAFEAH BINTI HAMIL","Nationality":"01","Gender":"P","State":"03","Result":"3.11"},{"StudentID":"010530","Name":"RABIHAH SHAHIZAN BINTI ROSLAN","Nationality":"01","Gender":"P","State":"11","Result":"2.85"}, …. , {"StudentID":"010533","Name":"HASSANUL ABIDDIN BIN YUSOF","Nationality":"01","Gender":"L","State":"14","Result":"2.60"}
]

Figure 8. JSON format (StudentRecords.xml)

Figure 9 shows the algorithm to read data from JSON format (StudentRecords.json).

Input : StudentRecords.xml (A), Query (Q), i=0, Value
Output : Date set (X)

Steps

1. Read Q
2. Read A
3. Read i
4. IF (i < count (A))
5. X=X.Value[i]
6. i++;
7. Repeat Step 4
8. ELSE {go to Step 9}
9. Display X
10. Exit

Figure 9. Algorithm to read the JSON document (StudentRecords.json)

The efficiency and scalability of JSON approach is discussed further in next section.

4. Experimental Studies and Discussions

In this section, we evaluate the performance of the accessing the data from XML and JSON the discussion is based on five different queries. The systems are built using a personal computer equipped with 2.40GHz Intel(R) Core(TM) i7-5500U CPU, 8.00GB RAM and a 250GB solid-state drive. The operating system is Microsoft Windows 10. The database implementing the XML database (approach I) using X-Path for querying purposes and JSON database (approach II).

We use real dataset obtained from UniSZA. The variation in query time with the size of the database is also studied. For each of the two database approaches, the time taken to

extract the data from MySQL to XML and JSON format, and to make the queries with varying complexity specified above is measured with databases containing 1000, 5000, 10000, 15000 and 25000 records respectively. Table 4 shows the complexity of queries. For query retrieval, at each setting, the query is made for 10 times to calculate the average time and standard deviation (SD) [6].

The discussion is based on two experiments in the database development and their application for the storage of structured data, from the perspectives of test data, efficiency and scalability, and extensibility.

4.1. Test Data

The performance of the two database approaches is evaluated by using real dataset from UniSZA. The data contain 25,000 student records.

Table 4. Queries with different complexity

Query	Query Description
I	To retrieve results which is gender equal to "Male"
II	To retrieve results which is gender equal to "Female" and PNGS > "3.00".
III	To retrieve results which is PNGS between 2.5 and 3.5, and state equal to Terengganu.
IV	To retrieve result which is program equal to "Diploma in Information Technology" or "Diploma in Information Technology (Multimedia)", or program equal to "Diploma in Marketing" or "Diploma in Finance", and PN equal or more than "2.5".

4.2. Time for Data Extraction from Relational Database (XML vs JSON)

In this section, we evaluate the performance for extracting data from MySQL into XML and JSON format. Table 5 and Figure 10 show the execution time to extract complete XML and JSON. From the results, the performance of JSON approach is better compared to XML.

Table 5. Storage performance: XML vs. JSON

Approach	Database Implementation	Mean ± SD (ms)				
		1000 records	5000 records	10000 records	15000 records	25000 records
I	XML	84.386 ± 0.686	559.565 ± 21.732	1055.477 ± 24.975	1654.655 ± 18.485	3099.944 ± 47.374
II	JSON (Modified JSON)	29.614 ± 0.546	93.215 ± 0.961	215.666 ± 2.925	353.298 ± 20.325	568.207 ± 23.761

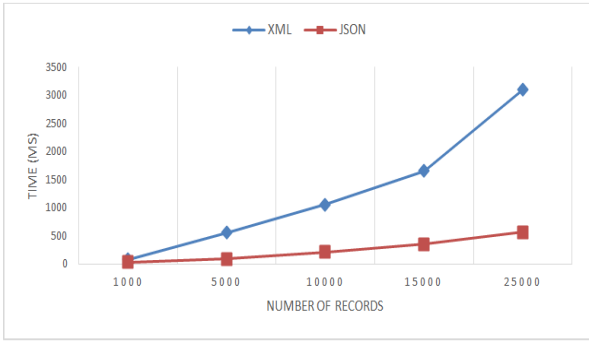

Figure 10. Time for extracting data into XML and JSON

4.3. Query retrieval performance (XML vs JSON)

In this section, we evaluated the performance of search the data from XML and JSON format. Four (4) different queries were executed and time for the query retrieval are executes in 10 times. Table 6 to Table 9 depict the query retrieval performance in term of time are taken to process the query in miliseconds (ms). The data are split into 5:- 1000 records, 5000 records,

10000 records, 15000 records and 25000 records. Mean and standard deviation are calculated based on standard algorithm [29].

Table 6. Query performance of the two approaches on database with different size: query I

Approach	Database Implementation	Mean ± SD (ms) – query I				
		1,000 records	5,000 records	10,000 records	15,000 records	25,000 records
I	XML	6.431 ± 0.107	31.112 ± 0.189	59.091 ± 0.250	84.612 ± 0.386	143.751 ± 0.400
II	JSON	3.791 ± 0.389	20.053 ± 0.553	46.223 ± 0.622	64.110 ± 0.175	109.028 ± 0.216

Table 7. Query performance of the two approaches on database with different size: query II

Approaches	Database Implementation	Mean ± SD (ms) – query II				
		1,000 records	5,000 records	10,000 records	15,000 records	25,000 records
I	XML	9.153 ± 0.941	32.895 ± 1.008	61.580 ± 3.310	89.131 ± 3.762	147.652 ± 1.880
II	JSON	4.993 ± 0.528	20.648 ± 1.336	49.417 ± 1.359	67.090 ± 1.722	116.716 ± 2.776

Table 8. Query performance of the three approaches on database with different size: query III

Approaches	Database Implementation	Mean ± SD (ms) – query III				
		1,000 records	5,000 records	10,000 records	15,000 records	25,000 records
I	XML	7.168 ± 0.120	32.572 ± 0.830	61.861 ± 0.431	88.783 ± 1.012	148.317 ± 1.386
II	JSON	5.348 ± 0.421	21.147 ± 0.655	49.639 ± 0.640	69.996 ± 3.020	119.689 ± 2.454

Table 9. Query performance of the three approaches on database with different size: query IV

Approaches	Database Implementation	Mean ± SD (ms) – query IV				
		1,000 records	5,000 records	10,000 records	15,000 records	25,000 records
I	XML	10.950 ± 0.280	45.993 ± 0.321	89.167 ± 2.142	131.366 ± 2.206	223.466 ± 0.123
II	JSON	4.990 ± 0.153	24.876 ± 0.277	56.307 ± 0.991	80.190 ± 0.699	133.172 ± 0.861

4.4. Efficiency and Scalability

This section evaluate the efficiency and scalability of the XML and JSON approach in storage and query retrieval.In time execution performance of data extraction, Figure 2 shows the time taken by XML and JSON which almost similar when extracting the small size of records (1000 records). However, the performance of XML is significantly faster than JSON execute large number of database records (5000 records to 25000 records). In JSON, time increases steadily based on number of records. Based on time performance for extracting the data from MySQL, the results shows JSON approach has good scalability compared to XML approach. Meanwhhile, in query retrieval, based on Figure 11 until Figure 14, JSON approach is proven to be better compared to XML. Query performance for JSON increases steadily based on number of records started with 1000, 5000, 10000, 15000 and 25000 records. However, query performance using XML approach is increase significantly when retrieving huge records. The results also proven that JSON approach is more scalability compared to XML.

4.5. Flexibility

In relational database approach, data modelling is restricted by the permission number of columns of the database management system. But, JSON is more flexible in that there is no need to pre-define the required number of columns. In JSON, data with complex structures can always be added subsequently. Further, re-design of schema is not required when the content is changed since the schema is generalized for any UniSZA data.

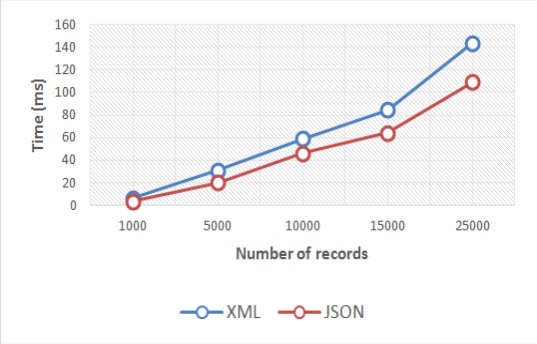

Figure 11. Variation in query time with the size of database: Query I

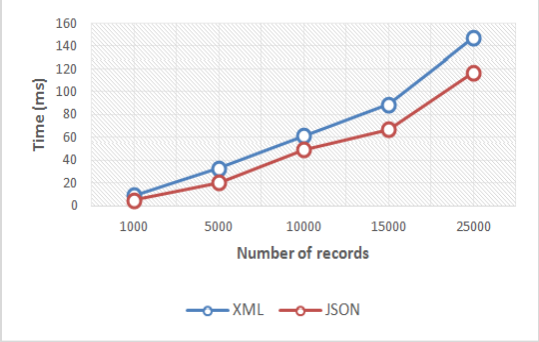

Figure 12. Variation in query time with the size of database: Query II

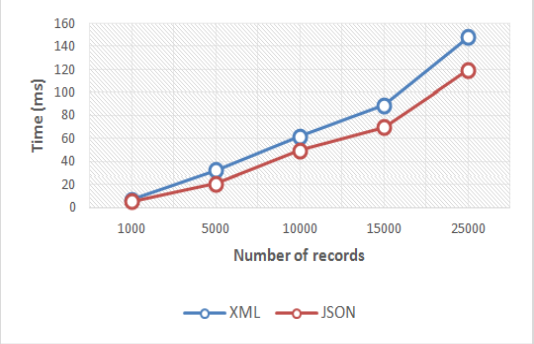

Figure 13. Variation in query time with the size of database: Query III

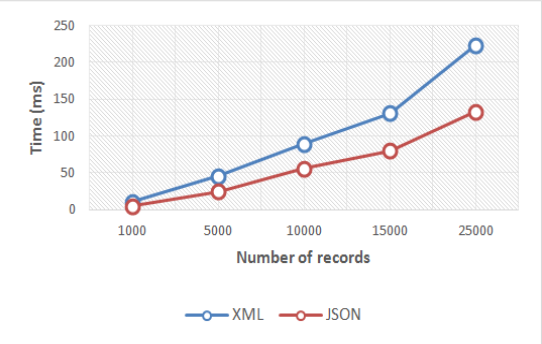

Figure 14. Variation in query time with the size of database: Query IV

4.6. Extensibility

JSON format is portable and independent platform. It is both human readable and machine process able. The format also facilities logical data management. Advantages JSON is the potential interoperability with other systems. For example other systems can easily retrieving the data from JSON format. By using JSON structure, other systems can easily integrate with this standard with minimal development effort.

5. Conclusion

A review on current approaches of database approach indicates that the JSON approach is viable alternative to relational database and XML as it provides better performance for storage and query retrieval while still retaining certain degree of scalability and flexibility. In this research, the performance of XML and JSON approach are compared by using real data provided by Information Technology Center, UniSZA. In this study, JSON is found to be flexible approach in handling huge records but it falls short in term of scalability and extensibility when compared to XML approaches.

In data extraction from relational database experiment, the execution time using JSON approach is lower compared to XML approach. In term of scalability, JSON approach shows the steady increment of time which is based on number of records. Meanwhile in XML approach, the execution of time changes rapidly when extracting large number of records. In this cases, JSON approach is more practical and significant to be used for extracting large or huge records.

In query retrieval experiment, four different type of complexity queries has been implemented. Based on the results, the execution time using JSON approach also lower compared to XML approach. However, in term of scalability, both approaches reflect the time is increases steadily based on number of records. Further optimization is required to fully exploit the potential of XML database and minimize the performance of the data search engine.

This study attempts to explore the vast opportunities JSON technologies in management of huge data. The prototype system developed is initially tested with maximum of 25,000 records only. Further evaluation using larger datasets, or even multiple databases and data warehouse, should give more comprehensive and thorough findings on the performance of data extraction and query retrieval of XML and JSON approaches.

References

[1] M Cai. Integrating Constraint and Relational Database Systems. 2004: 173-180.
[2] BD Blansit. The Basics of Relational Databases Using MySQL. *J. Electron. Resour. Med. Libr.* 2006; 3: 135-148.
[3] B Krüger. Why You Should Use A Relational Database Instead of A Spreadsheet. 2016; 9722.
[4] M Anthony. Understanding Relational Aggression. 2012: 15-24.
[5] RH Tajiri, EZ Marques, BB Zarpelão, L De Souza Mendes. A new approach for fuzzy classification in relational databases. *Lect. Notes Comput. Sci.* 2011; 6861: 511-518.
[6] KKY Lee, WC Tang, KS Choi. Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage. Comput. Methods Programs Biomed. 2013; 110(1): 99-109.
[7] A Grillenberger, R Romeike. Big Data - Challenges for Computer Science Education. Proc. ISSEP 2014. 2014: 29-40.
[8] A Kuckelberg, R Krieger. of XML Documents Using ORDBMS. 2003: 131-143.
[9] K Ahmad. A comparative analysis of managing XML data in relational database. Lect. Notes Comput. Sci. 2011; 6591(1): 100-108.
[10] H Su-Cheng, L Chien-Sing. Efficient Preprocesses for Fast Storage and Query Retrieval in Native XML Database. IETE Tech. Rev. 2009; 26(1): 28.
[11] J Van den Hoven. Database Management and XML: Interchange of data. Inf. Syst. Manag. 2002; 0530: 94-96.
[12] H Su-Cheng, L Chien-Sing, N Mustapha. Bridging XML and Relational Databases: Mapping Choices and Performance Evaluation. IETE Tech. Rev. 2010; 27(4): 308.
[13] JA Reyes. The Skinny on Big Data in Education. TechTrends. 2015; 59(2): 75-80.
[14] KA Jørgensen. Introduction to Databases. Introd. to Databases. 2011: 1-16.
[15] A Balmin, Y Papakonstantinou. Storing and Querying XML Data Using Denormalized Relational Databases. VLDB J. 2005; 14(1): 30-49.
[16] FZZM Ma. Representing and Reasoning About XML with Ontologies. 2014: 74-106.
[17] SR Collins, S Navathe, L Mark. XML schema mappings for heterogeneous database access. 2002; 44: 251-257.
[18] F Taghaboni-dutta, AJC Trappey, CV Trappey. An XML based supply chain integration hub for green product lifecycle management. Expert Syst. Appl. 2010; 37(11): 7319-7328.
[19] R Salem, O Boussa. Active XML-based Web data integration. 2013: 371-398.
[20] F Hernández, C Wert, I Recio, B Aguilera, W Koch, M Bogensperger, P Linde, G Günter, B Mulrenin, X Agenjo, R Yeats, L Bordoni, F Poggi. Xml for libraries, archives, and museums: The project covax. Appl. Artif. Intell. 2013; 17(8-9): 797-816.
[21] P De Meo, D Rosaci, GML Sarnè, D Ursino, G Terracina. Ec-Xamas: Supporting E-Commerce Activities By an Xml-Based Adaptive Multi-Agent System. Appl. Artif. Intell. 2007; 21(6): 529-562.
[22] F Pühretmair, W Wöß. XML-Based Integration of GIS and Heterogeneous Tourism Information. 2001: 346-358.
[23] WJ Shiang, MY Ho. An Interactive Tool Based on Xml Technology for Data Exchange Between Heterogeneous Erp Systems. J. Chinese Inst. Ind. Eng.. 2005; 22(4): 273-281.
[24] V Rajeswari, DK Varughese. A Novel Approach for Integrating Heterogeneous Database through XML. 2011; 2(2): 633-640.
[25] SM Bachrach. Integration of chemical data using XML. SAR QSAR Environ. Res. 2002; 13(3–4): 381-391.
[26] YS Chang, HD Park. XML Web Service - based development model for Internet GIS applications. 2015; 8816.
[27] N Nurseitov, M Paulson, R Reynolds, C Izurieta. Comparison of JSON and XML Data Interchange Formats : A Case Study.
[28] M Enoki. Event Processing over a Distributed JSON Store: Design and Performance. 2014: 395-404.
[29] A Khani, SD Boyles. An exact algorithm for the mean-standard deviation shortest path problem. Transp. Res. Part B Methodol. 2014; 81: 252-266.