Efficiency of Flat File Database Approach in Data Storage and Data Extraction for Big Data

Mohd Kamir Yusof¹, Mustafa Man²
¹ Universiti Sultan Zainal Abidin Kampus Tembila, 22200 Besut, Terengganu, Malaysia
² Universiti Malaysia Terengganu, 21300 Kuala Terengganu, Terengganu, Malaysia

ABSTRACT
Big data is the latest industry buzzword to describe large volume of structured and unstructured data that can be difficult to process and analyze. Most of organization looking for the best approach to manage and analyze the large volume of data especially in making a decision. XML and JSON are chosen by many organization because of powerful approach during retrieval and storage processes. However, these approaches, the execution time for retrieving large volume of data are still considerably inefficient due to several factors. In this contribution, three databases approaches namely Extensible Markup Language (XML), Java Object Notation (JSON) and Flat File database approach were investigated to evaluate their suitability for handling thousands records of publication data. The results showed flat file is the best choice for query retrieving speed and CPU usage. These are essential to cope with the characteristics of publication’s data. Whilst, XML, JSON and Flat File database approach technologies are relatively new to date in comparison to the relational database. Indeed, Text File Format technology demonstrates greater potential to become a key database technology for handling huge data due to increase of data annually.

Copyright © 2018 Institute of Advanced Engineering and Science. All rights reserved.

Corresponding Author:
Mohd Kamir Yusof,
Universiti Sultan Zainal Abidin
Kampus Tembila, 22200 Besut, Terengganu,
Malaysia.

1. INTRODUCTION
Big data or big data analytic have been used to describe the data sets and analytical techniques in applications that are so large and complex that they require advanced and unique data storage, management, analysis and visualization and technology [1]. Big data also refers to tools/application, processes, and procedures that allow organizations to create, manipulate and manage very large data sets and storage facilities. Tools in big data is required in order to handle the issues in big data such as analysis, capture, data duration, sharing, storage, transfer, visualization, querying, updating, and information privacy. Most of organizations or industries such as healthcare, academic publications, etc. are looking for the best approach or method in order to handle big data. For instance, in academic publications, according to sociology and research article, a number of reports have pointed to the growing use of big data across economic sectors and its potential to bolster productivity, efficiency and growth [2]. In this paper, issue about efficiency during access the publications data is considerable inefficiency. The efficiency of accessing publications data is relate to how data is stored. Big data or huge data must be stored using suitable approach. In traditional approach, data is stored using relational database. By using this approach, the data can be represented in a table form. Database Management System (DBMS) is used to control and manipulate the data [3][4]. However, by using this approach, time to fetch the data are considerably inefficiency. One of the solution to handle this problem is XML approach. XML is an emerging standard for exchanging representation over the Internet [5]. XML is widely used to store and manage huge of data. This approach is chosen because of
simple syntax, easy to generate and parse, easy to debug, extensibility, etc. [6]. This approach is successful and currently used by most of industries such as health care, education, business, etc. especially involves with huge data. A second database approach is JSON. JSON provide unique strength similar with XML approach. JSON is chosen because this approach directly support inside JavaScript and is best suited for JavaScript and provide significant performance compared to XML [7]. JSON is estimated to parse up to one hundred times faster than XML in modern browse. JSON format is proven more powerful compare to XML approach in term of time to fetch or retrieve data from database [7][13][14]. However, academician and researchers still looking for the best database approach specially involved with huge data.

This paper proposed text file as an alternative database approach compared to XML and JSON. Publication datasets is used for experimental purposes. The performance of Flat File approach will compared with XML and JSON approach. The comparisons are made from the following aspects: query performance and CPU usage for data retrieving process. The rest of contribution is organized as follows: Section 2 gives the related works. Section 3 describes about the kinds of data model such as relational database, XML, JSON and Text File. Section 4 discusses the three database approaches concerned based on experimental results and our experience in the development. Finally, a conclusion is given in Section 5.

2. RELATED WORKS
Based on past researches, the most popular approach compared to relational database is XML and JSON. XML stands for eXensible Mark-up Language a standard for data exchange issued by the World Wide Consortium (W3C) in 1998 [8]. XML has been widely accepted as a data format standard for data interchange and storage with the rapid development of internet and web service [9]. XML approaches have been implemented for clinical data storage [10]. This technique is effective to manage the clinical data and transform the data into structured format. The advantages of XML approach for clinical data are better in term of scalability, flexibility and extensibility. Native XML approach also has been implemented in external and distributed database [11]. The purpose of native XML is to minimize the query retrieval speed [12]. XML approach successful to handle huge data around 100000 records. In chemical industry, XML also used for integration of chemical data [13]. The implementation of XML approach because of chemistry community has been slower to adopt the Internet as a central service for exchanging information. Chemical data involves with large number of data file. XML approach can improve the efficiency of query processing when involves with the large number of data file. XML is implemented to overcome the information sharing each other and large number of databases issues. Through XML approach, different systems can share and exchange the information easily. By implementation of XML approach in different domains, XML is proven to handle large number of data. The efficiency of query processing using XML is efficiency compared to relational database. However, the efficiency of query processing using XML still can improve by using another approach as an alternative database approach. Meanwhile, JSON is lightweight data-interchange format is easy for humans to read and write, and for machines to parse and generate [12]. Nowadays, more and more data represented as JSON document. JSON is becoming the universal standard data format for the representation and exchanging the information. JSON approach is more powerful compared to XML approach. JSON approach has been implemented and able to handle 1000 records to 25000 records. The result shows JSON approach is powerful and more efficient in term of storage and query retrieval compared to XML [8][15]. However, the researchers still looking the best technique for handling huge data. In this paper, text file database approach is introduced as a new approach to handle and manage huge data. Text file is a computer file that only contains text and has no special formatting such as bold text, italic text, images, etc. Text file is simply, that way text files are commonly used for storage of information. In this paper, comparison will made between XML and JSON approach to handle huge data which is more than 50,000. This is important to shows the efficiency of JSON approach for handling huge data.

3. TYPES OF DATABASE MODEL
Four type of database approaches are represented in this section. Currently, most of data sources are store in traditional database approach which is called relational database. Because of limitation this approach, many researches looking an alternative database approach. Three alternative approaches are identified and performance among them are compared in order to show which one is better to use as an alternative for database model. They are XML, JSON and TXT. Figure 1 shows the diagram which is contains publication data. Based on Figure 1, publication data coming from different sources such as article, book, inproceeding, master thesis (msthesis), proceeding, website/URL (www) and PhD thesis (phdthesis).
These data sources are collect and extract to relational database approach. Figure 2 shows how publication data source in structured data format is extract and store in relational database. Number of records are stored in relation database is around 50,000 records. These records are split into four (4) segments: 1,000 records, 5,000 records, 10,000 records and 50,000 records. After records segmentation is done, these records from the relational database are convert into three different data format. They are XML, JSON and TXT format. XML, JSON and TXT data format can considered as an alternative approach for database approach.

Algorithm is designed in order to allow data from relation database approach converted into XML, JSON and flat file (text format). Section 3.1 demonstrate how data from the relational database is converted into XML, JSON and flat file (text format).

3.1. The Relational Database Approach
The definition about relational database is a data abstraction that presents the data in a database as a set of tables [16]. Relational data is complex, it mimics the way people think by grouping similar objects together and breaking down complex objects into similar ones. TABLE 1 until TABLE 7 shows how publication data is stored. Tables that contains the publication data is divided into two part; row and column. Column represent attributes name and rows represent number of data (something is called tuples).
Table 1: Article

| Id     | author         | title                                                                 | pages   | year | volume | Journal          | url                                      |
|--------|----------------|----------------------------------------------------------------------|---------|------|--------|------------------|------------------------------------------|
| 27422  | N. Prati       | A Partial Model of NP with E. Godel Theorems for Non-Constructive Logics. | 1245-1253 | 1994 | 59     | J. Symb. Log.    | db/journals/jsyml/jsyml59.html#Prati94    |
| 27424  | J. Barkley Rosser | Run-time performance optimization of an FPGA-based deduction engine for SAT solvers. | 547-562 | 2002 | 7      | ACM Trans. Design Autom. Electr. Syst. | db/journals/todaes/todaes7.html#DandalisP02 |
| 296027 | Andreas Dandalis, Viktor K. Prasanna |                                                     |         |      |        |                  |                                          |

Table 2: Book

| id | isbn   | author    | title                                                                 | Series                        | volume | publisher | year | url                                      |
|----|--------|-----------|----------------------------------------------------------------------|-------------------------------|--------|-----------|------|------------------------------------------|
| 214| 3-540-55382-7 | Andrew Cheese | Parallel Execution of Parlog                                           | Lecture Notes in Computer Science | 586    | Springer  | 1992 | -                                        |
| 219| 3-540-12282-6 | Heinz Bender | Korrekte Zugriffe zu verteilten Daten                                | Informatik-Fachberichte       | 63     | Springer  | 1983 | -                                        |
| 338398 | 3-540-280-7 | Rosalind Barden, Susan Stepney, Alf Smith, David Gries | Support for Using Z. Equational Logic: A Great Pedagogical Tool for Tea | Z User Workshop, ZUM | 255-280 | -         | -   | db/conf/zum/zum1992.html#BardenS92        |
| 338405 | 3-540-39 | Alf Smith | On Recursive Free Types in Z.                                         | Z User Workshop                | 3-39   | -         | -   | db/conf/zum/zum1991.html#Smith91         |
| 338419 | 3-540-508-509 | David Gries |                                                     | ZUM                           | 508-509 | -         | -   | -                                        |

Table 3: Inproceeding

| id   | author             | title                                                                 | Pages  | year | booktitle  | url                                      |
|------|--------------------|----------------------------------------------------------------------|--------|------|------------|------------------------------------------|
| 338398 | Rosalind Barden, Susan Stepney, Alf Smith, David Gries | Support for Using Z. Equational Logic: A Great Pedagogical Tool for Tea | 255-280 | 1992 | Z User Workshop, ZUM | db/conf/zum/zum1992.html#BardenS92        |
| 338405 | Alf Smith | On Recursive Free Types in Z.                                         | 3-39   | 1991 | Z User Workshop | db/conf/zum/zum1991.html#Smith91         |
| 338419 | David Gries |                                                     | 508-509 | 1995 | ZUM | 508-509 | -                                        |

Table 4: Msthesis

| Id  | author            | Title                                                                 | year | School                                           |
|-----|-------------------|----------------------------------------------------------------------|------|-------------------------------------------------|
| 12  | Tolga Yurek       | Efficient View Maintenance at Data Warehouses.                       | 1997 | University of California at Santa Barbara, Departm |
| 14  | Peter Van Roy     | A Prolog Compiler for the PLM.                                       | 1984 | University of California at Berkeley            |
| 15  | Tatu Ylnen        | Shadow Paging Is Feasible.                                           | 1994 | Helsinki University of Technology, Department of C |

Table 5: Phdthesis

| Id   | editor           | Title                                                                 | Year | Month | School                  |
|------|------------------|----------------------------------------------------------------------|------|-------|------------------------|
| 1    | Joann J. Ordille | Descriptive Name Services for Large Internets.                       | 1993 |       | Univ. of Wisconsin-Madison |
| 2    | Francisco Reverbell | Persistence in Distributed Object Systems: ORB/ODB... | 1996 | April | University of New Mexico |
| 4    | Dietmar Seipel   | Decomposition in Database and Knowledge-Based Systems.               | 1989 |       | Uni Wurzburg            |

Table 6: Proceeding

Efficiency of Flat File Database Approach in Data Storage and Data Extraction ... (Mohd Kamir Yusof)
3.2. XML Approach
XML provides a standard for the semantic management of data. It is a formal meta-language facility for defining a markup language. The basic unit in an XML file is entity or chunk that contains content and mark up. Many excellent model-mapping schemas have been proposed for storing and retrieving XML data into/from relational database [17]. The markup describes a content. More generally, markup consists of tags, attributes, comments, and processing instructions for the content. In a start tag, the name and any additional information are surrounded by the “<” and “>” characters. Figure 3 shows the algorithm how data from relational database is convert into XML format.

![Figure 3. Algorithm (Relational Database to XML Format)](http://www.w3.org/TR/query-datamodel)

### Table 7: www

| id | editor | title | Booktitle | year | url |
|----|--------|-------|-----------|------|-----|
| 2  | Mary F. Fernandez, Jonathan Robie | XML Query Data Model | - | 2001 | http://www.w3.org/TR/query-datamodel |
| 3  | Arnon Rosenthal | The Future of Classic Data Administration: Objects | SWEE | 1998 | http://www.mitre.org/support/swee/rosenthal.html |

**Input**: Tables names, records  
**Output**: Data set (A)  
**Steps**:

1. Read number of tables  
   1.1 Create XML tag by representing table title, x  
   1.2 Read records/tuples, y  
   1.3 Create indents/attributes tag by representing attribute name  
   1.4 Assign each record to each attribute tag, i  
   1.5 Close XML tag of x  
   1.6 Repeat step 1.1 until end of records  
2. Repeat step 1 until end of tables  
3. Assign A = 
   <x>  
   <y1>1</y1>  
   <y2>1</y2>  
   ...  
   <yn>1</yn>  
   </x>  
4. Display data set A
After algorithm in Figure 3 is convert in programming code, then execution is occur, system will produce XML file as represented in Figure 4. Similarly, an end tag consists of the tag name surrounded by the “< /” and “>”. XML is case sensitive so start and end tag names must match exactly. Figure 3 shows how the publication data is represented in XML format.

```xml
<record>
  <article>
    <id>274222</id>
    <author>N. Pratt</author>
    <title>A Partial Model of NP with E.</title>
    <pages>1245-1253</pages>
    <year>1994</year>
    <volume>59</volume>
    <journal>J. Symb. Log.</journal>
    <url>db/journals/jsyml/jsyml59.html#Prati94</url>
  </article>

  <book>
    <id>211</id>
    <isbn>3-540-60058-2</isbn>
    <author>Marco Cadoli</author>
    <title>Tractable Reasoning in Artificial Intelligence</title>
    <series>Lecture Notes in Computer Science</series>
    <year>1995</year>
    <url>...</url>
  </book>

  <inproceeding>
    <id>338396</id>
    <author>Regine Laleau, Amel Mammar</author>
    <title>A Generic Process to Refine a B Specification into</title>
    <pages>22-41</pages>
    <year>2000</year>
    <booktitle>ZZ</booktitle>
    <url>db/conf/zzm/zzm2000.html#LaleauM00</url>
  </inproceeding>

  <msthesis>
    <id>11</id>
    <author>Kurt P. Brown</author>
    <title>PRPL: A Database Workload Specification Language, v1.3.</title>
    <year>1992</year>
    <school>Univ. of Wisconsin-Madison</school>
  </msthesis>

  <phdthesis>
    <id>1</id>
    <editor>Joann J. Ordille</editor>
    <title>Descriptive Name Services for Large Internets.</title>
    <year>1993</year>
    <month>...</month>
    <school>Univ. of Wisconsin-Madison</school>
  </phdthesis>

  <proceeding>
    <id>1325</id>
    <editor>Elen Balka, Richard Smith</editor>
    <title>Woman, Work and Computerization: Charting a Course to the Future, IFIP TC9/WG9.1 Seventh International Conference on Woman, Work and Computerization, June 8-11, 2000, Vancouver, British Columbia, Canada</title>
  </proceeding>
</record>
```
3.3. **JSON Approach**

In this approach, data is represented in array format. JSON is built on two structures. The first is a collection of name/value of pairs. In various language, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associate array. The second is an ordered list of values. In most language, this is realized as an array, list or sequence. Each object begins with “{“ and ends with “}”. Array is an ordered collection of values. An array begin with “[“ and ends with “]”. Meanwhile, a value can be a string in double quotes, or a number, or true or false, or an object or an array. Figure 5 represents algorithm how to convert relational database to JSON approach. In this algorithm, two input are required which is table name and data/tuple in each table. Variable of x is assign to table name and variable y represented to tuple for each table. Data set B are represented as a JSON file after execute this algorithm.

| Input | Tables names, records |
|-------|----------------------|
| Output | Data set B |
| Steps |
| 1. | Read number of tables, x |
| 1.1 | Assign table name to variable x |
| 1.2 | Read data record/tuple |
| 1.3 | Assign data record/tuple to y |
| 1.4 | Combine variable of x and y as below: |
| &nbsp;&nbsp;&nbsp;{x: [y_i, y_{i+1}, y_n+1]} |
| 1.5 | Repeat step 1.2 to step 1.4 until end of records |
| 2. | Repeat step 1 until end of tables |
| 3. | Assign B = |
| &nbsp;&nbsp;&nbsp;{x: [y_i, y_{i+1}, ..., y_{n+1}]} |
| 4. | Display data set (B) |

**Figure 5. Algorithm (Relational Database to JSON Format)**

JSON file is produced as a Figure 6 after execute algorithm in Figure 4. JSON file is simple which each data/record separate by line.
Flat File Approach

In this approach, data is represented in flat file (text format). Flat file are text files stored in computer science. Data in flat file is simple and can ported to any program. The basic characteristics of a flat file are that data are stored as plain text, even the number are plain text, and that each line of the file contains one record or case in the data set. Each line a flat file, several contain the values for the different variables in the data set. Fields within a record are separated by a special character, or delimiter. Each line after the header consists of two fields separated by a colon (the character ":" is the delimiter). Alternatively, we can used “white space” (one or more space tabs) as the delimier. Figure 7 show the algorithm how data from relational database is converted into flat file (text format).

```
Input : Tables names, records
Output : Data set C
Steps
1. Read number of tables, x
2. Assign table name to variable x
   2.1 Read records/tuples in the table
   2.2 Assign record/tuple to y
   2.3 Assign x and y to z
      Z = {x, y, y_{i+1}, y_{n+1}}
   2.4 Repeat step 2.1 to step 2.3 until end of record in x
3. Assign variable m
   M = {x, y, y_{i+1}, y_{n+1}, ..., x, y_{i+1}, y_{n+1}, ..., x, y_{i+1}, y_{n+1}}
4. Repeat step 1 until 3 until end of tables
```

Figure 6. Publication Data in JSON Format

3.4. Flat File Approach

In this approach, data is represented in flat file (text format). Flat file are text files stored in computer science. Data in flat file is simple and can ported to any program. The basic characteristics of a flat file are that data are stored as plain text, even the number are plain text, and that each line of the file contains one record or case in the data set. Each line a flat file, several contain the values for the different variables in the data set. Fields within a record are separated by a special character, or delimiter. Each line after the header consists of two fields separated by a colon (the character “:” is the delimiter). Alternatively, we can used “white space” (one or more space tabs) as the delimiter. Figure 7 show the algorithm how data from relational database is converted into flat file (text format).
4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the accessing the data from XML and JSON. Four different queries are used in experiments. The systems are build using a personal computer equipped with 2.40GHz Intel® Core ™ i7-5500U CPU, 8.00 GB RAM and a 250 GB 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 benchmark dataset DBLP [18]. The variation in query time with the size of the database is also studied. For each of two database approaches, the time to query and CPU usage with varying complexity specified above is measured with databases containing 1000, 5000, 10,000 and 50,000 records respectively. For query retrieval, at each setting, the query is made for 10 times to calculate the average time and standard deviation [10].

The discussion is based on two experiments in the databases 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 two database approaches is evaluated by using benchmark dataset DBLP. The data contain 50,000 records. TABLE 8 shows the queries with different complexity and TABLE 9 shows the queries constructed in the SQL statement.

### Table 8: Queries with Different Complexity [6]

| Query | Query description |
|-------|-------------------|
| I     | List out all the URLs which begin with the “db/journals” path |
| II    | List out all the titles of the master thesis which contains the “Data” keyword |
| III   | List the titles of inproceeding where the author is “Regine Laleau, Mammar” |
| IV    | Count the number of phd thesis published in each year |

### Table 9. Queries Constructed in SQL Commands

| Query | Query description |
|-------|-------------------|
| I     | Select * from url where text like ‘%db/journals%’ |
| II    | Select * from title where text like ‘%Data%’ |
| III   | Select title from inproceeding where author=’Regine Laleau, Mammar’ |
| IV    | Select count(id), year from phdthesis group by year |

### 4.2 Data Extraction (XML, JSON and Flat File (text format))

In this section, data from relational database are extract and convert into three different data format. The data size for each format are represented in KB. TABLE 10 until TABLE 13 show the data size and performance query retrieval in three different format which are XML, JSON and Flat File (text format). Data are split into 4: 1000 records, 5000 records, 20,000 records and 50,000 records. Then, these records are convert into different data format. Based on TABLE 10 until TABLE 13, Flat File (text format) format is smaller compared to XML and JSON. That way, time to data retrieval also shows flat file in text format faster compared to XML and JSON.

### Table 10. Query performance of the three approaches on database with different size: Query I

| Approach | Database Implementation | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) |
|----------|-------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| I        | XML                     | 339       | 14.41     | 1712      | 75.30     | 3405      | 136.66    | 16300     | 685.67    |
|          |                         | ± 0.28    | 0.42      | ± 1.44    | 11844     | 397.64    |
| II       | JSON                    | 249       | 10.41     | 1266      | 40.35     | 2515      | 80.89     | 11844     | 397.64    |
|          |                         | ± 0.35    | 0.21      | ± 0.43    | ± 2.23    |
| III      | TXT                     | 185       | 8.18      | 948       | 38.56     | 1890      | 76.35     | 8868      | 370.48    |
|          |                         | ± 0.14    | 0.77      | ± 0.25    | ± 0.26    |

### Table 11. Query performance of the three approaches on database with different size: Query II

| Approach | Database Implementation | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) |
|----------|-------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| I        | XML                     | 339       | 11.36     | 1712      | 40.56     | 3405      | 73.51     | 16300     | 333.28    |
|          |                         | ± 0.31    | 0.25      | ± 0.32    | ± 3.11    |
| II       | JSON                    | 249       | 8.41      | 1266      | 35.36     | 2515      | 66.48     | 11844     | 309.75    |
|          |                         | ± 0.32    | 0.25      | ± 0.233   | ± 1.26    |
| III      | TXT                     | 185       | 8.24      | 948       | 23.54     | 1890      | 45.57     | 8868      | 223.94    |
|          |                         | ± 0.07    | 0.28      | ± 0.20    | ± 0.64    |

### Table 12. Query performance of the three approaches on database with different size: Query III

| Approach | Database Implementation | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) | Size (KB) | Time (ms) |
|----------|-------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| I        | XML                     | 339       | 8.56      | 1712      | 41.53     | 3405      | 85.52     | 16300     | 383.48    |
|          |                         | ± 0.28    | 0.26      | ± 0.26    | 1.07      | ± 1.84    |
| II       | JSON                    | 249       | 8.14      | 1266      | 35.14     | 2515      | 65.42     | 11844     | 362.82    |
|          |                         | ± 0.10    | 0.53      | ± 0.99    | ± 0.89    |
| III      | TXT                     | 185       | 7.92      | 948       | 25.51     | 1890      | 50.73     | 8868      | 268.42    |
4.3. Data Retrieval (XML, JSON and TXT)

In this section, we evaluated the performance of searching the data from XML, JSON and Flat File (text format). Four (4) different queries were executed and time for query retrieval are executed in 10 times. Figure 6 until Figure 9 depict the query retrieval performance in terms of time taken to process the query in milliseconds (ms). The data are split into 5 groups: 1000 records, 5000 records, 10,000 records and 50,000 records. Mean and standard deviation are calculated based on standard algorithm.

Table 13. Query performance of the three approaches on database with different size: Query IV

| Approach | Database Implementation | Size (KB) | Time (ms) | Mean ± SD (ms) - Query IV |
|----------|------------------------|----------|-----------|--------------------------|
| I        | XML                    | 339      | 10.50 ± 0.37 | 112 ± 0.35 | 3403 ± 0.54 | 16300 ± 1.72 |
| II       | JSON                   | 249      | 8.57 ± 0.28  | 1266 ± 0.77 | 2515 ± 0.73 | 11844 ± 0.75 |
| III      | TXT                    | 185      | 7.84 ± 0.16  | 948 ± 0.28  | 1890 ± 0.34 | 8868 ± 0.39  |

Figure 6: Time Retrieval – Query I

Figure 7: Time Retrieval – Query II

Figure 8: Time Retrieval – Query III

Figure 9: Time Retrieval – Query IV
4.4. CPU Usage Performance (XML vs. JSON)

The performance of two database approaches is evaluated by using benchmark dataset DBLP. The
result shows flat file (text format) data format used less CPU usage compared to XML and JSON. The result
show more significant when involves huge data especially 50,000 records.

![Figures 10-13 showing CPU usage for different queries.]

4.5. Efficiency and Scalability

The performance of two database approaches is evaluated by using benchmark dataset DBLP. The
data contain 50,000 records. TABLE 8 shows the queries with different complexity. Meanwhile, TABLE 9
shows the SQL commands based on query complexity in Table 8. In term of execution time for query
retrieval, Figure 5 until Figure 8 shows, flat file approach is powerful compared to XML and JSON approach.
The scalability of time execution is changing based on number of records. Meanwhile, in term of CPU usage,
flat file approach is also better compared to XML and JSON. Figure 9 until Figure 12 shows the CPU usage
are used by flat file approach is low compared to XML and JSON. In this case, percentage of CPU are used
to execute the query based on number of records still considerable lower compared to XML and JSON
approach. The scalability of CPU usage using flat file approach is moving steadily based on number of
records.

4.6. Flexibility

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

4.7. Extensibility
Flat file approach is portable and independent platform. It is both human readable and machine process able. The format also facilities logical data management. Advantages of flat file is the potential interoperability with other systems. For example other systems can easily retrieving the data from flat file (text format). By using flat file database approach, 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 flat file approach is viable alternative to relational database, XML and JSON as it provides better performance for query retrieval and CPU usage while still retaining certain degree of scalability and flexibility. In this research, the performance of Flat File, XML and JSON approach are compared by using DBLP dataset. In this study, Flat file is found to be flexible approach in handling huge records but it falls short in term of scalability and extensibility when compared to XML and JSON approaches.

In query retrieval experiment, four different type of complexity queries has been implemented. Based on the results, the execution of time using flat file approach also lower compared to XML and JSON 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 and JSON database and minimize the performance of the data search engine.

In CPU usage experiment, the percentage of CPU usage using flat file approach is lower compared to XML and JSON approach. In term of scalability, flat file approach shows the steady increment the percentage of CPU usage based on number of records. Meanwhile in XML and JSON approach, the percentage of CPU usage changes rapidly when execute large number of records. In this cases, flat file approach is more practical and significant to be used for extracting large or huge records.

The study attempts to explore the vast opportunities flat file technologies in management of huge data. The prototype system developed is initially tested with maximum of 50,000 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 query retrieval and CPU usage of Flat File, XML and JSON approaches.

References
[1] Hsinchun Chen, Roger H.L. Chiang, Veda C.Storey. 2012. “Business Intelligence and Analytics: From Big Data to Big Impact”. Special Issue: Business Intelligence Research. Vol 36, No. 4, pp: 1165 – 1188.
[2] Ralph Schroeder. “Big data business models: Challenges and opportunities”, *Sociology: Research Article*, 2016, pp: 1-15.
[3] B. Douglas Blansit MPS, MLIS, “The Basics of Relational Database Using MySQL”, *Journal of Electronic Resources in Medical Libraries*, 2006, pp. 135-148.
[4] VV. Agarwal. 2012. Understanding Relational Databases. *Beginning C# 5.0 Databases*.
[5] Haw Su Cheng, Lee Chien Sing, and Norwati Mustapha, “Bridging XML and Relational Databases: Mapping Choices and Performance Evaluation”, *IETE Technical Review*, Jul-Aug 2010, Issue 4, Vol 4, pp: 308-317.
[6] V. Rajeswari, Dr. Dharnishanth, K. Varughese. 2011. A Novel Approach for Integrating Heterogeneous Database through XML. *International Journal of Computer Science and Information Technologies*. Vol. 2 (2), 2011 633 – 640.
[7] Nurzhan Nurseitov, Micheal Paulson, Randall Reynolds, Clemente Izurieta. Comparison of JSON and XML Data Interchange Formats: Case Study.
[8] M. Meneghello. XML (extensible markup language) – The New Language of Data Exchange. “Cartography”, Vol. 30, No. 1, June 2001, pp: 51 – 57.
[9] Kanagaraj.S, Sunita Abburu. Converting Relational Database Into XML Document. *International Journal of Computer Science*, Vol 9, Issue 2, No. 1, March 2012.
[10] Ken Ka-Yin Lee, Wai-Choi Tang, Kup-Sze Choi, “Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage”, *Computer Methods and Programs in Biomedicine*, 2013, Vol 110, pp: 99-109.
[11] Andrew Clarke, Eric Pardee, Robert Steele, “External and Distributed Databases: Efficient and Secure XML Query Assurance”, *International Journal of Computational Intelligence System*, 2012, pp: 421 – 433.
[12] Haw Su-Cheng and Lee Chien Sing, “Efficient Preprocesses for Fast Storage and Query Retrieval in Native XML Database”, *IETE Technical Review*, Sep 2014, Vol. 26, Issue 1, pp: 28 – 40.
[13] S.M. Bachrach, “Integration of Chemical Data using XML”, *SAR and QSAR in Environmental Research*, 2002, Vol. 13(3-4), pp: 381- 390.
[14] Ankit Bhartan and Devesh Bharathan, “Relational JSON, An Enriched Method to Store and Query JSON Records”, *International Journal of Computer Application*, July 2014, Vol. 98, No. 7, pp: 1 – 4.
[15] Mohd Kamir Yusof, Mustafà Man, “Efficiency of JSON Approach for Data Extraction and Query Retrieval”, *Indonesian Journal of Electrical Engineering and Computer Science*, Oct 2016, Vol. 4, No. 1, pp: 2013 – 214.

[16] P. Revesz. 2010. Introduction to Database: From Biological to Spatio -Temporal. Springer – Verlag London Limited 2010.

[17] Jun Wu, Shang Yi-Shang. (2009). “An Efficient Mapping Schema for Storing and Accessing XML Data in Relational Databases”. *International Journal of Web Information System*. Vol. 5, No. 3, pp. 327 – 347.

[18] DBLP, Available from [https://kdl.cs.umass.edu/display/public/DBLP](https://kdl.cs.umass.edu/display/public/DBLP)