Presented at the Research Congress 2013
De La Salle University Manila
March 7-9, 2013

COMPARATIVE ANALYSIS OF RELATIONAL AND NON-RELATIONAL DATABASE MODELS FOR SIMPLE QUERIES IN A WEB-BASED APPLICATION

Remedios de Dios Bulos, Jay-Vee Bonsol, Rhon Diaz, Adela A. Lazaro and Veronica E. D. Serra
De La Salle University

Abstract: This research investigates, provides empirical evidence and conducts a comparative study among relational and non-relational databases involving simple queries in a web-based application. Our testbed is a simple recipe recommendation system (MyRef) which (ideally) uses left-over foods in the refrigerator. For comparative analysis, the system is implemented using relational, XML and JSON database models. In the relational database implementation, records or recipes are searched, selected and fetched through SQL queries using MySQL. In the XML and JSON implementation, PHP programs are used to search, select and fetch XML/JSON recipe documents. Two types of file storage strategies are used; the first strategy involves storing each recipe in separate XML/JSON files; and the other strategy uses only one XML/JSON file to store all the recipes. Experiments involving 10 sets of test query cases are conducted. The database for each implementation consists of 1568 recipes. In recording observations for each test case, two timestamps are used. A timestamp is recorded every time a query is submitted and every time the result of the query is returned. The absolute difference between the two timestamps is used to measure the amount of time (in seconds) it takes to execute each test case. Based on the results involving all test 10 cases, SQL implementation retrieved the result set of recipes the fastest, followed by JSON and then XML.

Key Words: relational databases; non-relational database; SQL; XML; JSON

1 INTRODUCTION

Relational databases have dominated the realm of database applications ever since they were introduced through the pioneering research of Edgar F. Codd a little less than forty years ago. However, of late non-relational databases (e.g. XML and JSON) have been gaining ground in usage particularly among internet-based companies such as Facebook, Twitter, Amazon, and Google; and the NoSQL technology has been adopted especially in applications that require data exchange over the internet. The relational model is rigidly structured and it has been around for almost 4 decades, and the systems that implement it have been around for almost as long. XML and JSON are relatively new models and they follow a semi-structured format. The main objective of this research is to undertake a comparative study among the three models. Our approach to the study is two pronged: (1) evaluate the state of the art of the

three models by investigating relevant and up-to-date literature and (2) conduct an empirical study by implementing the three models in a web-based application system (MyRef) and then evaluate their query processing performance.

2 STATE OF THE ART: RELATIONAL MODEL vs XML vs JSON

In our survey of literature, we identified some of the main features that differentiate the three data models – relational model, XML and JSON. In (Widom, 2013) these are enumerated as: schema, queries, ordering and implementation. Table 1 summarizes the comparison between the three models.

Table 1. Relational Model vs XML vs JSON [Widom, 2013]

	Relational	XML	JSON
Structure	tables	hierarchical tree, graph	sets, nested arrays
Schema	fixed in advance	flexible "self-describing"	flexible "self-describing"
Queries	simple, easy, expressive languages (e.g. SQL)	less simple, not widely used (e.g. XPATH, XQUERY, XSLT)	less simple, not widely used (proposals like JSON Path, JSON Query, JAQL)
Ordering	unordered	implied	arrays
Implementation	native	add-on	coupled with programming languages, NoSQL systems

In terms of data structure, the relational model is basically a set of tables, with rows and columns. XML has a hierarchical structure; it uses document or string format and it often follows a tree structure (or sometimes a graph structure). JSON is based on sets of label pairs and nested arrays. (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

With respect to schema, the relational model is quite rigid; it requires that a schema be designed in advance before data is loaded; the data must conform to the schema. On the other hand, XML has a self-describing schema; this allows more flexibility because the schema and data can be combined together. Furthermore, XML allows elements of a schema (e.g. enumeration of attributes) to be optional. However, in the relational model, all attributes are uniform for all elements; and in case of missing values, null values are used instead. Because XML has a flexible schema, the creation, addition, deletion and introduction of inconsistencies in the structure do not pose any problem. JSON, just like in XML has a flexible self-describing schema; however, it is easier to undersrtand than its XML counterpart. (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

When it comes to querying the database, the relational model uses SQL, which is a simple, easy, fairly efficient and an expressive high-level language. On the other hand, querying in XML is still evolving and some of the query languages currently being used include XPath, XQuery, and XSLT. For JSON, query languages are still being proposed at

this stage (e.g. JSON Path, JSON Query, and JAQL). Presently, JSON data is manipulated programmatically. (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

With respect to ordering, the relational model is unordered while XML and JSON are ordered. In the relational model, ordering of query results requires the inclusion of the ORDER BY clause in SQL statement. In XML there is an implied ordering because the document structure is hierarchical. JSON makes use of ordered arrays. XML and JSON data are usually written in files, which are naturally ordered. (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

Finally, in terms of implementation, relational database management systems which have been around for almost 40 years implement the relational model natively; in addition, they're widely used, very mature, efficient and powerful. Compared to the relational model, XML hasn't been around for as long. In conventional database systems, XML is typically utilized as an add-on feature. By and large, in most application systems, XML will be a layer over the relational database system. This allows data entry and querying in XML as well as combining XML and relational data in a single system. These are converted to relational implementation but they are not the native model of the system itself. For JSON, there are still no stand alone database systems that employ JSON as their data model; JSON is usually coupled with programming languages. However, JSON is used in NoSQL systems: (1) as a format for reading data into and writing data out from the systems; and (2) as "Document Management Systems" where the documents themselves may contain JSON data and then the systems will have special features for manipulating the JSON document (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

Both XML and JSON are very good for writing semi-structured data into a file format and are both popularly used for data interchange. In general, XML is more verbose. In terms of complexity, a JSON document is quicker and easier to understand. With respect to validity, that is, the ability to specify constraints or restriction or schema on the structure of data and have it enforced by tools or by a system, XML has the notion of document type descriptors (DTDs) and XML Schema Descriptors (XSDs). These are specified, checked and at present are fairly widely used. On the other hand, JSON has JSON Schema which can also be specified and then checked for conformity, albeit presently not widely used. With respect to the programming interface, JSON is better than XML. In XML, there is an impedance mismatch, that is, the XML model does not typically match the model of data inside a programming language. Some manipulation at the interface between programming languages and the database system still needs to be carried out. On the other hand, in JSON there is a more direct mapping between many programming languages and its structures. (Elmasri and Navathe, 2010; Garcia-Molina et al., 2007; Ramakrishna and Gehrke, 2007; Silberschatz et al., 2010; Ullman and Widom, 2007; Widom, 2013)

3 MyRef: A WEB-BASED FOOD RECOMMENDER SYTEM

For our testbed, we developed and implemented the MyRef, a web-based application system, which is a simple recipe recommendation system that matches the user's chosen ingredients. MyRef allows the user to select a set of currently available ingredients (ideally in the refrigerator), and then should enable the user to prepare a meal by following the recipes that are suggested by the system. It is implemented using relational, XML and JSON database models. Our comparative study aims to determine which among the three models provides the most efficient method of generating simple queries based on the performance of running simple queries from the recipe databases of the three models. In carrying out our specific objectives, MyRef was developed in five versions using five different database implementations: (1) relational database (2) XML with only one document for all recipes (3) XML with separate documents for each recipe (4) JSON with only one document for all recipes and (5) JSON with separate documents for each recipe. In implementing the different versions, PHP was used for the main program. In addition, HTML, Javascript and MySQL were also used. In populating the recipe databases, a website crawler was dispatched and it extracted data from various recipe websites. A total of 1568 recipes were collected. Using specially designed programs, the extracted data was then cleaned, parsed and loaded in the relational databases as well as created the respective JSON and XML recipe document files.

3.1 Database Structure

In relational model implementation of MyRef, the database schema consisted only of one table (tblrecipes). The table structure as well as sample data is shown in Table 2. It is implemented in MySQL, and uses SQL statements for queries. SQL is a high-level language that allows writing codes in a fairly compact fashion and since it is declarative, it does not need writing the algorithms to generate the queries. (Widom, 2013)

Table 2. Table tblrecipes (relational model)

Column	Data Type	Sample Data
recipeID	INT	1000
recipeName	TEXT	Poppadums with lime & coriander dip
recipeDescription	TEXT	Start your Indian meal in style with Meena Pathak's tangy dip - serve with plain poppadums
recipeRating	VARCHAR	6 ratings 5
recipeDifficulty	VARCHAR	[id:201207290256380009.27][link:http://www.bbcgoodfood.com/images/recipes/easy_btn.gif][name:easy_btn.gif]
recipeServing	VARCHAR	8 poppadums
recipePrepTIme	VARCHAR	
recipeCookTime	VARCHAR	Ready in 35-45 minutes, plus overnight marinating
recipeMethod	TEXT	Put the mango chutney in a bowl and chop up any large pieces of mango. Mix with the spring onions, coriander and lime juice. Cover and chill until ready to serve. (You can make this up to a day ahead and keep it in a covered container in the fridge.) Serve the chilled dip in a bowl on a plate, with the poppadums broken in pieces and placed around the sides.
recipeNutrients	TEXT	227 kcalories, protein 8g, carbohydrate 27g, fat 10 g, saturated fat 2g, fibre 3g, sugar 5g, salt 2.19 g
recipeIngredients	TEXT	4 tbsp mango chutney 1 bunch spring onions 4 tbsp fresh coriander 5 tbsp lime juice (about 3 limes) 8 ready-to-eat poppadoms
recipeImage	VARCHAR	[id:201207290256380009.28][link:http://www.bbcgoodfood.com/recipes/1691/images/1691_MEDIUM.jpg][name:1691_MEDIUM.jpg]

In the XML implementation of MyRef, the recipe document file (see Figure 2) follows a semi-structured format. The document format is similar to HTML. It has three basic components: tagged elements (which can be nested), attributes and text. The tagged element has an opening tag (e.g. <recipeID>), text or other sub-elements and a closing tag (e.g. </recipeID>). An attribute consists of an attribute name, the equal sign and then an attribute value. The text is a string within the elements (e.g. 1000 for recipeID). The XML document follows a tree structure where the strings or text form the leaves. An XML document is considered well formed if it adheres to the basic structural requirements of XML, which are single root element, matching tags with proper nesting and unique attribute names within elements. (Widom, 2013)

```
<recipe>
        <recipeID><![CDATA[1000]]></recipeID>
        <recipeName><![CDATA[Poppadums with lime &amp; coriander dip]]></recipeName>
        <recipeDescription><![CDATA[Start your Indian meal in style with Meena Pathak's tangy dip - serve
        with plain poppadums]]></recipeDescription>
        <recipeRating><![CDATA[6 ratings 5]]></recipeRating>
        <recipeDifficulty><![CDATA[[id:201207290256380009.27][link:http://www.bbcgoodfood.com/images/recipes/ea
        sy_btn.gif][name:easy_btn.gif]]]></recipeDifficulty>
        <recipeServing><![CDATA[8 poppadums]]></recipeServing>
        <recipePrepTIme><![CDATA[Ready in 5 minutes]]></recipePrepTIme>
        <recipeCookTime><![CDATA[]]></recipeCookTime>
        <recipeMethod><![CDATA[Put the mango chutney in a bowl and chop up any large pieces of mango. Mix with
        the spring onions, coriander and lime juice. Cover and chill until ready to serve. (You can make this
        up to a day ahead and keep it in a covered container in the fridge.) Serve the chilled dip in a bowl on
        a plate, with the poppadums broken in pieces and placed around the sides.]]></recipeMethod>
        <recipeNutrients><![CDATA[227 kcalories, protein 8g, carbohydrate 27g, fat 10 g, saturated fat 2g,
        fibre 3g, sugar 5g, salt 2.19 g]]></recipeNutrients>
        <recipeIngredients><![CDATA[4 tbsp mango chutney 1 bunch spring onions 4 tbsp fresh coriander 5 tbsp
        lime juice (about 3 limes) 8 ready-to-eat poppadoms]]></recipeIngredients>
        <recipeImage><![CDATA[[id:201207290256380009.28][link:http://www.bbcgoodfood.com/recipes/1691/images/16
        91_MEDIUM.jpg][name:1691_MEDIUM.jpg]]]></recipeImage>
</recipe>
```

Figure 2. A sample XML Recipe Document

In the JSON implementation of MyRef, JSON (JavaScript Object Notation) follows a semi-structured format (see Figure 2). It is human readable and is not as rigid as the relational model. The basic constructs in JSON are recursively defined. JSON has typical basic atomic values such as numbers, strings, Boolean values, and no values. It has two types of composite values: objects and arrays. Objects are enclosed in curly braces and they consist of sets of label-value pairs (sometimes called "property"). Arrays, which are a list of values, are enclosed in square brackets with commas between the array elements. Like XML, JSON has some basic structural requirements; however, unlike XML, JSON format can be heterogeneous (e.g. non-uniform of attributes). (Widom, 2013)


```
{"recipe":[{
"recipeID":"1000",
"recipeName":"Poppadums with lime &amp;amp; coriander dip",
"recipeDescription":"Start your Indian meal in style with Meena Pathak's tangy dip - serve with
    plain poppadums",
"recipeRating":"6 ratings 5",
"recipeDifficulty":"[id:20120729025638009.27][link:http:\/\/www.bbcgoodfood.com\/images\/recipes\/easy_btn.gif]
    [name:easy_btn.gif]",
"recipeServing":"8 poppadums","recipePrepTime":"Ready in 5 minutes",
"recipeCookTime":"",
"recipeMethod":"Put the mango chutney in a bowl and chop up any large pieces of mango. Mix with the spring
    onions, coriander and lime juice. Cover and chill until ready to serve. (You can make this up to a day ahead
    and keep it in a covered container in the fridge.) Serve the chilled dip in a bowl on a plate, with the
    poppadums broken in pieces and placed around the sides.",
"recipeNutrients":"227 kcalories, protein 8g, carbohydrate 27g, fat 10 g, saturated fat 2g, fibre 3g, sugar
    5g, salt 2.19 g",
"recipeIngredients":"4 tbsp mango chutney 1 bunch spring onions 4 tbsp fresh coriander 5 tbsp lime juice (about
    3 limes) 8 ready-to-eat poppadoms",
"recipeImage":"[id:20120729025638009.28][link:http:\/\/www.bbcgoodfood.com\/recipes\/1691\/images\/1691_MEDIUM.
    jpg][name:1691_MEDIUM.jpg]"}]}
```


Figure 3. A sample JSON Recipe Document

3.2 Implementation

For the SQL implementation, the database management system MySQL was used. Upon submission by the user of some selected ingredients, the system develops the query string to be passed on to the database connection by concatenating the POST variables (in this case, the ingredients selected). A database connection is established by using the mysql_connect() function in PHP. The constructed query string is then passed to the database connection, which handles the MySQL connection to PHP; and then executes the query string in the database by using the mysql_query() function in PHP. A result set is then generated, which provides the information needed by the client. These are the recipes that include one or several ingredients from the list. Parsing of the result set is being handled by PHP with the use of the ***mysql_fetch_array()*** function in PHP, which returns an array of recipes. From this array, the program then displays the information to users with the use of HTML codes.

For the XML, two different versions are implemented. The first version makes use of only one (1) XML document file, which contains all the 1568 recipes. The second version includes 1568 XML document files, one file for each recipe. For both the versions, the XML file(s) were used as data source. Each file is opened (***fopen***) and scanned by a PHP function

for each of the POST variables (ingredients selected). Once the ingredient(s) are found, the recipes are then added to an array which contains all the information needed by the user. The information are then parsed and shown/returned to the user using HTML codes.

Quite similar with the XML implementation, the JSON implementation adopts two approaches – single and multiple files. In order to decode the JSON files, the system opens the file(s) and uses the json_decode() function to translate the JSON object into an array. Similar to the XML implementation, the array that was provided by the decode function is scanned and matched with the ingredients selected by the user. When a match(es) is found, the relevant information is stored in another array; this is then parsed and shown/returned to the user using HTML codes.

4 RESULTS AND DISCUSSION

Experiments involving 10 query cases were conducted to determine the efficiency/performance of the five database implementations. Each test query was executed 5 times and the average execution time was calculated. In recording the observations for each test case, two timestamps are used. A timestamp is recorded every time a query is submitted and every time the result of the query is returned. The absolute difference between the two timestamps is used to measure the amount of time (in seconds) it takes to execute each query. Based on the results involving all test 10 query cases (see Table 3), the SQL implementation retrieved the result set of recipes the fastest, followed by JSON and then by XML.

Table 3. Query Performance Results of 5 Database Implementations

Ingredients	No. of recipes returned	SQL (sec)	XML multiple files (sec)	XML Merged file (sec)	JSON multiple files (sec)	JSON merged file (sec)
Zest	174	0.1125	0.7873	0.7443	0.6207	0.1792
Zest, yogurt	11	0.0608	0.7736	0.7128	0.4598	0.1704
Zest, yogurt, sugar	5	0.0569	0.7942	0.7245	0.4389	0.1784
Zest, yogurt, sugar, egg	2	0.0532	0.7799	0.7189	0.3716	0.1823
Zest, yogurt, sugar, egg, cheese	1	0.0826	0.7754	0.7181	0.4452	0.1843
Zest, yogurt, sugar, egg, cheese, cocoa powder	0	0.0539	0.7787	0.7452	0.4513	0.1884
Broccoli	29	0.0483	0.7663	0.7105	0.4285	0.1684
Broccoli, chicken breasts	4	0.0477	0.7684	0.7377	0.4395	0.1789
Broccoli, chicken breasts, olive oil	2	0.0491	0.7881	0.7121	0.4479	0.1770
Broccoli, chicken breasts, olive oil, almonds	1	0.0485	0.7751	0.8155	0.4724	0.1800

5 CONCLUSIONS AND FUTURE WORK

Efficiency is the overriding criteria we used to determine empirically which among the relational model, XML, and JSON is better with respect to query processing. Our results indicate that in terms of speed of querying, the relational implementation retrieved the fastest result set. We surmised that the relational implementation performed better because we used the MySQL environment to implent the system and run the queries; while for the XML and JSON implementations, we retrieved the results programmatically using PHP. Also, all 10 test cases (queries) made use of only one table. For future work, we plan to use query languages for XML and JSON, as well as adopt NoSQL systems (e.g. MongoDB). We also plan to include more complex queries that utilize more than one table. We also plan to implement other domains such as social networking.

6 REFERENCES

Elmasri, R. and Navathe, S., (2010). Fundamentals of Database Systems (6th edition), Addison-Wesley

Garcia-Molina, H., Ullman, J.D., and Widom, J., (2007). Database Systems: The Complete Book (2nd edition), Prentice Hall.

Ramakrishnan, T., and Gehrke, J., (2002). Database Management Systems (3rd edition), McGraw-Hill Science/Engineering/Math.

Silberschatz, A., Korth, H. and Sudarshan, S., (2010). Database System Concepts (6th edition), McGraw-Hill Science/Engineering/Math

Ullman, J. D. And Widom, J., (2007). A First Course in Database Systems (3rd edition), Prentice Hall.

Widom, J. (2013). Introduction to Database. (An on-line course) Stanford University. http://class2go.stanford.edu/db/Winter2013.