MongoDB: An open source alternative for HL7-CDA clinical documents management

Gómez Adrián, García Eijó Francisco, Martínez Marcela, Analia Baum, Luna Daniel, González Bernaldo de Quirós Fernán

Health Informatics Department – Hospital Italiano de Buenos Aires – Argentina

Abstract

One of the main needs of health care systems, are related to the ability to manage and process large volumes of data stored in heterogeneous clinical repositories. This issue is a topic of interest in the field of health informatics. This paper describes a new alternative for the clinical information retrieval, stored in HL7 Clinical Document Architecture repository, using a document database, NoSQL, schema free, in a laboratory environment.

Keywords


Introduction

The evolution of science applied to the management of information and communication has created new data storage technologies in heterogeneous and unstructured formats, with the capability of management of complex data structures. These technologies related to the current needs of health information systems, where patient’s clinical record, has multimedia information, which is usually composed of contextual data, analytical data and digital images [1].

The communication and information transfer needs are also represented in this area, where different new standards for clinical documents exchange, have brought new challenges to efficient storage and retrieval of information [2].

From these standards, we highlight the HL7 organization, as the most important international organization in this topic, with the goal of establishing a general framework for medical information exchange [3][4].

Clinical Document Architecture (CDA) is an XML-based document tags standard, derived from a reference model (HL7 RIM) which allows you to specify the structure and semantics of a clinical document, in order to facilitate the exchange of information, document management and integration of data that support patient care [5], [6].

CDA infrastructure is widely used by health organizations, creating large repositories of relevant clinical information, stored in a standard format.

The Italian Hospital clinical information system, is composed of an electronic medical record, implemented in the outpatient, inpatient, emergency and home care, consisting of a transactional data repository and a clinical data repository based on CDA, in a network health services provider with 50,000 admissions / year and 2.5 million outpatient visits / year [7].

This scenario presents the issues related to efficient management and retrieval of large data volumes[8][9].

There are several publications where the study problem focuses on the storage of XML data, presenting different complexities in managing heterogeneous data, comparatively others analyze XML file processing versus object-oriented repository or native XML databases, presenting favorable results in those technologies that use databases, introducing themselves as the best choice for processing large volumes of information. [10], [11], [12]

Based on these studies, we performed a proof of concept, using NoSQL databases, Schema Free, as an alternative tool for the management and processing of large volumes of heterogeneous clinical information.

Legacy systems mounted on transactions technologies are not included in the object of study, we focused on the analysis of efficient retrieval of clinical information contained in CDA documents

Materials and Methods

The Italian Hospital clinical data repository is based on the standard HL7-CDA and has a little more than 22 million documents. From this repository, a sample of 1,000,000 random clinical documents was obtained, defining a
grouped set in 31% of lab results, 22% of clinical notes, 28% results of imaging studies, 10% of drug prescriptions and 9% of nursing sheets.

**Architecture and Technology.**

For processing the documents a MongoDB database, documentary, NoSQL, schema free in the version 2.0.6 - 64-bit Ubuntu Server mounted on 64-bit, Intel Core 2, 4Gb RAM was used. The prototypes for the test were developed with the Python programming language in version 3.0 implemented in Windows 7 64-bit, Intel Core 2, 4 Gb RAM.

**Testing Methodology**

For the analysis of the performance 4 different types of test were used, all of which are described below:

- **Baseline Test:** This first test was run on the database created for analysis. Without any optimization, consultations were carried out with a minimum number of concurrent users, to determine a baseline of performance.

- **Load Test:** Once the causes of performance problems obtained from the initial test are established, necessary actions are implemented and Load Test runs simulating the concurrency of users of the productive environment.

- **Stress Test:** We test the limits of the processing system increasing the concurrency to reach the saturation of computational resources.

- **Soak Test:** We run a Load Test over long periods of time to find memory leaks.

Of these procedures we obtained average response times, number of documents retrieved, average time per document and we also measured the computational resource consumption by using the SAR command.

**Query Group**

Q1. Selection of a specific document knowing its identifier.

Q2. Selection of documents in which text information is assessed.

Q3. Selection of documents where numerical data are evaluated.

Q4. Selection of documents in which numerical data is evaluated and logical operator "and".

Q5. Selection of documents in which date type data are evaluated.

Q6. Selection of documents in which the data are evaluated within an array.

Q7. Selection of documents in which date type data are evaluated using logical operators "greater than" "less than"

Q8. Selection of documents in which assess different types of data are evaluated and a combination of logical operators "greater than" "less than" and "and" are used.

**Examples**

Q1. `db.coleccion.find({"_id": 4fa289d7da0908694a4bd2d5})`

Q2. `db.coleccion.find({Query: "VITELLI" })`

Q3. `db.coleccion.find({ Query: "348"})`

Q4 `db.coleccion.find({$and:[{Query:"348"}, {Query: "638"}]})`

Q5. `db.coleccion.find({ query: "19700806"})`

Q6. `db.coleccion.find({query: {"$all": ["348","369","6520","6521"]}})`

Q7. `db.coleccion.find({ "Sall": ["348","369","6520","6521"]})`
XML Data Conversion

Because the CDA clinical data repository is stored in an XML format FileSystem, it was necessary to perform a conversion to JSON format, native format used by the database, in order to conduct the study. For this process an adhoc application was developed with the ability to read XML files, convert JSON format and then import them into the database MongoDB. Data conversion of XML -> JSON is possible using a set of rules that have to evaluate conditions for each data type XML and JSON format equivalent representation.

Conversion Rules

- Feed is represented as a JSON object and each nested element attribute is represented as an object property attribute / value.
- The attributes are converted to property @attributes.
- The child elements are converted to object properties, if the node has attributes or child nodes.
- Repetitive elements become array properties.
- Text values of tags become properties of the parent node string if the node has no attributes or child nodes.

The result of this conversion is maintained JSON document semantics of logic nodes and attributes of the XML document.

CDA Version

```xml
  <typeId root="2.16.840.1.113883.1.3" extension="POCD_HD000040"/>
  <templateId root="2.16.840.1.113883.3.27.1776"/>
  <id root="2.16.840.1.113883.2.10.1.4.2" extension="HI194312-1"/>
  <code code="26436-6" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" display="ALL LABORATORY STUDIES"/>
  <title>INFORME DE LABORATORIO</title>
  <effectiveTime value="20120314081724"/>
  <confidentialityCode code="N" codeSystem="2.16.840.1.113883.5.25"/>
  <languageCode code="es-AR"/>
  <setId root="2.16.840.1.113883.2.10.1.4.3" extension="HI194312"/>
  <versionNumber value="1"/>
  <recordTarget>
    <patient>
      <name>
        <family>TROCCOLI</family>
        <given>JUAN ANTONIO</given>
      </name>
      <administrativeGenderCode code="M" codeSystem="2.16.840.1.113883.5.1" codeSystemName="AdministrativeGender"/>
      <birthTime value="19300203"/>
    </patient>
  </recordTarget>
</ClinicalDocument>
```

JSON Version

```json
{"ClinicalDocument":{
  "typeId":{"@attributes":{
    "root":"2.16.840.1.113883.1.3",
    "extension":"POCD_HD000040"}},
  "templateId":{"@attributes":{
    "root":"2.16.840.1.113883.3.27.1776"}},
  "id":{"@attributes":{
    "root":"2.16.840.1.113883.2.10.1.4.2",
    "extension":"HI194312-1"}},
  "code":{"@attributes":{
    "code":"26436-6",
    "codeSystem":"2.16.840.1.113883.6.1",
    "codeSystemName":"LOINC"}},
  "title":"INFORME DE LABORATORIO",
  "effectiveTime":{"@attributes":{
    "value":"20120314081724"}},
  "confidentialityCode":{"@attributes":{
    "code":"N",
    "codeSystem":"2.16.840.1.113883.5.25"}},
  "languageCode":{"@attributes":{
    "code":"es-AR"}},
  "setId":{"@attributes":{
    "root":"2.16.840.1.113883.2.10.1.4.3",
    "extension":"HI194312"}},
  "versionNumber":{"@attributes":{
    "value":"1"}},
  "recordTarget":{
    "patientRole":{
      "id":{"@attributes":{
        "root":"2.16.840.1.113883.2.10.1.1.1",
        "extension":"817139"},
        "patient":{
          "name":{
            "family":"TROCCOLI",
            "given":"JUAN ANTONIO"},
            "administrativeGenderCode":{"@attributes":{
```

Results

The set of queries on the basis of database state was run, without any optimization. *Table 1* shows the number of documents retrieved and the average time it took each query in milliseconds. The percentages of CPU consumed by each query shown in *figure 1* with an average value equal to 42% are also presented. From this analysis we can point out, that although the recovery times of the documents are admissible, we notice a large consumption in the percentages of CPU used by the system and further noted that much of the funds were retained by the system in implementation process, these observations were compounded when the application using multithreading technology was implemented, running queries concurrently, reaching saturation of computational resources when the simulation was over 25 concurrent users.

![Figure 1 - Without any optimization](image)

To avoid saturation of computational resources and better use of the CPU time, the queries were analysed, creating database indexes required for each query. After this new implementation using a database indexed, we observed a significant decrease in the execution times of each query, avoiding the retention of resources for long periods of time, as well as a decrease in the percentage of CPU used. (*Table 2*) (*Figure 2*) (*Figure 3*). Remarkably, about multithreading execution, there was a great increase in the number of concurrent users in the simulation, the value stipulated target of 120 concurrent users supported by the system was not generating resource saturation.

<table>
<thead>
<tr>
<th>Table 1 - Query Results</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>Time (ms)</td>
<td>Documents (N)</td>
</tr>
<tr>
<td>Q1</td>
<td>125</td>
<td>1</td>
</tr>
<tr>
<td>Q2</td>
<td>924</td>
<td>86</td>
</tr>
<tr>
<td>Q3</td>
<td>1216</td>
<td>3285</td>
</tr>
<tr>
<td>Q4</td>
<td>2026</td>
<td>1204</td>
</tr>
<tr>
<td>Q5</td>
<td>930</td>
<td>6</td>
</tr>
<tr>
<td>Q6</td>
<td>1917</td>
<td>2695</td>
</tr>
<tr>
<td>Q7</td>
<td>787</td>
<td>741</td>
</tr>
<tr>
<td>Q8</td>
<td>778</td>
<td>531</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2 - Query Results</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>Time (ms)</td>
<td>Documents (N)</td>
</tr>
<tr>
<td>Q1</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>Q2</td>
<td>119</td>
<td>86</td>
</tr>
<tr>
<td>Q3</td>
<td>96</td>
<td>3285</td>
</tr>
<tr>
<td>Q4</td>
<td>297</td>
<td>1204</td>
</tr>
<tr>
<td>Q5</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>Q6</td>
<td>470</td>
<td>2695</td>
</tr>
<tr>
<td>Q7</td>
<td>125</td>
<td>741</td>
</tr>
<tr>
<td>Q8</td>
<td>153</td>
<td>531</td>
</tr>
</tbody>
</table>
The purpose of stress test is to increase the number of concurrent users using iterative query executions to estimate the values of concurrency that saturate the computational resources of the system. The values obtained represent the process in figure 4, which shows that the system is stable within parameters of less than 170 users, generating after these values an exponential increase, saturating the computational resources when concurrency reaches values close to 300 users.

Soak Test execution did not introduce memory leaks problems, this test was running for 168 hours, with 120 users concurrent executions in 2 seconds. The performance of the database remained within the parameters set as a goal, presenting adequate response times, a good cache management, noting a decrease in the response time on repetitive queries and a storage system which presented an average growth rate of 1.3%.

**Discussion**

There are several studies in the literature that analyze comparatively processing XML files versus object-oriented repository or native XML databases, showing favorable results in those technologies that use databases, being the best choice for processing large volumes of information. [11],[12],[13],[14]

Based on these studies, we analyze a new technological tool, as a possible alternative for the handling and processing of heterogeneous clinical information. We believe that the work has limitations due to the observation of parameters obtained in a laboratory environment, but also highlight the presence of high levels of impact and relevance, because clinical information retrieval in heterogeneous models is a topic of great importance in the field of health informatics.

**Conclusions**

While there are various options in the market for management and retrieval of XML information, either, native databases or components used in transactional databases, the implementation of a document database,
schema free proved to be consistent in complex data management and heterogeneous schemas.

Remarkably, the database has a structure based on JSON language, proving to be simple to implement in Web environments. In addition, JavaScript and Java natively support the language, without the need for external libraries.

MongoDB has additional components that will be evaluated in future work, focusing on the ability to work in distributed computing schemes and on the implementation of the Map Reduce method.

References


Address for correspondence
Adrian Gomez
Health Informatics Department
Hospital Italiano de Buenos Aires
Juan D. Perón 4190 (C1181ACH) Buenos Aires, Argentina
Adrian.gomez@hiba.org.ar