RELATIONAL DATABASES VERSUS INFORMATION RETRIEVAL SYSTEMS: A CASE STUDY

Ozgur Yilmazel, Burcu Yurekli, Baris Yilmazel, Ahmet Arslan

Anadolu University, Computer Engineering Department
Eskisehir, Turkey

ABSTRACT
In this paper we present our evaluations of using an Information Retrieval library in a commercial employment website with over 300,000 searches a day. Comparison of existing relational database system and the proposed system has been evaluated over 2 million queries. It’s shown that using an information retrieval library provides 20 times faster results compared to current implementation.

KEYWORDS
Information retrieval, relational databases, solr, unstructured data.

1. INTRODUCTION
As Internet becomes widely accessible, the amount of free form text data available on the websites have increased substantially, and as a result, searches on textual data gain importance. Many businesses started to base their revenue solely on full text search. Full text search has some unique problems that require special attention. In this paper we present a case study how we help convert a commercial employment website’s job search functionalities from Relational Database to an Information Retrieval Library. We present our approach and provide details about the performance achieved by this conversion.

2. BACKGROUND
Starting from the early days of computer use in businesses relational databases has been the prevalent choice for storing and managing business data. Early 90’s businesses started using the Internet and World Wide Web to disseminate information and some early adopters started creating businesses to conduct transactions over the web. These early applications on the web relied on relational databases. Increased use of computers and internet for business and personal use increased the amount of unstructured data (emails, pictures, customer comments, reports and a like) that many businesses need to manage. It is currently estimated that 80% of the data a business have is in unstructured format, and most of this data is in text form [1].

Relational Databases had shortcomings in handling unstructured data. They are designed to provide search results that satisfy the user information need 100% because queries are built on structural field constraints, and also lack a ranking mechanism for the results. When searching over unstructured data, ranking mechanism is very important. Most users examine top 10 or 20 results and ignore the rest; therefore results must be sorted by relevance in order to satisfy user’s information need.

With the increase in unstructured text, developments of information retrieval systems have been gaining momentum. The aim of IR libraries is to perform fast full text search specifically on free form text data. There are over 17 open source information retrieval libraries available with different features [2].

Many database vendors (IBM DB2 [3], Microsoft SQL Server [4], MySQL [5], Oracle [6], PostgreSQL [7]) have recognized the need for free form text search and started to implement features that would support full text search capabilities. Table 1 shows the years and versions that relational database vendors first started to introduce full text search functionality. Unlike structured information access, unstructured information
access or free text search is language dependent. However, for many business organizations preferred way to handle text data is still to use relational database management systems regardless of the language used in their textual data.

<table>
<thead>
<tr>
<th>DB Vendor</th>
<th>Establishment year</th>
<th>Full text introduction year</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM DB2</td>
<td>1983</td>
<td>1995</td>
<td>5</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>1989</td>
<td>1998</td>
<td>7</td>
</tr>
<tr>
<td>MySQL</td>
<td>1995</td>
<td>2000</td>
<td>3.23</td>
</tr>
<tr>
<td>Oracle</td>
<td>1971</td>
<td>1999</td>
<td>8i</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1989</td>
<td>2001</td>
<td>7.0.3</td>
</tr>
</tbody>
</table>

Basically there are two ways to support full text search using relational databases. It is either to use %LIKE% operator or to use relational databases’ full text search functionality (if it exists). It is common in commercial websites to use LIKE %term% operator instead of using full text search capabilities of relational databases. Many commercial websites use keyword search: online newspaper sites, online shopping sites, e-commerce sites, forums, online dictionaries, etc.

Employment websites, which provide online tools for job seekers to find suitable jobs and for employers to manage and advertise job openings, are one of these websites.

In this research an IR system has been implemented and integrated to one of the famous employment websites in Turkey. IR System ran parallel to the existing relational database system with full text search capabilities. For user queries, the response times of these two systems were compared.

3. CURRENT SYSTEM

Current system uses Linux, Apache, MySQL, and PHP (LAMP) to serve users.

All job ad data were stored in several normalized MySQL tables. For the keyword search of the system, MySQL %LIKE% operator was used. The user interface on the website was implemented by PHP. And there is an Apache Web Server running on Linux.

The field that %LIKE% operator run on is the concatenation of the fields that the clients wanted to perform full text search on, is created only to support these searches.

4. NEW SYSTEM

Search web page of the employment website is analyzed. After identifying the fields that are searched by the user and displayed to the user, a similar system is implemented by using Solr [8] with the exact functionality. Solr is an open source enterprise search server based on the Lucene [9] Java search library. Queries are constructed by combining four structured fields and one keyword field. Results can be sorted by three of the structured fields.

Since the data is stored in a database, to index this data the data import handler feature of Solr is used. Data Import Handler (DIH) [10] is a useful contribution that is used to import data from various data sources into Solr. In this work JDBC data source – that can load and index data directly from SQL databases – is used. There are two main import types: full-import and delta-import. Full-import is used to import all data from the database into Solr Index. Delta-import is used for incremental imports. Time stamp info of both import executions (either full or delta) is saved as last import time.

DIH requires addition of two additional columns to database tables: last modified column and a Boolean column for soft deletion. Last modified column is used by delta-import to fetch updated or newly added rows since the last import time. Boolean column is also used by delta-import to fetch inactive rows which will be deleted from the Solr index. Current main table already has such Boolean column that holds activity info for job ads which can be used for soft deletion. Therefore the last modified column is added to the main table and this was the only modification we made to the current system’s database.
Full-import takes about 45 seconds while delta-import takes less than half second to execute. A cron-job is scheduled to invoke delta-import every minute. This ensured that any changes in job postings were reflected in the search index within 60 seconds. Figure 1 visualizes summary of integrated information retrieval system (right part) that works parallel with the current system (left part).

Figure 1. System Overview

5. SYSTEM SETUP

All of the experiments were completed on the following configuration:
- 8 Dual-Core AMD 2.612 GHz processors with 20 GB of RAM - Linux
- Solr 1.3.0 running on Apache Tomcat 6.0.18

6. EXPERIMENTAL RESULTS

It has been showed by Arslan et al. [11] that Lucene in conjunction with a Turkish analyzer produced best retrieval quality for the Turkish dataset, when compared with Relational Databases, in terms of retrieval quality. In this work the results on time benchmarking are presented.

Current system has 5057 active, total 104926 job ads and gets 300,000 hits a day on average. Size of the inverted index on the hard drive is about 125.3 Megabytes. The query, response time of the two systems, and session ids were logged. The log contains 2,245,705 queries.

Table 2 shows minimum, average and maximum response time of each system. Solr has an 8.8 milliseconds response time on the average, while MySQL has a 78.2 milliseconds response time on the average. MySQL answered 32% of queries less than 1 second; while Solr answered 99% of queries less than 1 second.

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solr</td>
<td>1.37</td>
<td>8.88</td>
<td>6367.78</td>
</tr>
<tr>
<td>MySQL</td>
<td>0.77</td>
<td>78.20</td>
<td>53126.20</td>
</tr>
</tbody>
</table>

But if we consider that response time less than 3 milliseconds is optimal response time for a search engine, and eliminate queries that have response time less than 3 milliseconds for both systems we obtain 922,553 queries. This reduced subset is 41% of total queries with response time more than 3 milliseconds for both systems.
If we compare running times of each system on this subset of queries, Solr is faster at 92% of queries; MySQL is faster at 8% of queries. Table 3 shows statistics of this subset of queries. This analysis shows that speed up on slower queries is more important than speeding up already fast queries.

Table 3. Statistics of response times in milliseconds > 3

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solr</td>
<td>3.00</td>
<td>9.50</td>
<td>6367.78</td>
</tr>
<tr>
<td>MySQL</td>
<td>3.00</td>
<td>188.88</td>
<td>53126.20</td>
</tr>
</tbody>
</table>

If we analyze the incoming queries we observed that 85% of them are structured queries without keywords. Users didn’t enter any keyword and performed their search on structured fields only like sector, city, department, and application date. Probably to widen the search results. Only 15% of queries are full text queries. These queries have a length of 1.2 on the average.

If we compare running times of each system on full text queries only, Solr is faster at 72% of queries; MySQL is faster at 28% of queries. Table 4 shows statistics of full text queries. Solr is twenty times faster than MySQL in terms of average response time.

Table 4. Statistics of full text search response times in milliseconds

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solr</td>
<td>1.66</td>
<td>9.13</td>
<td>4103.16</td>
</tr>
<tr>
<td>MySQL</td>
<td>1.93</td>
<td>187.59</td>
<td>50294.20</td>
</tr>
</tbody>
</table>

7. CONCLUSION & FUTURE WORK

Our work showed that; relational database management systems are insufficient for full text searches. Usage of %LIKE% operator is too primitive compared to information retrieval. Open source software Solr, speed up eight times the average response time of all user queries, 20 times the average response time of full text queries. These enhancements allow serving same number of users with fewer numbers of servers. Using fewer servers reduces required hardware costs as well as their maintenance costs.

Up to now, we only deal with the response time and we did not focus on the retrieval quality of the job ad search system. To compare the retrieval quality of the system, we will implement a web interface that displays results from both systems as lists. The results that turned into job applications will be taken as relevant and the comparison of the retrieval quality of the systems will be done based on this feedback.

REFERENCES