Proceedings of the 10th WSEAS International Conference on COMPUTERS, Vouliagmeni, Athens, Greece, July 13-15, 2006 (pp744-752)

Web-based High Performance Full-Text Search Application

BALA NAGENDRA RAO BETHA, VENKAT SRINIVAS MODALI,
ASRITA CHETAN AVASARALA,
KALYAN KUPPACHI, SHANKAR KAMBHAMPATY
Satyam Computer Services Limited
C5, TSR Towers, Raj Bhavan Road
Somajiguda, Hyderabad – 500 082
INDIA
Phone: +91-40-55237373
{Balanagendra_betha, Srinivas_MV, Asrita_chetan, Kalyan_kuppachi,

Abstract: - The paper focuses on the business usage of Full-Text Search at database level and compares its strengths and weaknesses with respect to B-Tree Index Search. The paper establishes why Full-Text Search is better suited for searching textual content and why B-Tree Index is limited in given situations. The paper uses a real-life application "Persons of Importance" for propagating its message. Performance Management Tools were applied to compare the results of Full-Text Search with B-Tree Index in single-user and Multiple-user scenarios.

Keywords- Full-Text Search, B-Tree Indexes, .NET Framework, SQL Server 2000, SQL Server 2005, Oracle Text

1 Introduction

Full-Text search is a powerful database feature which could search large contents of catalogs and documents with high effectiveness and performance.

It is uses are many depending on requirement and situation. For example, a Hotel Rooms Reservation web site books rooms for different hotels or hotel chains across the world. A search capability is provided on the catalog of hotels. The catalog describes hotel's star rating, location, prices and hotel chain names etc. User searches the catalog depending on his convenience by providing location and star-rating of the hotel etc., while leaves out other details. The search result lists a number of hotels meeting the search criteria and the user chooses to narrow down the search to a smaller scope. The end result is, this form of search, is lot less frustrating than usual options provided with Lists and Radio Buttons etc. This type of search falls in Catalog Search category of Full-Text Search.

The paper is organized as follows. Section 2 presents the overview of the Full-Text search. Section 3 discusses the architecture of persons of Importance Application (POI). Section 4 discusses the deployment view of the application. Section 5 describes the Performance Management of Full-Text Search application in single-user and multi-user scenarios. Section 6 concludes the paper with an appendix.

2 Overview of Full-Text Search

Full-Text search came into popularity with Google, Alta Vista and Ask Jeeves like global search engines. All these search engines depend on an indexing technology which is different from B-Tree Index and is far superior in searching textual content.

In this context the paper discusses the Full-Text Search feature provided at database level. ANSI Structured Query Language (SQL) supports Full-Text Search capability with extensions like CONTAINS. [1], [2], [7], [8], [9]

2.1 Differences between B-Tree Search and Full-Text Search

SQL uses B-Tree Index when searching with an equiv-join. An equiv-join is where "=" operator is used. But to issue an equiv-join the exact values of the elements being searched must known. The other alternative is LIKE keyword, which allows wildcard search but is restrictive. The obvious fall-out of using wildcard search is optimizer bypasses B-Tree Index. As with any formatting of data etc., the B-Tree optimizer bypasses the index and opts for full table search and hence performance hits a roadblock.

Full-Text goes beyond this searching capability. Full-Text depends on a special purpose index which has more intelligence built into it than a B-Tree Index. It also provides many programmable features such as Thesaurus, Stemming, Rooting and Grammatical variations etc. Full-Text allows indexing catalog items, MS Word Documents, PDF documents etc. During the indexing process the engine removes noise items and identifies key words with are relevant to the document. The relevance of a key word to the content of the document is given a score. A score of 100 means the keyword is very relevant to the document while lesser scores mean less relevance to the document. [7], [4], [9]

2.2 Advantages of Full-Text Search

- Full-Text Search is easy to build in to the application. It is non-invasive on the application. It could be developed as a plug-in to an existing application and search box could be added at many places in the screens.
- It is entirely at database layer without any changes to business layer or presentation layer.
- It is configurable at any stage of application development life cycle without disturbing the existing data model.

2.3 Disadvantages of Full-Text Search

Full-Text Indexes are not automatically refreshed. Changes made to database are not reflected in the Full-Text Index. The Index has to be refreshed periodically to reflect those changes. End-user's buy-in is required for the delayed refresh. However, there is some work-around to overcome this limitation of Full-Text Search.

2.4 Uses of Full-Text Search

For a beginning its uses are clearly visible in the following domains:

- Pharmaceutical Industry: In a pharmaceutical application names of drugs are searched frequently. A LIKE operator search is not as effective as a Full-Text search. There are some advanced features like Thesaurus which could be applied to search synonymous names. An categorized form of index is Ontology which facilitates search reduced to a particular field.[1]
- Travel and Logistics industry: As mentioned earlier, in a Hotel Rooms Reservation Application. Names of all the hotels and cities could be put in Full-Text index and searched on a frequent basis.
- Banking and Finance Industry: Names of customers, cities, products and services could be put in a Full-Text Search.

2.5 Tuning Full-Text Search

Full-Text is entirely fine-tunable to achieve high performance.

- SCORE is a feature built in to the SQL which gives the degree of relevance of a searched keyword to that of the result. A SCORE of 100 is high relevance while SCORE of 0 is no relevance at all.

Using SCORE one can restrict the search results to an acceptable level. [3]

- Noise elimination. An XML file provides the hints to the Full-Text Indexing engine which words to be treated as noise and eliminated before index is created. Words like “A”, “AS”, “IS”, “ARE” are treated as noise and eliminated fro index. This reduces the unwanted words from index and improves search performance. [1]

2.6 When to use B-Tree and when not to use Full-Text Search

Full-Text Search is not panache, a medicine to all diseases. It is useful only certain situations where textual search is required. While B-Tree Indexes are going to remain the forte of On-Line Transaction Processing (OLTP) applications like Bank Transaction Management Systems etc.

2.7 Expanding Full-Text Search

Thesaurus: An example of thesaurus is given below. It is known that names like TONY, ANTHONY and ANTONY are all synonymous. So, while searching for TONY, the search engine should look into ANTHONY and ANTONY as well. When included all the variations of the name in the search, the search results are far more effective.

There are examples of commonly used words and their equivalent botanical terms. For example, ONION is a commonly used term for ***Allium Triquetrum***. While searching for ONION, if the search engine includes documents containing both the terms, the results would be very effective. The thesaurus could be configured using an XML file which contains all the synonymous words and is included before creating the index for Full-Text Data. [4], [5]

3 Case Study

Persons of Importance Application (POI), gathers personal details about important people like politicians, industrialists, businessmen, terrorists, bankrupt businessmen or people from sanctioned lists generated by national security agencies. There are two categories of users for the POI. The “Editors” compile the data from whichever source it is available such as newspapers, magazines and autobiographies etc. While the “End-Users” use, information thus gathered, for searching with any combination of first name, middle name, last name, country of origin, country of residence and date of birth etc. This database is used in tracking the activities and movements of such people for National political and financial safety.

3.1 Salient Features: The salient features of the architecture are the following [4]:

3.1.1 Scalability: A loosely coupled architecture allows scalable solutions. The application could cope to peak loads within SLA.

3.1.2 Performance: The architecture guarantees performance SLA of 5+ seconds of response time for multi-users scenarios. It is feasible to generate such high performance only using features of Full-Text search. B-Tree Indexes would not deliver the desired effect or performance.

3.1.3 Maintainability: Due to componentization the application is easily maintainable.

3.1.4 High-availability: The architecture is highly available with limited outages and crash-proof recovery systems.

3.1.5 Security: Using Access Control Service and Enterprise-wide Single-Sign-On, the architecture is secure.

3.2 Layered Approach: A layered approach of architecture, addresses best the concerns of encapsulation and separation of User Interface from Business Logic Layer and Business Logic Layer from Data Persistence. [10], [11]

Separation of one layer from another secures the application from most frequently used web hacking methods like SQL Injection, Parameter Tampering, Information gathering and Cross-site scripting. Figure-1 shows the layered diagram of the architecture.

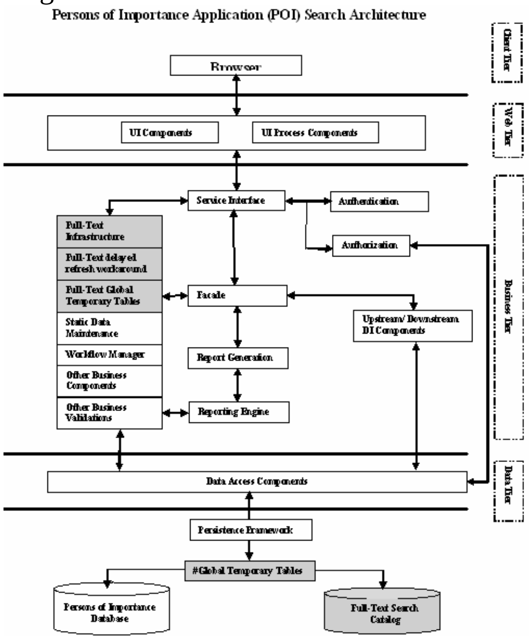

Fig 1: Persons of Importance Search Architecture

3.2.1 User Interface Layer/Presentation Layer: This layer groups together a set of customer facing screens such as User Interface (UI) Components and UI Process Components. UI Components are a collection of presentation services like HTML/CSS, XML/XSL and C# Web Forms. While UI Process Components, decouple the navigational and other presentation logic from individual forms. It encapsulates the dependencies between forms and the logic associated with forms.

3.2.2 Business Logic Layer: Most of the work is accomplished at business tier. MS .NET Framework and MS Enterprise Library Framework (MS-ENTLIB) are used for implementing business logic. A few of the patterns in business logic layer are Service Interface, Façade and Interfaces to External and Third party systems. Presentation layer talks only to Service Interface for any business functionalities and in turn Service Interface passes the request to appropriate services. Façade encapsulates non-core functionalities of the system and allows plug-and-play of external tools such as Reporting Engines into the applications. Authentication and Authorization are a few external interfaces which interact with the application. Other components of Business Logic Layer are:

Full-Text Infrastructure: Stored Procedure to fetch the data using Full-Text Search Infrastructure provided in SQL Server ordering according to the relevance score of the subject in question.

Full-Text delayed refresh workaround: Unlike B-Tree Indexes, Full-Text Search does not update the index immediately when changes are done to core tables. The Full-Text Index should be refreshed to reflect the changes made to core tables. Although, Full-Text provides incremental refresh, considering the size of the database and its user load, a delayed index refresh approach is adopted. But between refreshes all changes to database are be stored in a temporary table as a log. Including the log in the stored procedure gave us the workaround for delayed refresh and user's changes were reflected instantly in the query results.

Full-Text Global Temporary Storage: This storage was meant to reduce network traffic. Instead of sending the search results from data access layer to business logic layer in one go, it was retained in a Global Temporary Table and sent one page-full at a time. The global temporary table is initialized every time a new query is fired. But the results of query are available for view page-by-page. One global temporary table is created for each connection session.

Example: Say, a search resulted in 3000 rows. Now, passing them across network to Business Tier would result in 3.0 Mbps of data. If 100 users issue such queries simultaneously then there is no question of network not getting clogged. Instead all those rows were retained for a session in a table in database and passed only 30-40 rows at a time to the Business Logic Layer. This reduced network bottlenecks and fastened the application response time.

The other services that manifested are for updating the static data, work flow management, business computational components and validation components.

3.2.3 Data Access Layer: This layer separates the data access functionality from the business services layer. It is a thin layer where data access components interact with Relational Database. All the Data Manipulation (DML), Audit Trails and Exception Management are managed from this layer.

3.2.4 Full-Text Catalog: Full-Text Catalog in SQL Server resides outside the database. It is a datafile which can store many such indexes. Size of the catalog is proportional to the source table content. The catalog is not updated when changes are happening to the main database. Either an explicit refresh or an incremental refresh only updates the catalog. Thesaurus is another major feature of catalog which allows synonymous terms. So that names with same meaning such as "Tony" and "Anthony" are treated as one and the same. [7]

Table-1 Highlights of the application architecture and non-functional requirements user load, SLA etc.:

Database	MS SQL Server 2000, SQL Server 2005. T-SQL Stored procedures
Web Server	Microsoft IIS 6.0
Development Suite	Microsoft Visual Studio 2003 Enterprise Architect Edition Microsoft .NET Framework 1.1 Microsoft Enterprise Library 1.0 (MS-ENTLIB)
Operating System	Microsoft Windows Server 2003 Standard Edition
Approximate Database size	400+ MB
Table size	PERSON_OF_IMPORTANCE 400,000+ rows
Application type	Thin Client, Web based internet enabled application working from any standard internet browser.
Approximate user load	At any time 200+ users from all parts of globe.
Service Level Agreement (SLA)	5+ seconds for multi-user scenarios.

3.2.5 Data Architecture: The core of the application revolves around the Full-Text Search Infrastructure. The success of the application depends on how well it uses the features of Full-Text Search. Following is an example from main application to high-light the effectiveness of Full-Text Search in comparison with B-Tree Index Search. It is using the scaled down version of the PERSON_OF_IMPORTANCE table which stores the personal details over 400,000 persons.

The table contains three separate columns for First_name, Middle_name and Last_name. For the sake of indexing the Full-Text all the three names are clubbed together and put into single column namely FULL_NAME. This column was used for Full-Text Index.

```
Table PERSON_OF_IMPORTANCE
 Person_ID
 Person_category,
 First_name,
 Middle_name,
 Last_name,
 Country_of_origin,
 Country_of_residence,
 Date_of_birth,
 Full_name ...
```

A search for "Tony Blair" is issued using B-Tree Index in SQL with the LIKE operator. Then the same table is searched using Full-Text Infrastructure with CONTAINS clause. The results are compared and effectiveness of Full-Text Search is found to be far greater than B-Tree Index Search. Another salient feature of the search is LIKE operator can not detect if data is stored as "Tony Blair" or "Blair Tony". While Full-Text Search could identify such reverse combinations. There are many variation of the name "Tony". It can be spelled as "Anthony" or "Antony" etc. The thesaurus feature of Full-Text Search identifies those variations and treats them as one and the same. The following examples high-lights these features:

Example-1: Search using B-Tree Index (with LIKE operator)

```
SELECT   FULL_NAME
FROM     PERSON_OF_IMPORTANCE
WHERE              FULL_NAME    LIKE
%Tony%Blair%';
```

Results of the query are as follows:

```
Full_Name
Tony Blair
Tony Charles Lynton Blair
Tony Charles L. Blair
```

The same query is rephrased with Full-Text Search capabilities:

Example-2: Search using Full-Text Infrastructure

```
SELECT   FULL_NAME
FROM     PERSON_OF_IMPORTANCE
WHERE    CONTAINS (FULL_NAME,
         '"Tony*" AND "Blair*"')
```

Results of the query are as follows:

```
Full_Name
Blair, Anthony Charles Lynton
Tony Blair
Blair Tony
Tony Charles Lynton Blair
Tony Charles L. Blair
Anthony Blair
Anthony Charles Lynton Blair
Anthony Charles L. Blair...
```

The CONTAINS clause provides lot more flexibility than LIKE could provide. Thesaurus in Full-Text is an XML file which tells the search engine that "Tony", "Antony" and "Anthony" are synonymous. Hence both the names are included in search. Building thesaurus is a one-time activity but is very useful in retrieving variations in data. Thesaurus could be used in building ontology of synonymous medical terms.

4 Deployment View of the Application

The application is deployed at Satyam Global Data Center. The configuration of the servers is as follows:

- Web Server - HP PROLIANT DL360 G4 Intel XEON-DP 3.40GHz, 2GB RAM, 2x72 HDD
- Database Server - HP PROLIANT DL380 G4 Intel XEON-DP 3.20GHz/1MB, 4GB RAM, 2x36GB HDD, 4x72GB HDD

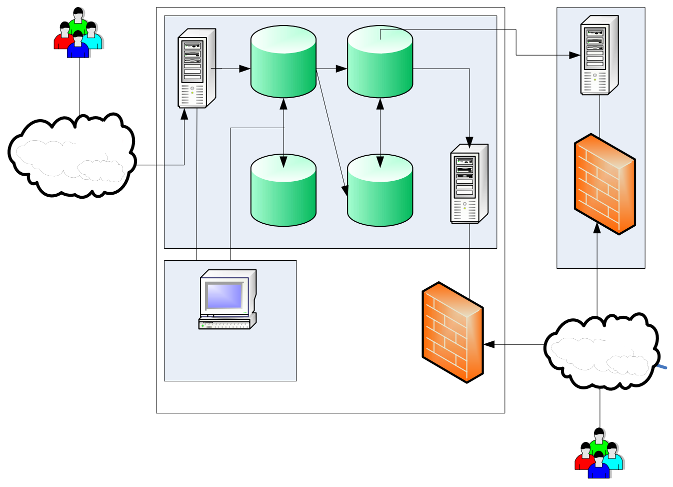

Fig 2: Deployment view of the application at Satyam Data Center

5 Performance Improvement of the Application

Performance improvement of the application is done in a two-pronged approach:

- Performance measurement: Base-line performance measurement data is captured in single user and multi user mode for development and production servers. Probes are inserted in applications at appropriate place to identify time taken at database level, application level and browser level. Tools, like Load-Runner is used to simulate multi user scenarios. [12], [13]
- Performance diagnosis and Tuning: The captured data is diagnosed for possible bottlenecks and changes to designs and infrastructure is implemented.

5.1 Observations: Initially, when using B-Tree indexes, the search yielded response times not up to the expectations. The index is bypassed totally when wildcard was applied to search criteria. However, the problem is resolved once Full-Text Search is implemented into the application.

In multi-user tests (please refer to Chart-2), the observations are given below:

5.1.1 While some spikes are noticed, response time of Full-Text Search has generally yielded single digit response times. It is assumed that the spikes are due to the tests that are run on machines that have bare minimum CPU and memory configuration (Web Server - 1 CPU, 512 MB, DB Server - 1 CPU, 1 GB).

5.1.2 Response time with SQL Server 2005 seems to be twice as fast as SQL Server 2000 on Full-Text Search. Comparing tabs with multi-user test results done against SQL Server 2000 and SQL Server 2005 are shown in Chart-1.

5.1.3 It is also understood that, response time is sensitive to traffic on the network. To eliminate network traffic congestions, Global Temporary Tables are implemented and dependency on network bandwidth is reduced. Chart-1 shows Full-Text Search results with Global Temporary Tables.

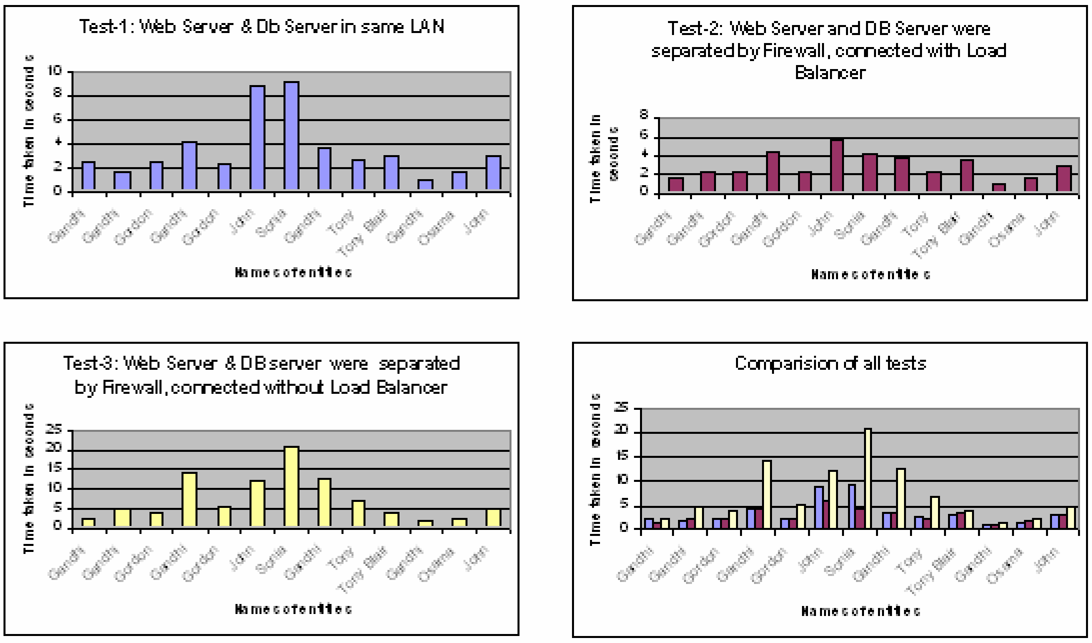

Chart1: Search results for load test carried out in single user mode

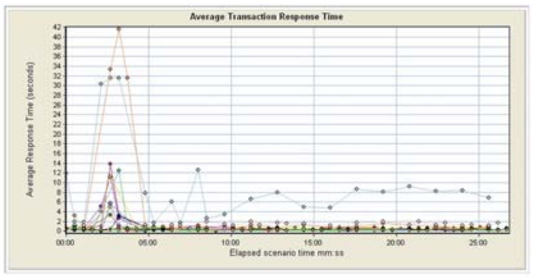

Chart-2: Multi-users tests for 20 users in SQL Server 2000

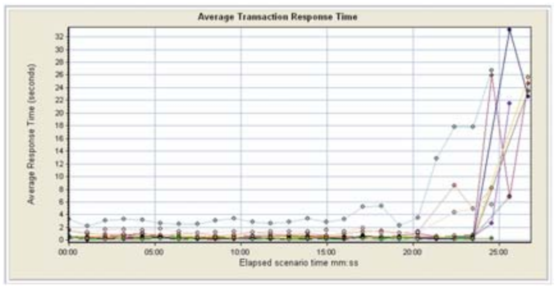

Chart-3: Multi-user tests for 20 users in SQL Server 2005

6 Conclusions

It is evident from the POI case study that Full-Text search is an efficient tool for searching large numbers of text files, documents or catalog fields. It is well supported by most of RDBMS. It may be implemented with little or no learning curve.

References:

[1] http://support.microsoft.com/kb/appliesto#appliesto SQL Server 2005 full-text search includes improved and updated noise word files

[2] http://forums.microsoft.com/MSDN/ Any full-text gurus - Help writing a query

[3] http://msdn2.microsoft.com/en-us/library/ms142557(en-us,SQL.90).aspx Performance Tuning and Optimization (Full-Text Search)

[4] http://msdn2.microsoft.com/en-us/library/ms142547(en-us,SQL.90).aspx Full-Text Search Architecture

[5] http://msdn2.microsoft.com/en-us/library/ms142541(en-us,SQL.90).aspx Full-Text Indexing Architecture

[6] http://www.databasejournal.com/features/mssql/index.php Full Text Search on SQL 2000 Part 1 Installation of SQL Server 2000 By Don Schlichting

[7] http://www.databasejournal.com/features/mssql/index.php Full Text Search on SQL 2000 Part 2 Searching Full-Text on SQL Server 2000 By Don Schlichting

[8] Oracle® Text Reference 10g Release 1 (10.1) Part Number B10730-02

[9] Oracle® Text Application Developer's Guide 10g Release 1 (10.1) Part Number B10729-01

[10] Microsoft, *Improving .Net Application Performance and Scalability*. Patterns and Practices, Microsoft Corporation ISBN 0-7356-1851-8.

[11] Microsoft's Patterns and Practices Group http://msdn.microsoft.com/practices/

[12] C. U. Smith and L. G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Addision-Wesley.

[13] Shankar kambhampaty and Srinivas Venkata Modali, Performance Modeling for Web based J2EE and .NET applications, *Transactions On Engineering, Computing and Technology , V8 OCTOBER 2005 ISSN 1305-5313*

Bala Nagendra Rao Betha, PMP works as Data Architect in Technology Architecture Group (TAG) of Satyam Computer Services Limited, India. He has over 20 years of experience in software industry and his area of interest is Relational Database Management System (RDBMS).

Venkata Srinivas Modali works with the TAG. His areas of interest include Software Architectures and Performance Engineering

Asrita Chetan Avasarala works with the TAG. His areas of interest include Microsoft Technologies

Kalyan Kuppachi works as a Project Manager with the Financial Services Group in Satyam Computer Services Limited, India. His areas of interest include Project Management and Microsoft Technologies.

Shankar Kambhampaty, Microsoft Certified Architect (MCA), **PMP** heads the TAG and has been involved for 16 years in architecture, design, development and management for a number of software projects, USA, UK, Singapore, Australia and India.