A Study on the DB-IR Integration: Per-Document Basis Online Index Maintenance

Du-Seok Jin, Hoe-Kyung Jung, *Member, KIMICS*

Abstract—While database(DB) and information retrieval(IR) have been developed independently, there have been emerging requirements that both data management and efficient text retrieval should be supported simultaneously in an information system such as health care, customer support, XML data management, and digital libraries. The great divide between DB and IR has caused different manners in index maintenance for newly arriving documents. While DB has extended its SQL layer to cope with text fields due to lack of intact mechanism to build IR-like index, IR usually treats a block of new documents as a logical unit of index maintenance since it has no concept of integrity constraint. However, In the DB-IR integrations, a transaction on adding or updating a document should include maintenance of the posting lists accompanied by the document. Although DB-IR integration has been budded in the research filed, the issue will remain difficult and rewarding areas for a while. One of the primary reasons is lack of efficient online transactional index maintenance. In this paper, performance of a few strategies for per-document basis transactional index maintenance - direct index update, pulsing auxiliary index and posting segmentation index - will be evaluated. The result shows that the pulsing auxiliary strategy and posting segmentation indexing scheme, can be a challenging candidates for text field indexing in DB-IR integration

Index Terms—Inverted File, Search Engine, Online Index Maintenance.

Manuscript received April 21, 2009; revised May 14, 2009.

Du-Seok Jin is with the Department of Information Technology Research, Korea Institute of Science and Technology Information, Daejeon, 305-806, Korea (Tel: +82-42-869-1777, Fax: +82-42-869-0779, Email: dsjin@kisti.re.kr)

Hoe-Kyung Jung(Corresponding author) is with the Department of Computer Engineering, Paichai University, Daejeon, 302-735, Korea (Tel: +82-42-520-5640, Fax: +82-42-520-5405, Email: hkjung@pcu.ac.kr)

I. INTRODUCTION

"DB and IR have evolved as separate communities for historical reasons. They were spawned in the sixties with focus on very different application areas: accounting and reservation systems on the DB sides, and library and patent information on the IR side. Consequently, they have emphasized different methodological paradigms: precise querying over schematized data, based on logic and algebra(DB), vs. keyword search and ranking over text and uncertain data, based on statistics and probability theory(IR). However, there are now many applications that require managing both structured and unstructured data and thus mandate serious consideration on how to integrate the DB and IR worlds at both foundational and software system levels." – Sihem Amer-Yahia, Djoerd Hiemstra and Gerhard Weikum [1].

DB-IR integration is a recent budding area of interest in the research fields of database area [2] and information retrieval area [3]. There have been several approaches to merge DB's structured data management and IR's text search facilities.

Web search engine is a kind of IRS customized to deal tens of billions of web pages. Web search engines commonly rebuild the index from the documents collected by the web crawlers periodically. Therefore, data management is not an important factor but the retrieval feature is essential for web search engines. In web search engines, web page contents are managed by web crawlers which can be regarded as extremely simplified database system. However, Recently IRS is being faced with user requirements such as transaction processing, join, materialized view, trigger, etc.

DB applications seem to be getting more and more user-oriented as opposed to the traditional database world. Examples are health care systems, XML data management, desktop search and personal information management systems. These applications are closer to IR world where awareness of human-user aspects has a long tradition. Thus has adopted many additional features which were out of interest in traditional DB world [4], including IR features.

Although their start points were different, we see, they will converge into single ridge of DB-IR integration. Recently, DB-IR integration has been

budded in the research field. However, the issue will remain difficult and rewarding areas for a while because of their limited online index update and transactional index maintenance.

In this paper, we propose two on-disk index strategies for per-document basis transaction of index maintenance and compare their performance with direct update strategy. The result shows that the pulsing auxiliary strategy and posting segmentation index scheme can be challenging candidates for text field indexing in DB-IR integration.

In Section2, we give a brief overview of related work on online index maintenance. Section 3 describes transactional index maintenance of which transaction unit is an incoming document rather than a block of documents in traditional index maintenance approaches in the IR field. Experimental methods and results are described in section 4. Finally section 5 summarizes the contributions of this paper.

II. RELATED WORK

Inverted files have been proved to be the most efficient data structure in high-performance retrieval systems for large text data [5]. There has been work on devising inverted lists that can efficiently handle document insertions, deletions or updates [6, 7, 8]. For efficiency, these work approached the index maintenance issue in the manner of "as many as possible documents" as a unit process. Pulsing technique, an in-memory buffering scheme keeping short lists in the vocabulary structure while pulsing long lists to on-disk supplementary heap file, was introduced in [9]. The studies in [10] explored space allocation strategies in in-place strategy where short lists are grouped in fixed-size buckets to reduce the problem of managing space for numerous small inverted lists. The granularity of propagation of index updates to disk was considered in [11]. A technique, called landmark-diff update method was introduced to improve the index update performance for existing documents of which contents are changed [12].

The QUIQ engine, a DBMS extender for text fields, applies index changes to the on-disk index based on a fixed time interval [13]. Works in IR usually propagate changes to disk driven by events such as exceeding a memory threshold or the number of updated documents [6, 7]. This way, disk accesses can be amortized over a large number of index update operations, resulting in increased indexing performance. However, under crashes, data integrity between documents and indexes can be damaged in these bunch of documents approaches.

Chunk method, where the document collection is divided into "chunks" of order documents' scores, is newly introduced to deal with ranking queries for structured data values in relational databases under update-intensive environment [14]. However, since chunk method was designed for numeric data types and for only update-intensive systems, it is difficult to be applied to text data types and insert-intensive environment.

III. TRANSACTIONAL INDEX MAINTENANCE

Since most of the index maintenance strategies lack the concept of integrity constraint, they have opportunities to regard a block of new documents – for efficiency – as a unit of index maintenance. However, towards DB-IR integration and to meet the field requirements from database managers in text service area, index maintenance should be supported in the on-disk storage level, not in-memory. This means the logical unit of index maintenance process should consist of each document (not multiple documents) and its accompanying index information. Furthermore, to support ACID property of database systems, an integrated DB-IR must support logged processing of index maintenance. We have tested three strategies for per-document basis index maintenance regarding each document and its index as a transaction unit: naïve *direct index update*, *pulsing auxiliary index*, and *posting segmentation index*.

For all experiments in this work, every change in index data is logged to disk in per-document basis transaction unit. If the transaction –i.e., insertion of a new document- is successful, the log is discarded. On the other hand, if the transaction fails for some reason, the log is used to roll-back to the state of just before the transaction. Since our pulsing auxiliary index approach contains two separate on-disk indexes, we call primary index for existing documents as main index and supplementary index for newly incoming documents as auxiliary index. Also posting segmentation index approach contains a posting segmentation function; we call PS(x), it finds segmentation for a direct access to read/write posting information, not a sequential access.

A. Direct Index Update

The direct index update is an intuitive and very simple index update strategy. Whenever a document is arrived to the system, the posting list for each term is appended to the main index. In general this requires relocating existing on-disk postings list for each term in the new document. Since a new document will usually have hundreds of unique terms, this kind of

direct, in-place update for main index requires hundreds of time-consuming relocations of postings list in the main index. Overallocation of postings list – which leaves some amount of free space after every postings list – in the main index, is proposed to relieve these costly operations [7]. However, overallocation size is not predictable in real databases and overallocation can be degenerated into garbage if there is few update of addition of documents, or falls into same situation of no-overallocation after a huge amount of documents are inserted hence filling over-allocated free spaces in the posting list. Therefore we do not regard the overallocation strategy in this experiment.

In direct update strategy, query processing performance will be comparable to totally re-built databases since postings lists are guaranteed to be contiguous. However, the frequent relocations of postings list in the main index will degrade the update performance seriously. As a typical document contains hundreds of distinct terms, such as update involves hundreds of disk accesses, a cost that is only likely tolerable if the database is very small or the rate of update is very low indeed.

B. Pulsing Auxiliary Index

Direct update strategy is expected to suffer from frequent relocations of long postings list. If the size of relocation is reduced to be small enough, the overall update performance can be improved. To reduce each relocation size, we introduced an on-disk pulsing auxiliary index for incoming documents. Whenever a new document is arrived, the indexing results are appended to the auxiliary index rather than to the main index. At initial state, the auxiliary index is empty or small, hence new postings lists from the incoming documents will be short and can be easily inserted to the auxiliary index. However as new documents are accumulated, postings lists in the auxiliary index grow in size resulting in deteriorated performance due to frequent relocations of grown postings lists.

To reduce the number of long postings list to process simultaneously index the auxiliary index, every auxiliary postings list longer than a given threshold is in-place updated or pulsed to the main index and discarded from the auxiliary index. This way, there is no postings list longer than the given threshold, and updating of postings lists for every unique terms in a document can be very small compared with the direct update index strategy. Therefore, inserting a document will have hundreds of unique terms to be processed but the transaction consists of hundreds of updates of small postings lists in the auxiliary index. Figure 1 shows the pulsing Auxiliary index with a threshold of DF=3.

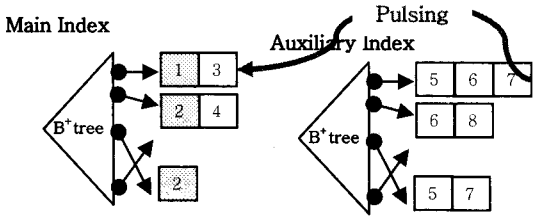


Fig. 1 Pulsing Auxiliary index with Pulsing Threshold of DF=3

C. Posting Segmentation Index

In both the direct update strategy and the pulsing auxiliary index update scheme, 'delete' of a document is achieved simply tagging it as deleted in the delete list and update of a document is processed as delete and insert. However, sometimes true live update is better than virtual delete or virtual update. When a document updated very frequently or continuously, the virtual strategy lays a burden on index maintenance gradually. This often leads to increase of the update processing time because as a document is updated, the size of postings list in the index growing more and more.

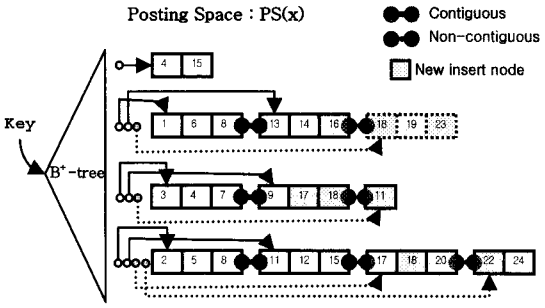


Fig. 2 Segmentation Index Structure with segmentation size of DF=3.

We now introduce another update strategy, posting segmentation index method, which can adapt itself to the true live update schemes under update-intensive environment. The key idea is to divide the long postings list into multiple short lists which are updated directly. Therefore, since the index size is not growing under update-intensive processing. In our segmented index structure the update performance will not be degenerated significantly after a large amount of documents are updated. Figure 2 shows the segmentation index structure with a segmentation threshold of DF=3.

IV. EXPERIMENT & RESULTS

We experimented with one million bibliographic data for journal and proceeding articles, and measured index maintenance times and querying times for direct, pulsing and segmentation strategies. Experiments were performed on a dual Pentium Xeon 3GHz machine with 8GB of memory and RAID-5 SCSI storage. We prepared two sets of text collections form 1 million and 10 thousands of bibliographic data. A set of the first 10 thousand items is referred to as 10K set (12MB in ASCII texts) and the million items to 1M set (1,310MB in ASCII texts). Each set was bulk-loaded and updated with the rest 10 thousand items (12.2MB of ASCII texts) for three update strategies (Experiments of direct update for 1M set was omitted due to time constraint). The different collection sizes were chosen to explore the maintenance cost of the three strategies with different amount of data. The sizes of the final indexes which contain full location information for all terms in 10K and 1M data sets were 29MB and 2,373MB, respectively.

Table 1 Brief database schema and statistics of index terms per-document

Section	Data Type	Index Type	Unique Terms	Total Terms
Title	string	token	11.2	11.9
Author	string	token	3.0	3.0
Journal	integer	token	5.0	3.4
Volume	integer	no index	-	-
Number	integer	no index	-	-
PubDate	char(8)	Index as is	1.0	1.0
Abstract	string	by token	85.3	143.0
Keywords	string	by token	2.5	2.4
SUM			107.9	166.8

For the pulsing and segmentation strategies, document frequency of 500 was chosen for pulsing threshold and document frequency of 1000 was chosen for segmentation postings lists, since it balances the trade-off between query processing and index maintenance. We compared times to process queries in Boolean model against a 1M+10,000 documents database of which index was constructed by re-build method, 1M database with 10,000 additional documents indexed with the pulsing auxiliary index and 1M with 10,000 documents indexed with the posting segmentation index strategy. Table 1 shows average term counts in every section in the text collection.

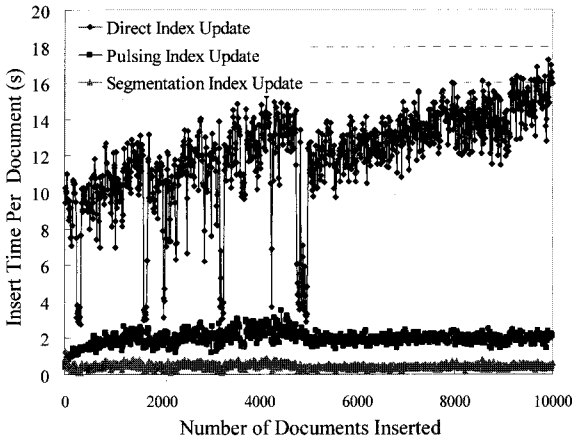


Fig. 3 Insert times for 10,000 input documents for 10K set

The results for three index update strategies are shown in Figure 3 and Figure 4. Each point in the figures was obtained by averaging insertion times of every 10 insertions, to reduce the effects of the biases in document lengths. Figure 3 shows the result of pouring 10 thousand documents to the 10K database where 10 thousand of documents were indexed. As mentioned Section 3.*A*, Figure 3 shows directly updating new documents' index information to the main index is very poor compared with pulsing auxiliary index or posting segmentation index approaches. This means direct update of main index is not feasible for even in very small databases. On the other hand, pulsing auxiliary index and posting segmentation index strategies can improve the update efficiency. Furthermore these strategies show nearly stable performance though many inserts are done.

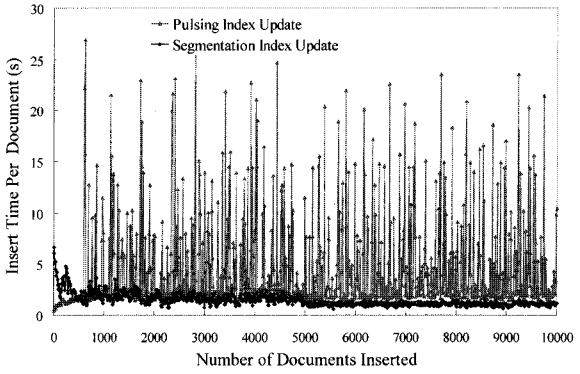


Fig.4 Insert times for 10,000 input documents for 1M set

In Figure 4, the experiments were done with same condition but with 1M database omitting direct update

strategy due to prohibitive time constraint of more than 3 minutes per document. Averaged over 10,000 insertion transactions to 1M database, the pulsing auxiliary index and posting segmentation index approach show the insertion speed of 4.38 and 1.46 seconds per document, respectively.

As in the pulsing strategy with non-contiguous nature of postings list, index maintenance inevitably contains a trade-off between update performance and query processing performance. The query processing times in figure 4 shows that the query processing performance of pulsing strategy is around 81% of re-build strategy. The main disadvantage of pulsing auxiliary strategy is that query processing is slower than re-build strategy. However, the query processing performance of our posting segmentation index strategy is slightly deteriorated while the magnitude improvement can be achieved in index maintenance. Our segmentation index strategy is not much different from a Re-Build inverted file structure except that the postings lists are stored in multiple files with a number of segmented posting lists.

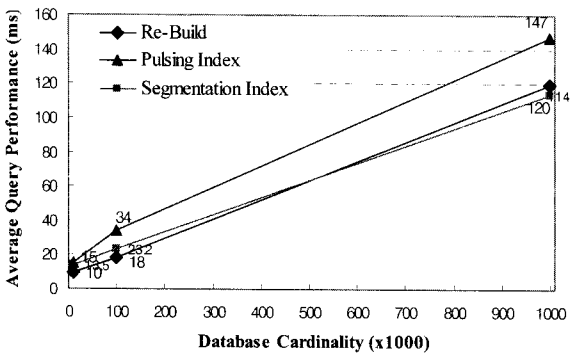


Fig. 5 Average query processing time for complex query after insertion of 10,000 new documents to 10K, 100K and1M sets

V. CONCLUSIONS

DB-IR integration has been budded in the research field. However, the issue will remain difficult and rewarding areas for a wile because of their limited online index update and transactional index maintenance. We have presented and evaluation of two per-document basis transaction strategies for dynamic index maintenance. In our experiments, the *pulsing auxiliary index* strategy has shows shortcomings of query processing performance. But the strategy has shows better performance of index maintenance than *direct index update* scheme. Our *posting segmentation index* strategy has also shows slightly deteriorated query processing performance. But the strategy shows the best performance of index maintenance in the case of all stage.

We believe *the pulsing auxiliary index and the posting segmentation index* strategies can be competing candidates for text indexing schemes in the DB-IR integration. Using the key idea of pulsing strategy, we will blend the pulsing auxiliary strategy and posting segmentation strategy in the future work.

REFERENCES

[1] S. Amer-Yahia, Djoerd Hiemstra, Thomas Roelleke, Divesh Srivastava, Gerhard Weikum, DB&IR Integration: Report on the Dagstuhl Seminar "Ranked XML Querying", *SIGMOD Record*, Sep. 2008, 37(3). pp. 46-49.

[2] S. Amer-Yahia, P. Case, J. Shanmugasundaram, T. Roelleke, and G. Weikum, Report on the DB/IR panel at SIGMOD 2005, 34(4). pp. 71-74.

[3] H. Bast and I. Weber, The completesearch engine: Interactive, efficient, and towards ir&db integration, *Third Biennial Conference on Innovative Data Systems Research(CIDR 07)*, Jan. USA, 2007, p.88-95.

[4] J. Gray. The next database revolution. *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, p.1-4.

[5] J. Zobel, A. Moffat, and K. Ramamohanarao, Inverted files versus signature files for text indexing. *ACM Trans. Database Systems*, 1998, 23(4), pp. 453-490.

[6] S. Buttcher, C. L. A. Clarke, and B. Lushman, Hybrid index maintenance for growing text collections. *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp.356-363.

[7] N. Lester, J. Zobel, and H. Williams, Efficient online index maintenance for contiguous inverted lists, *Inf. Process. Manage.*, 2006, 42(4), pp.916-933.

[8] N. Lester, J. Zobel, and H. Williams. In-Place versus Re-Build versus Re-Merge: Index Maintenance Strategies for Text Retrieval Systems. *In Computer Science 2004, 27th Australasian Computer Science Conference*, New Zealand, Jan. 2004, pp.15-22.

[9] D. R. Cutting and J. O. Pedersen. Optimization for dynamic inverted index maintenance. *In Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, 1990, pp.405-411.

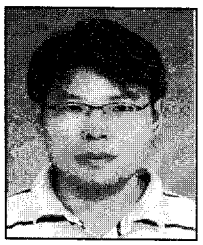
[10] A. Tomasic, H. Garcia-Molina, and K. A. Shoens. Incremental updates of inverted lists for text document retrieval. *In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, USA, May 1994, pp.289-300.

[11] T. Chiueh and L. Huang, Efficient real-time index updates in text retrieval systems, Technical Report ECSL-TR-66, Computer Science Department, SUNY at Stony Brook, 1999.

[12] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Agrwal, Dynamic maintenance of web indexes using landmarks. *In Proceedings of the 12th international conference on World Wide Web*, 2003, p.102-111.

[13] N. Kabra, R. Ramakrishnan, and V. Ercegovac, The QUIQ engine: A hybrid IR-DB system, *In Proceedings of the 19th International Conference on Data Engineering (ICDE)*, India, Mar. 2003, pp.741.

[14] L. Guo, J. Shanmugasundaram, K. Beyer, and E. Shekita, Efficient inverted lists and query algorithms for Structured Value Ranking in update-intensive relational databases. *In Proceedings of the 21st International Conference on Data Engineering*, 2005, p.298-309.



Du-Seok Jin
received the B. S. and M. S. d egrees in computer engineering from Chonbuk National Universi ty, Korea in 1999 and 2001, res pectively. Since 2001, he has been worked as a researcher in the Korea Institute of Science and Technology Information. He is currently interested in Information Retrieval System, Dynamic Index Structure and Data Management.

Hoe-Kyung Jung
received B.S degree in 1987, and Ph. D. degree in 1993, in the Department of Computer Engineering from Kwangwoon University, Korea. During 1994-2005, he worked for ETRI as Researcher. Since 1994, he has worked in the department of Computer Engineering at Paichai University where he now works as a Professor. His current research interests Multimedia Document Architecture Modeling, Information Processing, Web Services, Semantic Web, USN and MPEG-21.