

Journal of International Technology and Information Management

Volume 14 | Issue 3


Article 2

2005

Time Dimension for Business News in the Knowledge Warehouse

Pawel J. Kalczynski
The University of Toledo

Follow this and additional works at: <http://scholarworks.lib.csusb.edu/jitim>

 Part of the [Business Intelligence Commons](#), [E-Commerce Commons](#), [Management Information Systems Commons](#), [Management Sciences and Quantitative Methods Commons](#), [Operational Research Commons](#), and the [Technology and Innovation Commons](#)

Recommended Citation

Kalczynski, Pawel J. (2005) "Time Dimension for Business News in the Knowledge Warehouse," *Journal of International Technology and Information Management*: Vol. 14: Iss. 3, Article 2.
Available at: <http://scholarworks.lib.csusb.edu/jitim/vol14/iss3/2>

This Article is brought to you for free and open access by CSUSB ScholarWorks. It has been accepted for inclusion in Journal of International Technology and Information Management by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

Time Dimension for Business News in the Knowledge Warehouse

Pawel J. Kalczynski
The University of Toledo

ABSTRACT

This paper proposes a data structure to support the integration of business news and data in the knowledge warehouse using the Temporal Document Retrieval Model (TDRM). Temporal document retrieval takes into account the time context expressed as temporal phrases in the document content. The maintenance and scalability of the proposed data cluster is discussed. The results of temporal document retrieval experiments based on typical and ad-hoc time constraints, performed on a large corpus of business news are reported. The experiments showed that the instantaneous TDRM-based retrieval using the proposed cluster is possible with the currently available database technology and confirmed the highly constraining character of the temporal context.

INTRODUCTION

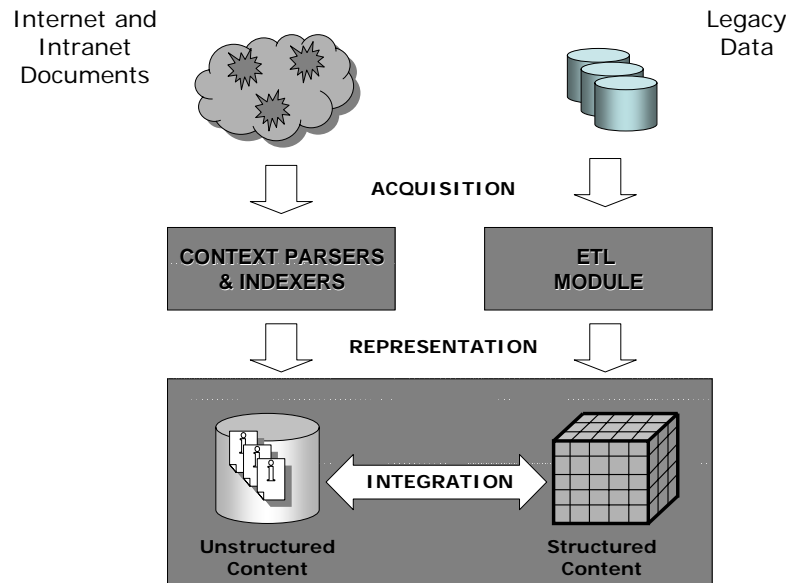
Since the dawn of data warehousing in the early 1990s (Inmon, 1992; Kimball, 1996), numerous extensions of the model were proposed. The major evolutionary concepts, i.e., Information Data Superstore (Bischoff & Yevich, 1996), Virtual Data Warehouse (Genesereth, Keller, & Duschka, 1997), Enterprise Information Portals (Shilakes & Tylman, 1998), Web Warehouse (Mattison, 1999), Web Farming (Hackathorn, 1999), Data Webhouse (Kimball & Merz, 2000), Active Data Warehouse (Thalhammer, Schrefl, & Mohania, 2001), Knowledge Warehouse (Nemati, Steiger, Iyer, & Herschel, 2002), and enhanced Data Warehouse (Abramowicz, Kalczynski, & Wecel, 2002) considered populating the data warehouse with data and information acquired from Internet sources. However, only a few models emphasized the need for a full integration of structured and unstructured data.

The idea of integrating documents with data in the knowledge warehouse is to instantaneously and automatically supply the users with a ranked list of business documents relevant to the current state of the data warehouse report being viewed (Abramowicz et al., 2002).

This task may be accomplished by applying constraints to the collection of documents stored in the warehouse. These constraints represent various contexts, such as subject, temporal, spatial, semantic, or personal (Wecel, Abramowicz, & Kalczynski, 2004). Presents the idea of context-based integration of documents and data in the data warehouse.

The focus of this paper is on integration of documents and data based on the time (temporal) context. The time context in the knowledge warehouse is typically defined by temporal constraints of the current state of the data warehouse report being viewed by the user (e.g. years, quarters, months, weeks, or days). These constraints can be easily extracted from the warehouse reports and used to subset the collection of documents stored in the data warehouse. In order for subsetting to be done efficiently, the temporal context of documents must be represented in the repository.

The theoretical TDRM paper (Kalczynski & Chou, 2005) proposed a model for representation of temporal content of documents and queries. The present work shows how TDRM can be implemented in an organization to integrate documents and data in the knowledge warehouse and extends the discussion (Kalczynski, 2004). A data cluster, integrated with the enterprise data warehouse, is proposed. The proposed model enables instantaneous temporal document retrieval for typical and ad-hoc time constraints. The present work also reports the results of technical feasibility and scalability analyses of the proposed structure.

Figure 1: Context-Based Integration of Documents and Data in the Knowledge Warehouse.

BACKGROUND

In the present work, the theoretical Temporal Document Retrieval Model (TDRM) (Kalczynski & Chou, 2005) provides the background for time-based integration of documents and data in the knowledge warehouse.

Before a document's time context may be described in TDRM terms, all temporal expressions such as "last week" or "tomorrow" must be extracted from the content (Abramowicz, Chmiel, Kalczynski, & Weceł, 2001; Goralwalla, Leontiev, Ozsu, & Szafron, 1998; Kalczynski, Abramowicz, Weceł, & Kaczmarek, 2003; Koen & Bender, 2000; Llido, Berlanga, & Aramburu, 2001) and represented as subsets of the calendar term space Ω . The calendar space is a finite, enumerable, and continuous list of calendar days (e.g. 1/1/1992, 1/2/1992 ... 12/30/1994, 12/31/1994). Each day in Ω serves as an equivalent of an indexing term in the traditional Vector Space Model (Salton & Lesk, 1968). In addition, unlike words or phrases, calendar terms are mutually independent (orthogonal) at this granularity level.

In TDRM, every document and every query are represented as comparable vectors of weights corresponding to calendar terms (days). The weights are based on the frequencies of terms in certain views of the document collection. Unlike in the traditional Vector Space Model (VSM), the frequency of calendar terms in TDRM is represented as a real number rather than an integer. This is due to the fact that when temporal expressions are transformed into calendar frequencies, *partial occurrence* of any term in the document is possible. For example, the temporal phrase "last month" extracted from a document published on 3/4/2005 is represented as 0.036 (1/28) of occurrence of each term (day) between 2/1/2005 and 2/28/2005. For some temporal phrases, such as "early last month," a fuzzy model (Kalczynski & Chou, 2005) is used to represent term frequencies. The term frequency diagram for a particular document is created by tallying frequencies for each respective term occurring in the document content. shows two term frequency diagrams for actual business documents.

A slightly modified Term Frequency – Inverse Document Frequency (TF-IDF) weighting scheme (Salton & Buckley, 1988) is used for determining the weights of document index terms in TDRM. TF is normalized over the maximum frequency and prioritizes terms that frequently occur in the document content. Conversely, IDF prioritizes terms that are rarely referenced by other documents in the collection. Raw IDF is a number of documents that contain the full reference ($\text{freq} \geq 1$) to a particular term. presents the raw IDF values for a large corpus of business news and the calendar space Ω ranging from 1/1/1992 to 12/31/1994.

Figure 2: Sample Document Term Frequency Diagrams.

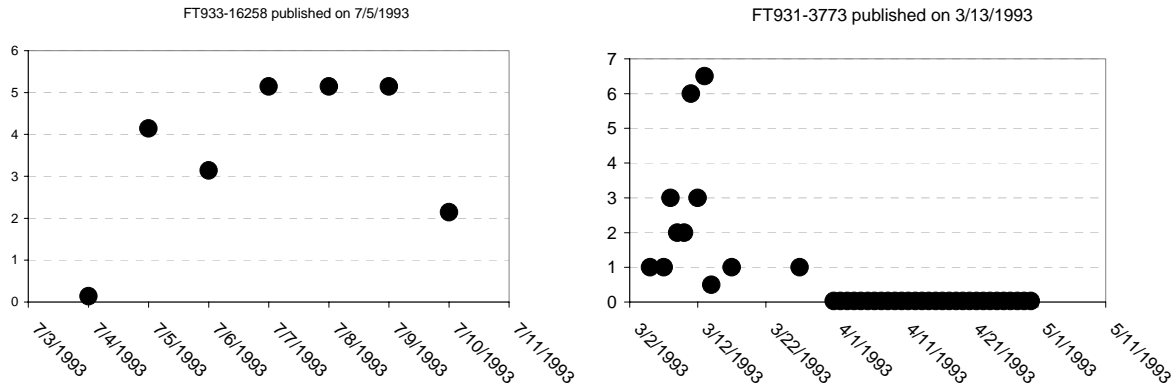
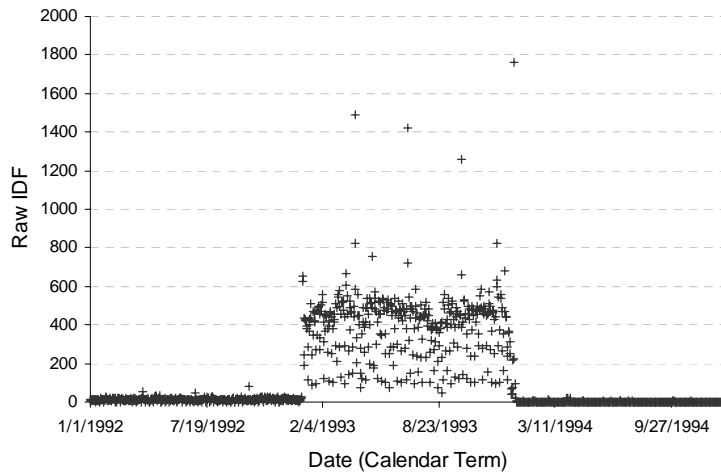


Figure 3: Raw IDF of FT1993.



Note that in the Temporal Document Retrieval Model, raw IDF is transformed to $IDF_i = \log(N/n_i)$ where N is the total number of documents in the collection and n_i is the number of documents with full reference ($freq \geq 1$) to the calendar term c_i . If c_i does not fully occur in any document, then n_i is set to .5 in order to emphasize the importance of the term for the query and avoid division by zero at the same time. The product of TF and IDF serves as weight for each calendar term referenced by the document. For query index terms $TF=1$; therefore they are all weighted with IDF values.

Let $p=|\Omega|$, let $w_{i,j}$ denote the weight of calendar term c_i referenced by document d_j , and let $w_{i,q}$ be the weight of term c_i in query q . Then the similarity between the document d_j and the query q , measured by the angular distance between their vectors of weights, is given by:

$$\text{sim}(d_j, q) = \left(\sum_{i=1}^p w_{i,j} \times w_{i,q} \right) / \left(\sqrt{\sum_{i=1}^p w_{i,j}^2} \times \sqrt{\sum_{i=1}^p w_{i,q}^2} \right). \tag{1}$$

Therefore, in order to compute similarity for a document-query pair the sum of squared weights for both vectors and the sum of weight products for matching terms must be determined.

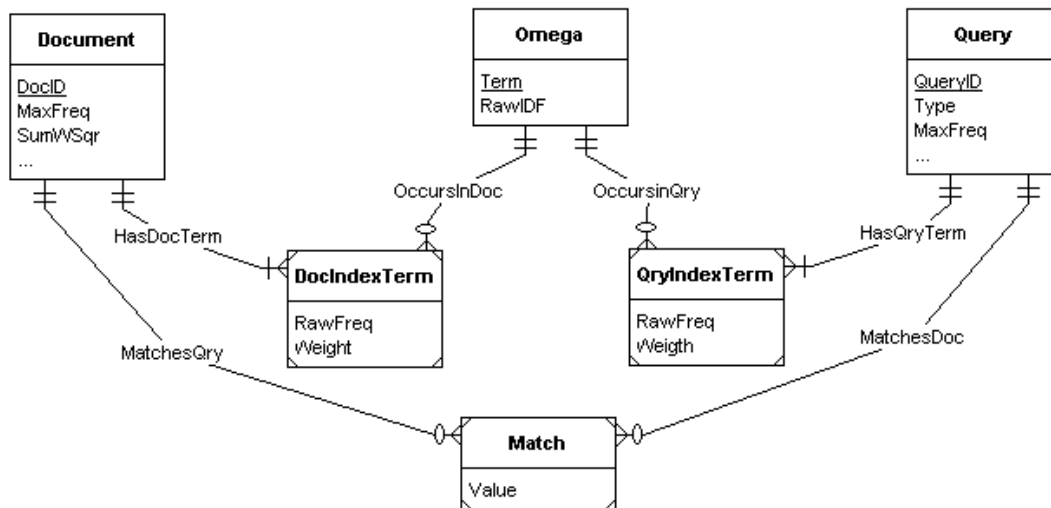
The following section will present a data cluster for TDRM-based integration of documents and data in the knowledge warehouse.

TIME DIMENSION CLUSTER

Logical Design

A typical business report is based on natural calendar granules; i.e., it refers to particular years, quarters, months, weeks or days. Such granules may be conveniently represented as TDRM queries. For example the first full week of March 2005 can be represented as seven subsequent calendar terms and their corresponding weights. The proposed data model takes advantage of this fact and stores pre-computed values for typical temporal queries and all documents in the collection, still enabling and facilitating ad-hoc temporal queries. Moreover, the model may be easily adjusted to work with different weighting schemes and similarity measures. presents the proposed data structure.

Figure 4: Time Dimension for Documents in the Knowledge Warehouse.



The *Document* table stores basic information about documents in the system. *Omega* stores the calendar terms (dates) and the corresponding number of documents in which the term fully occurs. *Query* stores all typical time constraints of business reports. Each calendar year defines 424 typical constraints, namely, one year, four quarters, 12 months, 52 weeks, and 365 (366) daily constraints.

DocIndexTerm and *QueryIndexTerm* are weak associative entity types which store raw frequencies and weights of calendar terms occurring in documents and queries. These structures provide efficient representations of sparse vectors in the term space Ω because no database records exist if weight=0. Finally, *Match* stores values which represent similarity between the two vectors (e.g. cosine) computed for all document-query pairs in the collection. The records in *Match* are derived from *DocIndexTerm* and *QueryIndexTerm*; no records are stored if there is no match (cosine=0).

Physical Design

It should be noted that physical design is a complex and on-going process. Therefore the present work describes just the initial choice of database indexes that may be further updated based on the cluster usage statistics. All experiments described in this paper were performed with Microsoft SQL Server 2000 running on a 3.8GHz desktop PC with 512MB RAM.

presents the auxiliary (non-primary key) indexes introduced to the Time Dimension (see) for the Knowledge Warehouse.

Table 1. Auxiliary Indexes for the Cluster

Table	Index	Clustered
DocIndexTerm	Term	No
	DocID	No
QryIndexTerm	Term	No
	QueryID	No
Match	QueryID	Yes

The proposed initial choice of database indexes focuses on efficient query processing (frequent) and fast updates (less frequent) of the cluster. In addition to the indexes, data usage statistics were added to most fields.

Most TDRM queries involve initial joining of two tables: *Query* and *Match* and return multiple records. Although the SQL Server Index Tuning Wizard recommended the use of multiple auxiliary indexes (including clustering and multi-column indexes), we found that this significantly affects the time it takes to rebuild the cluster. In fact, rebuilding the cluster with the Index Tuning Wizard recommendations took 39 times longer than with the proposed set of indexes. Thus, we believe that the initial choice of indexes, limited to keys (primary and foreign), is justified.

A slight modification in the physical design that eliminated primary-key constraints (clustered indexes) for *DocIndexTerm* and *Match* further sped up the cluster rebuilding process. Data integrity (entity and referential) was not a concern because both *DocIndexTerm* and *Match* are secondary repositories populated by automated processes.

The proposed cluster can be implemented using the database technology available to any organization running an enterprise database management system. It can be easily adapted to other variations of the VSM-based Temporal Document Retrieval Model (e.g. different weighting schemes or similarity measures).

In the following sections we show that the maintenance of the cluster is relatively easy and that it scales quite well. In addition, we show the results of temporal document retrieval experiments that confirm the highly constraining character of the temporal context.

MAINTENANCE OF THE TIME DIMENSION CLUSTER

Adding new documents or queries to the collection triggers a sequence of updates in the proposed time dimension. Adding a new query to the database is relatively easy because it only requires creating the query index and computing the match values for all documents in the collection. The process of adding a new document is far more complex and, theoretically, it requires (1) creating the document index, (2) updating all IDF values, (3) updating all document and query weights in the system, and (4) computing cosine values for every document-query pair.

Analysis of the Update Process

Let N be the current number of documents stored in the collection, let M be the total number of temporal queries in the system, and let p be the number of terms in calendar space Ω . After the document is processed with temporal indexer, its temporal index must be stored in the database. The index of document d_j consists of the set of C_j distinct calendar terms with corresponding frequencies and weights. Unfortunately all weights include IDF which is based on the number of documents (now $N+1$).

The average number of distinct days referenced by a single document from the TREC/FT 1993 collection (C_j) was found to be almost 301. The typical number of temporal queries in a year is 434. Let Q_i be the total number of distinct calendar terms referenced by a query; it is equal to the number of calendar days within the context of this query (e.g. $Q_i=7$ for weekly constraints). Upon addition of a single document to the collection C_{N+1} records must be

added to the *DocIndexTerm*, and $\sum_{j=1}^N C_j$ weights must be updated in this table. In addition $\sum_{i=1}^M Q_i$ records must be

updated in *QueryIndexTerm*, C_{N+1} records must be updated in *Omega*, N records must be updated in *Document* and $M \times (N+1)$ cosine values must be determined for the *Match* table. Computing the cosine value for a single document-query pair requires *DocIndexTerm* and *QueryIndexTerm* to be joined on *Term*.

Assuming that Ω extends over the period of five calendar years and $N=500,000$, over 200 million records would have to be modified upon addition of just one new document. Because the knowledge warehouse is designed to be constantly supplied with new documents, accepting hundreds of documents daily, this process may prove very inefficient. However, if we take advantage of the fact that, in a large collection of documents, IDF is not significantly affected by adding a few documents, we may reduce the number of physical record accesses. We therefore propose to update the IDF, the corresponding weights, and the match values periodically, say, daily, instead of doing that each time a new document is added to the collection. Then, the number of records added or updated upon arrival of a new document reduces to $C_{N+1}+M$. For the sample collection of 500,000 documents and $|\Omega|=1,826$ (5 years) this means about 650 records added or updated upon addition of a new document instead of 200 million.

Experiments

In the cluster-building experiment 64,138 documents published in 1992 and 69,162 documents published in 1993 were extracted from the Text Retrieval Conference (TREC) Financial Times (TREC/FT) collection and processed with the Time Indexer (Kalczynski et al., 2003). We extracted 423,272 (444,805) temporal references from the FT1992 (FT1993) collection, which averages 6.6 (6.4) references per document. The average number of daily publications for the FT1992 (FT1993) collection was 210 (227). Because only about 10% of all temporal references are imprecise (Kalczynski & Chou, 2005), for simplicity, all references in this experiment were represented as temporal intervals rather than fuzzy numbers. Nevertheless the proposed model supports the fuzzy representation of temporal references. As a result 20.5M (20.8M) document index terms were generated for all FT1992 (FT1993) documents in the analyzed collection. The respective Ω s were defined as 1/1/1991-12/31/1993 for FT1992 and 1/1/1992-12/31/1994 for FT1993.

Next, temporal queries representing time contexts of typical analytical reports were created for each collection. The queries extended over the respective Ω s with the following granules: three years, 12 quarters, 36 months, 156 calendar weeks, and 1096 days.

Monday was considered the first day of the calendar week; hence the first week in 1992 starts on Jan. 6, and the last week of 1994 ends on Jan. 1, 1995. The total number of query index terms turned out to be 5,474 (5,473) for FT1992 (FT1993).

Finally, the cosine values were computed for all document-query pairs in the collection and appropriate records were added to the *Match* table for all cosine ≥ 0.05 (the choice of the threshold is explained later). The process resulted in 1,070,409 (781,938) document-query pairs for FT1992 (FT1993). The corresponding time of rebuilding the entire cluster executed as a batch was 57 (36) min. The query batch used to rebuild the cluster can be found in the appendix. We also observed that the total disk space consumed by the cluster was 2.6 (2.9) GB for FT1992 (FT1993).

It should be noted that a significant improvement of the updating time has been achieved by removing most of the indexes recommended by the Index Tuning Wizard and adding appropriate statistics to the cluster.

Scalability

In the scalability experiment the FT1992 and FT1993 documents were loaded into a single cluster. Ω was defined as an interval 1/1/1991-12/31/1994. The purpose of the experiment was to check if the proposed structure scales well in terms of the update time, retrieval time and memory consumption given that the hardware system remains unchanged.

As a result a cluster of documents published in the Financial Times between 1992 and 1993 emerged. This revealed 41.3M document index terms in the *DocIndexTerm* table, 7,297 records in the *QryIndexTerm*, and

1,725,761 entries in the *Match* table. The total time for updating the composite cluster turned out to be 155 min. on the same desktop PC. The space consumed by the cluster was 5.5 GB. In addition 12GB of temporary space (tempdb) was required to rebuild the cluster.

In sum, daily updates of the proposed data cluster can be performed using the existing database technology.

TEMPORAL RETRIEVAL

The present section offers a description of the experiments with temporal document retrieval using the proposed cluster.

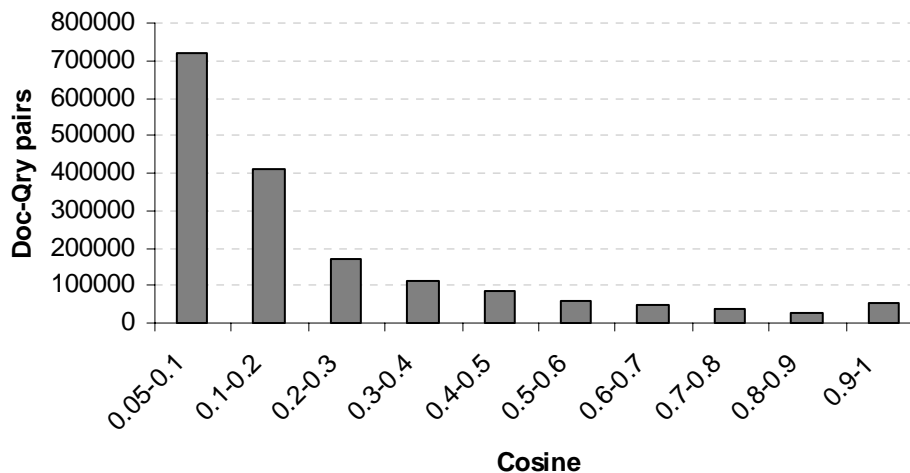
Generally, we distinguish two types of temporal queries: typical and ad-hoc. Typical queries are usually temporal intervals corresponding to calendar granules (e.g. “March 1993”, “the first week of May 1992”). Ad hoc queries, in turn, can be any subsets of the term space Ω .

Typical queries (common calendar granules) usually correspond to temporal contexts of most business reports. As the data warehouse user drills the report down or up, the corresponding list of documents relevant to the report should be displayed. Therefore, instantaneous retrieval of documents was the major goal of that research.

Typical Time Constraints

shows the distribution of cosine values for the typical calendar granules. About 6 million records which contained documents “relevant” to queries with $\text{cosine} < 0.05$ were dropped, i.e., a relevance threshold $t=0.05$ was introduced. This is due to the impact of temporal references like “this year” or “in 1991” which are quite frequent in business documents.

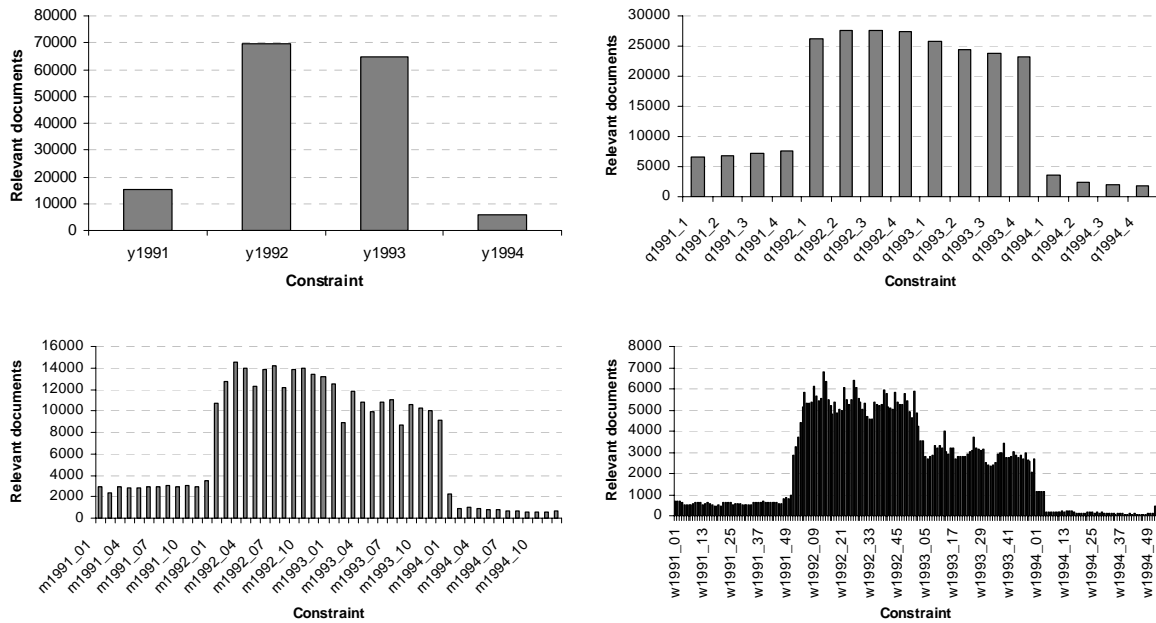
Figure 5: Distribution of Cosine Values ($t=0.05$).



Another interesting feature of the distribution of cosine values is the relatively large number of Document-Query pairs with cosine between 0.9 and 1.0. This is due to the fact that for a large number of documents in the collection, the publication date was the only temporal context extracted and such documents are strongly relevant to the corresponding daily granules in terms of cosine.

Error! Reference source not found.0 shows the percentage of documents of the FT1992 \cup FT1993 collection relevant to the typical time constraints for the assumed Ω .

Figure 10: Number of Documents Relevant to Typical Time Constraints.



The results confirm the highly constraining character of temporal retrieval. We suggest that the time context should be used before keywords because if a document is not relevant to time constraints specified it is also not relevant to keywords and vice versa. However, the assessment of the temporal relevancy is typically more precise than the assessment of relevancy in terms of keywords.

A typical query is in the form: “SELECT * FROM Match WHERE queryID='y1993' ORDER BY Cosine Desc.” We found that the execution time did not exceed one second for any of the 7,297 typical queries in FT1992∪FT1993 cluster. We concluded that the goal of instantaneous temporal retrieval has been achieved for typical time constraints.

Ad-Hoc Constraints

In order for an ad-hoc query to be processed within the proposed data model, it must be indexed, i.e., represented as a set of calendar terms. Because the weights are already computed for query and document terms and the sum of squared weights are stored for all documents, only the *Match* value must be determined for every document in the collection. This requires joining *DocIndexTerm* with the temporary table (representing the query) on *Term* and with *Document* on *DocID*.

As an example, the query “Weekends of January 1993” was indexed with ten calendar terms and appropriate weights were computed. Presents the details of the ad-hoc temporal query.

Table 2: Sample Ad-Hoc Temporal Query “Weekends of January 1993.”

QueryID	Term	TF-IDF Weight
ad-hoc1993_01	1/2/93	5.630778
ad-hoc1993_01	1/3/93	5.907765
ad-hoc1993_01	1/9/93	5.488215
ad-hoc1993_01	1/10/93	6.416819
ad-hoc1993_01	1/16/93	5.638875
ad-hoc1993_01	1/17/93	6.655571
ad-hoc1993_01	1/23/93	5.542088
ad-hoc1993_01	1/24/93	6.559239
ad-hoc1993_01	1/30/93	5.545785
ad-hoc1993_01	1/31/93	5.345114

It took 64 seconds to provide a ranked list of matching documents using the same set of queries as for typical time constraints. This result was certainly not satisfactory; hence a new approach to ad-hoc queries is described below.

Recall that, in the proposed cluster the similarity between a document d_j and query q is computed as:

$$\text{sim}(d_j, q) = \left(\sum_{i=1}^p w_{i,j} \times w_{i,q} \right) / \left(\sqrt{\sum_{i=1}^p w_{i,j}^2} \times \sqrt{\sum_{i=1}^p w_{i,q}^2} \right). \quad (2)$$

The *Match* table of the cluster contains cosine values computed for typical time constraints, including daily constraints. Daily constraints are very specific because each of them consists of only one calendar term α . Therefore the cosine value for a daily constraint is given by:

$$\text{sim}(d_j, q_\alpha) = \frac{w_{\alpha,j} \times w_{\alpha,q}}{\sqrt{\sum_{i=1}^p w_{i,j}^2} \times w_{\alpha,q}} = \frac{w_{\alpha,j}}{\sqrt{\sum_{i=1}^p w_{i,j}^2}}. \quad (3)$$

Further, we can transform the above equation into a formula for computing the weight of calendar term α when the similarity between this term and the daily constraint q_α is given.

$$w_{\alpha,j} = \text{sim}(d_j, q_\alpha) \times \sqrt{\sum_{i=1}^p w_{i,j}^2} \quad (4)$$

The set of K ad-hoc query terms $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ is a subset of Ω . For each document d_j the cosine value for the ad-hoc query can be computed using the pre-computed values stored in the cluster. By substituting each $w_{i,j}$ with corresponding values computed according to formula (4) we get:

$$\text{sim}(d_j, q) = \frac{\text{sim}(d_j, q_{\alpha_1}) \times w_{\alpha_1,q} \times \sqrt{\sum_{i=1}^p w_{i,j}^2} + \text{sim}(d_j, q_{\alpha_2}) \times w_{\alpha_2,q} \times \sqrt{\sum_{i=1}^p w_{i,j}^2} + \dots + \text{sim}(d_j, q_{\alpha_K}) \times w_{\alpha_K,q} \times \sqrt{\sum_{i=1}^p w_{i,j}^2}}{\sqrt{\sum_{i=1}^p w_{i,j}^2} \times \sqrt{w_{\alpha_1,q}^2 + w_{\alpha_2,q}^2 + \dots + w_{\alpha_K,q}^2}} \quad (5)$$

and finally, we get

$$\text{sim}(d_j, q) = \frac{1}{\sqrt{K}} \text{sim}(d_j, q_{\alpha_1}) \times w_{\alpha_1,q} + \text{sim}(d_j, q_{\alpha_2}) \times w_{\alpha_2,q} + \dots + \text{sim}(d_j, q_{\alpha_K}) \times w_{\alpha_K,q}. \quad (6)$$

The final formula enables computing cosine for ad-hoc temporal queries by joining just two tables: *Query* and *Match*. As a result, the ad-hoc queries execute instantaneously, i.e., in less than one second.

CONCLUDING REMARKS

This paper proposed a data model to support the time-dimension-based integration of document and data in the knowledge warehouse. The Temporal Document Retrieval Model (TDRM) provided a framework for determining the similarity between the temporal context of documents and queries. We demonstrated that it is possible to implement and maintain the proposed data structure with reasonable processing time and memory consumption. We managed to achieve instantaneous retrieval for typical and ad-hoc temporal constraints on a desktop PC running SQL Server 2000 and showed that the structure scales quite well.

The lack of explicit handling of temporal granularities can be considered a major disadvantage of TDRM. In particular, one should be able to match documents and queries assuming a given precision level. This precision level could be expressed in terms of calendar granularities. Although TDRM enables working with calendar granularities, all matching is performed on the finest granularity, i.e., day. Therefore, an extension of TDRM that would work with multiple granularities might be a potential research avenue to be explored. Such an extension would also require revising the database model presented in this paper.

Another disadvantage of the proposed model is that it works only for the time domain. Extending TDRM to other granularity systems (e.g. spatial or organizational) and introducing similar clusters for these retrieval contexts could be another potential research avenue. Perhaps a generic retrieval model for granularity systems could be developed.

Nonetheless, this model as it stands contributes significantly to practice and theory of temporal document retrieval because it supports the existing theoretical model, enabling its practical application in the business world.

APPENDIX

Views

```
CREATE VIEW DOCW_X_QRYW AS
SELECT DocIndexTerm.DocID, QryIndexTerm.QueryID, SUM(DocIndexTerm.Weight * QryIndexTerm.Weight)
AS DocWProdQryW
FROM DocIndexTerm INNER JOIN
  QryIndexTerm ON DocIndexTerm.Term = QryIndexTerm.Term
GROUP BY DocIndexTerm.DocID, QryIndexTerm.QueryID
HAVING (SUM(DocIndexTerm.Weight * QryIndexTerm.Weight) > 0)
```

```
CREATE VIEW COSINE AS
SELECT DOCW_X_QRYW.DocID, DOCW_X_QRYW.QueryID, DOCW_X_QRYW.DocWProdQryW /
(SQRT(Document.SumWSqr) * SQRT(Query.SumWSqr)) AS Cosine
FROM DOCW_X_QRYW INNER JOIN
  Query ON DOCW_X_QRYW.QueryID = Query.queryID INNER JOIN
  Document ON DOCW_X_QRYW.DocID = Document.DocID
WHERE (DOCW_X_QRYW.DocWProdQryW / SQRT(Document.SumWSqr) * SQRT(Query.SumWSqr)) >=
0.05)
```

```
CREATE VIEW MAX_FREQ
WITH SCHEMABINDING AS
SELECT DocID, MAX(RawFreq) AS MaxFreq
FROM DocIndexTerm
GROUP BY DocID
```

```
CREATE VIEW MAX_FREQ_REFDATE AS
SELECT MAX_FREQ.DocID, MAX_FREQ.MaxFreq AS NewMaxFreq, Document.MaxFreq
FROM MAX_FREQ INNER JOIN
  Document ON MAX_FREQ.DocID = Document.DocID
```

```
CREATE VIEW FOT AS
SELECT Term, DocID, RawFreq
FROM DocIndexTerm
WHERE (RawFreq >= 1)
```

```
CREATE VIEW FOT_IDF AS
SELECT Term, COUNT(DocID) AS fIDF
FROM FOT
GROUP BY Term
```

```
CREATE VIEW FOT_OMEGA AS
SELECT Omega.IDF, FOT_IDF.Term, FOT_IDF.fIDF
FROM FOT_IDF INNER JOIN
  Omega ON FOT_IDF.Term = Omega.Term
```

Maintenance Queries

```
UPDATE FOT_OMEGA
SET FOT_OMEGA.[IDF] = FOT_OMEGA.[fIDF]
FROM FOT_OMEGA
```

```
DECLARE @doccount bigint
SET @doccount = (SELECT Count(*) FROM Document)
```

```
UPDATE QryIndexTerm
SET Weight = Log(@doccount/IDF)
FROM QryIndexTerm INNER JOIN Omega ON QryIndexTerm.Term = Omega.Term
```

```
UPDATE Query
Set Query.SumWSqr=A.SumWSqr
FROM Query INNER JOIN(SELECT QueryID, SUM(Weight*Weight) AS SumWSqr
FROM QryIndexTerm
Group BY QueryID) A ON Query.QueryID=A.QueryID
```

```
DECLARE @documentcount bigint
SET @documentcount = (SELECT Count(*) FROM Document)
```

```
UPDATE DocIndexTerm
SET Weight = (RawFreq/MaxFreq)*Log(@documentcount/Omega.IDF)
FROM Omega INNER JOIN
DocIndexTerm INNER JOIN Document ON DocIndexTerm.DocID = Document.DocID
ON Omega.Term = DocIndexTerm.Term
```

```
UPDATE Document
Set Document.SumWSqr=A.SumWSqr
FROM Document INNER JOIN(SELECT DocID, SUM(Weight*Weight) AS SumWSqr
FROM DocIndexTerm
Group BY DocID) A ON Document.DocID=A.DocID
```

REFERENCES

Abramowicz, W., Chmiel, D., Kalczyński, P. J., & Wecel, K. A. (2001, May 20-23, 2001). *Time Consistency among Structured and Unstructured Contents in the Data Warehouse*. Proceedings of the 2001 Information Resource Management Association International Conference, Toronto, Canada, pp. 815-818.

Abramowicz, W., Kalczyński, P. J., & Wecel, K. A. (2002). *Filtering the Web to Feed Data Warehouses*. London: Springer-Verlag.

- Bischoff, J., & Yevich, R. (1996). The Superstore: Building More than a data warehouse. *Database Programming and Design online magazine*, Sept. 1996.
- Genesereth, M., Keller, A., & Duschka, O. (1997). Infomaster. *ACM SIGMOD*, 26(June 1, 1997), 539-542.
- Goralwalla, I. A., Leontiev, Y., Ozsu, T. M., & Szafron, D. (1998). *Temporal Granularity for Unanchored Temporal Data*. Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98), New York, pp. 24-31.
- Hackathorn, R. (1999). *Web Farming for the Data Warehouse*. San Francisco: Morgan Kaufman Publishers.
- Inmon, W. (1992). *Building a data warehouse*: QED Technical Publishing Group.
- Kalczynski, P. J. (2004, Nov 19-23, 2004). *Time Dimension for Documents in the Knowledge Warehouse*. Proceedings of the Decision Science Institute Conference (DSI 2004), Boston, MA, pp. 3011-3017.
- Kalczynski, P. J., Abramowicz, W., Wecel, K. A., & Kaczmarek, T. (2003, May 18-21, 2003). *Time Indexer: A Tool for Extracting Temporal References from Business News*. Proceedings of the 2003 Information Resource Management Association International Conference, Philadelphia, PA, pp. 832-835.
- Kalczynski, P. J., & Chou, A. (2005). Temporal Document Retrieval Model for Business News Archives. *Information Processing & Management*, 41(3), 635-650.
- Kimball, R. (1996). *The data warehouse toolkit*. New York: John Wiley & Sons, Inc.
- Kimball, R., & Merz, R. (2000). *The Data Webhouse Toolkit*. New York: John Wiley & Sons, Inc.,.
- Koen, D. B., & Bender, W. (2000). Time Frames: Temporal augmentation of the news. *IBM Systems*, 39(3&4), 597-616.
- Llido, D., Berlanga, R., & Aramburu, M. J. (2001). *Extracting Temporal References to Assign Document-Event Time Periods*. Proceedings of the Database and Expert Systems Applications, 12th International Conference (DEXA-2001), Berlin, pp. 62-71.
- Mattison, R. (1999). *Web Warehousing and Knowledge Management*: McGraw Hill.
- Nemati, H. R., Steiger, D. M., Iyer, L. S., & Herschel, R. T. (2002). Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing. *Decision Support Systems*, 33(2), 143-161.
- Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic retrieval. *Information Processing & Management*, 24(5), 513-523.
- Salton, G., & Lesk, M. E. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(January), 8-36.
- Shilakes, C. C., & Tylman, J. (1998). *Enterprise Information Portals* (November 16, 1998 ed.). New York, NY: Merrill Lynch, Inc.
- Thalhammer, T., Schrefl, M., & Mohania, M. (2001). Active data warehouses: complementing OLAP with analysis rules. *Data and Knowledge Engineering*, 39, 241-269.
- Wecel, K. A., Abramowicz, W., & Kalczynski, P. J. (2004). Web Services Modeling Framework for the enhanced Data Warehouse. In B. Montano (Ed.), *Innovations of Knowledge Management* (pp. 149-174). Hershey, PA: IRM Press.