[11] H.F. Korth, W. Kim, and F. Bancilhon, On Long-Duration CAD Transactions, Information Science, 46, pp. 73-107, Oct.1990..

[12] C. Beeri, P.A. Bernstein, and N. Goodman, A Model for Concurrency in Nested Transaction System, Journal of ACM, Vol.36, No.2, pp.230-269, 1989.

[13] G. Weikum, Principles and Realization Strategies of Multi-level Transaction Management, ACM Transaction on Database Systems, Vol.16, No.1, pp.132-180, 1991.

[14] H.F. Korth, and G. Speegle, Formal Model of Correctness without Serializability, in ACM SIGMOD International Conference on Management of Data, NewYork, pp.379-386,1988.

[15] H. F., Korth, G. Speegle, Formal Aspects of Concurrency Control in Long-duration Transaction Systems using NT/PV Model, ACM Transactions on Database Systems, Vol.19, No.3, pp.492-535, Sept.1994.

[16] S.K. Madria, Concurrency Control and Recovery Algorithms in Nested Transaction Environment, Ph.D. Thesis, Indian Institute of Technology, Delhi, India, 1995.

[17] Madria, S.K., Maheshwari, S.N, Chandra, B., Bhargava, B., Crash Recovery Algorithm in an Open and Safe Nested Transaction Model, 8th International Conference on Database and Expert System Applications (DEXA'97), France, Sept.97, Lecture Notes in Computer Science, Springer Verlag, Vol. 1308, 1997.

[18] W. Kim, R. Lorie, D. Mcnabb, and W. Plouffe, A Transaction Mechanism for Engineering Design Databases, in Proceedings of the 10th International Conference on Very Large Databases, VLDB Endowment, pp. 355-362, 1984.

An Efficient Conflict Detection Scheme for Concurrent Temporal Transactions

Bong-Ok Ha and Yoo-Sung Kim

Department of Computer Science & Engineering
INHA University
Inchon 402-751, KOREA
yskim@dragon.inha.ac.kr

Abstract

An efficient conflict detection scheme for Temporal DataBase Systems(TDBSs) is proposed. In TDBSs, a temporal transaction may access more data records than a transaction in traditional database systems because a TDBS usually manages both the historical versions and the current version of a data item. Hence, a concurrency control subsystem should be able to correctly and efficiently detect actual conflicts among concurrent temporal transactions while the cost of detecting conflicts is maintained in low level without detecting false conflicts which cause severe degradation of system throughput.

1. Introduction

Traditional database systems may be not appropriate to some applications such as hospital patient information systems since hospital patient information systems should manage not only current information but also historical information of patients. Instead of traditional database systems, temporal database systems(TDBSs) are appropriate to such applications, since TDBS allows users to access not only the current snapshot information but also the historical information of temporal database. Hence, TDBS has to manage the historical database in addition to the snapshot database of real world modelled into temporal database.

Since a TDBS may have several versions of a data item, the number of data items accessed by a temporal transaction must be larger than that in traditional database sys-

tems. Accessing large number of data items of a temporal transaction means that the execution time of the temporal transaction lasts longer than that of traditional transaction and at the same time, the overhead for conflict detection might be high due to the huge concurrency control information and time consuming for conflict detection. Hence, a concurrency control scheme for temporal database management systems(TDBMSs) should use small number of concurrency control information for conflict detection. However, when a coarse granularity for conflict detection is used in order to reduce the number of concurrency control information, the system throughput might be severely degraded since false conflicts make long-lived transaction wait or even restart unnecessarily. Hence, TDBMS has to have a concurrency control scheme which is able to efficiently detect actual conflicts not false conflicts while the number of concurrency control information is minimized. To our knowledge, however, there has been a little previous research on the management of temporal transactions([1]), while there have been several previous researches on temporal indexing methods for efficient management of the huge temporal database([2][3][4][5][6]). In this paper, as an efficient concurrency control scheme, we propose a conflict detection scheme named Two-Level Conflict Detection(TLCD) which is able to detect conflicts correctly and at the same time efficiently between concurrent temporal transactions.

This paper is organized as follows. In section 2, we point out some distinguishable characteristics of TDBSs from traditional database systems. In section 3, we describe the basic ideas of proposed scheme. Finally, we conclude this paper in section 4.

2. Temporal Database Systems

Since TDBSs are different from traditional database systems in several ways such as the volume of the database, the effects of data manipulation operations and the number of data items accessed by a transaction, a concurrency control scheme for TDBMSs must reflect the characteristics of TDBSs in order to efficiently coordinate concurrent execution of temporal transactions.

2.1 Distinguishable Characteristics of Temporal Database Systems

In TDBSs, the time concepts of *valid time* and *transaction time* are systemically supported by TDBMS to manage historical database. Valid time of data record is of

logical time concept which represents the time duration of real world clock when the meaning of a data item is valid in the real world. That is, the *start of valid time* of a data item stands for the time when the meaning of the data item begins to make sense, and the *end of valid time* stands for the time when the meaning of the data item does not make sense further in the real world modelled into database. Hence, the start and the end of the valid time of a data item can be given by users and can be modified by users. However, since *transaction time* of data record is managed by TDBMS itself and represents the time duration of database system clock when the data record is managed in temporal database as the meaningful data value, users cannot update the transaction time of a data record. When a data record is inserted into a relation, the *start of transaction time* of the data record is set to the insertion time of the system clock and the *end of transaction time* is set to the unspecified value(∞). The unspecified value means the current time of system clock. This pair of transaction time values of the data record means that the data record is the current version which is useful as the current version from the specified insertion time until just now. But when a data record is deleted from a relation, the end of transaction time of the data record is changed to the exact deletion time of the system clock from the unspecified value instead of actually deleting the data record from the relation. This data record becomes a new historical version based on transaction time. In TDBSs, since, instead of deleting records from database TDBMS keeps the deleted records as historical versions on the basis of transaction time, temporal database has more data records than traditional database.

We can represent a temporal database as a three-dimensional space (Figure 1). In TDBSs, user can use transaction time and valid time concepts in addition to data key in order to specify the set of data items accessed from temporal database. The transaction time can be used to select data versions which are managed by TDBMS as the meaningful data versions during the given transaction time. If the unspecified value is used as the transaction time value, it is used for specifying current versions (the shady rectangle in Figure 1). The value of the data key is used to distinguish data items by given key values from other objects in selected versions by the specified transaction time. And the valid time is used to select a data record that is for the given valid time of real world clock from the data set.

As in traditional database systems, a temporal transaction is the basic unit for users

to access database in TDBSs. A temporal transaction consists of one or more operations which manipulate temporal database. The types of data manipulation operations are the same as those of traditional database systems. However, the semantics of the operations for temporal database systems are different from those for traditional database systems due to the transaction time concept. That is, TDBMS maintains temporal database in append-only mode according to transaction time concept since no data record is actually deleted from temporal database [5].

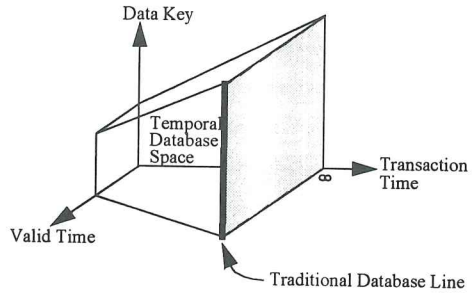


Figure1 Three-Dimensional Model of Temporal Database

In TDBSs, users can retrieve and update the data records which belong to current versions according to transaction time (i.e., the data records which are included in the shaded space in Figure 1). However, the data records which belong to historical versions on the basis of old time values of transaction time, that is, the data records whose end value of transaction time is a previous system clock not the unspecified value can be accessed in read-only mode. Hence, two or more temporal transaction can access the same data records of historical versions without any restriction for the consistency of temporal database. However, TDBMS should coordinate concurrent temporal transactions that are willing to read and update a data record, respectively, which belong to current versions of temporal database, since two or more temporal transactions can be in conflicts.

2.2 Related works

If two or more transactions concurrently access a database without any concurrency control, database might be in an inconsistent state([7]). In TDBSs, since current versions of temporal database can be concurrently read and written by two or more temporal transactions, a concurrency control scheme should coordinate concurrent temporal transactions to preserve the consistency of temporal database.

In TDBSs, however, since the number of data records accessed by a temporal transaction is larger than the number of data records accessed by a transaction in traditional database systems and, at the same time, some distinguishable characteristics of TDBS from traditional database system are available to efficiently coordinate concurrent temporal transactions, an efficient concurrency control scheme which is able to reduce the overhead for concurrency control is required for TDBMSs. However, there have been only a few related works on concurrency control for TDBMSs. To our knowledge, [1] is just one mentioning a concurrency control method for a temporal indexing mechanism. But it did not deal with the concurrency control of temporal transactions itself but dealt with only the concurrent management of the indexing mechanism named TSB-tree.

3. The Two-Level Conflict Detection Scheme

3.1 Conflicts between Temporal Transactions

Since as described in previous sections, temporal transactions can only read the data records of historical versions, no conflict can occur between two temporal transactions which will access data records of historical versions. However, since two temporal transactions may concurrently read and write, respectively, a data records of current versions conflicts may occur between concurrent temporal transactions which will access same data records of current versions of temporal database. Hence, a concurrency control scheme is required for TDBMS to preserve the consistency of current versions in TDBS.

The set of data items accessed by a read operation of a temporal transaction is represented as a three-dimensional sub-space of the temporal database space of Figure 1, since temporal queries usually specify the target data set by using not only data key and valid time but also transaction time as described in Figure 2 (a). Here after, DKS_i and DKE_i stand for the start value and the end value of specified data key in temporal transaction T_i, respectively. Similarly, VTS_i and VTE_i stand for the start time and the end time of specified valid time, respectively, and TTS_i and TTE_i stand for the start time

and the end time of specified transaction time in temporal transaction T_i, respectively. However, the set of data items modified by a write operation is represented as a two-dimensional sub-space of the temporal database space, since write operation does not use transaction time to specify data set as described in Figure 2 (b). We refer the sub-space composed by the specified values of data key, valid time and transaction time of the read operation of T_i to *read space* RS_i. On the other hand, we call the sub-space for write operation of T_j to *write space* WS_j.

```
Transaction Ti:
        range of t_v is relation_name
        retrieve t_v.all
                where t_v.key between DKSi and DKEi
                when begin of t_v equal VTSi and end of t_v equal VTEi
                as of TTSi through TTEi
```

(a) A read transaction

```
Transaction Ti:
        range of t_v is relation_name
        delete from t_v
                where t_v.key between DKSi and DKEi
                when begin of t_v equal VTSj and end of t_v equal VTEj
```

(b) A write transaction

Figure 2. Temporal Transactions in TQUEL

For two temporal transactions T_i and T_j of Figure 2 which concurrently read and write temporal database, respectively, two relationships between RS_i and WS_j can be established according to the values of transaction time, valid time, and data key of T_i and T_j. As the first case, no conflict space between RS_i and WS_j exists since T_i wants to read historical versions, i.e., the specified end value of transaction time (TTE_i) of the retrieve operation of T_i is less than the unspecified value(∞) which stands for the current time of system clock or T_i wants to read current versions of different data set represented by the values of data key and valid time of T_i from those values of data key and valid time specified by T_j. In this case, T_i and T_j can be concurrently executed. But, as depicted in Figure 3, if there is a conflict space CS_{ij} between RS_i and WS_j, there is a possibility of the existence of actual conflicts between T_i and T_j. Hence, detecting conflicts between T_i and T_j is required for this case.

If a record-based physical locking scheme in which data records are used as the granularity of locking is used, then it costs much to control the locks on all data records since a temporal transaction might access more data records than a traditional transaction. To minimize the cost for checking conflicts, a logical locking scheme such as the predicate locking scheme([8][9]) is useful in such circumstances. By using a predicate locking scheme for concurrency control of TDBMSs, a conflict is notified whenever an intersected conflict space CS_{ij} exists between the read space RS_i and the write space WS_j of temporal transactions T_i and T_j, respectively. However, if there is no actual data item in the conflict space CS_{ij}, T_i and T_j are not in actual conflict. It is a false conflict.

As an example, consider two temporal transactions T_i and T_j depicted in Figure 2 and the possible relationship between RS_i and WS_j depicted in Figure 3. If a predicate locking scheme is used for concurrency control and, at the same time, the conflict space CS_{ij} is not null, then T_i and T_j are in conflict. If a record-based physical locking scheme is used for concurrency control, all data records included in both RS_i and WS_j must be checked whether or not they are concurrently accessed. If any data item is accessed by both transactions, T_i and T_j are in conflict.

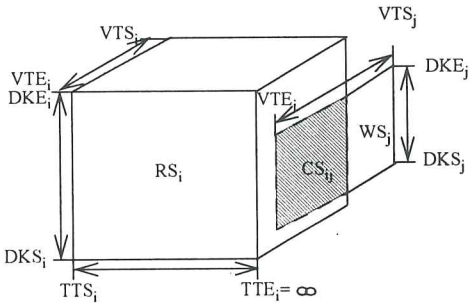


Figure 3 Conflict Space CS_{ij} between RS_i and WS_j

There is a trade-off between logical locking and physical locking as concurrency control scheme for TDBMS. If a logical locking scheme is used, there is the possibility of false conflicts while the cost for checking conflicts is maintained at a minimum level. On the other hand, if a physical locking scheme is used, the cost for checking conflicts is usually high while no false conflict occurs.

3.2 The Two-Level Conflict Detection Scheme

We propose an efficient conflict detection scheme for TDBMSs named Two-Level Conflict Detection(TLCD) scheme. By two-level, we mean that the information for conflict detection is maintained at two-levels: logical level and physical level and the conflict checking between concurrent transactions is done at these levels. The algorithm of the Two-Level Conflict Detection scheme is in Algorithm 1.

Algorithm 1: Two-Level Conflict Detection Algorithm
(* Conflict detection for an operation OP_i of T_i *)

```
    (* no conflict if Ti accesses historical versions *)
1.  IF TTEi:< ∞ THEN RETURN(NO_CONFLICT);
    (* otherwise, Ti accesses current versions in read mode or write mode *)
2.  IF OPi is read operation THEN BEGIN (* if read operation *)
3.      FOR each concurrent write transaction Tj DO BEGIN (* against write Tj *)
4.          CSij = INTERSECTION(RSi, WSj); (* compute conflict space *)
5.          IF CSij is not null THEN (* logical conflict is detected *)
6.              IF data records exist in CSij THEN (* physical conflict occurs*)
7.                  RETURN(CONFLICT with Tj);
8.              ELSE (* false conflict *)
9.                  CONTINUE; (* no actions against a false conflict *)
10.         END; (* IF logical conflict occurs *)
11.     END; (* FOR each concurrent transaction Tj *)
12. END (* IF read operation of Ti *)
13. ELSE IF OPi is write operation THEN BEGIN
14.     FOR each concurrent read or write transaction Tj DO BEGIN
15.         CSij = INTERSECTION(WSi, RSj); (* compute conflict space *)
16.         IF CSij is not null THEN (* logical conflict is detected *)
17.             IF data records exist in CSij THEN (* physical conflict *)
18.                 RETURN(CONFLICT with Tj);
19.             ELSE (* false conflict *)
20.                 CONTINUE; (* no actions against a false conflict *)
21.         END; (* IF logical conflict occurs *)
22.         CSij = INTERSECTION(WSi, WSj); (* compute conflict space *)
23.         IF CSij is not null THEN (* logical conflict is detected *)
24.             IF data records exist in CSij THEN (* physical conflict *)
25.                 RETURN(CONFLICT with Tj);
26.             ELSE (* false conflict *)
27.                 CONTINUE; (* no actions against a false conflict *)
28.         END; (* IF logical conflict occurs *)
29.     END; (* FOR each concurrent read or write transaction Tj *)
30. END; (* IF write operation of Ti *)
```

When temporal transaction T_i is going to access temporal database, the proposed

TLCD scheme check whether T_i is in a conflict with other concurrent transaction T_j. First of all, TLCD checks the end of transaction time (TTE_i) which is specified in the operation of T_i. If the operation has a smaller value than the unspecified value(∞) as TTE_i, i.e., T_i will access historical versions, T_i is not in any conflict with other concurrent transactions, since historical versions of temporal database can be accessed in read-only mode as described in section 2.1. Otherwise, i.e., T_i will access data records of current versions, TLCD scheme checks whether T_i is in conflict with other concurrent transactions on the basis of the type of operation of T_i. If the type of operation of T_i is of read type, TLCD scheme checks read-write conflicts against concurrent write transactions. On the other hand, if the type of operation is of write type, TLCD scheme checks both write-read and write-write conflicts against concurrent read transactions and write transactions, respectively.

The conflict checking is done as following. First, TLCD scheme computes the conflict space between RS_i and WS_j at line 4 of Algorithm 1 if the type of operation of T_i is of read type, or computes the conflict space between WS_i and RS_j at line 15 and WS_i and WS_j at line 22, if the type of operation is of write type. And then T_i must be checked against logical conflicts with other concurrent transaction T_j on the basis of the existence of T_i and T_j at line 5 or at line 16 and 23, respectively. If there is no conflict, that is, the conflict space CS_{ij} is null, T_i can access the data item. Otherwise, when a logical conflict is detected during logical conflict checking, physical conflict checking must be performed against all data records which are included in the conflict space CS_{ij} at line 6 or at line 17 and 24, respectively.

3.3 Performance of the Proposed Conflict Detection Scheme

Using the proposed algorithm, a temporal transaction which does not share a conflict space with other concurrent transactions can be examined with only the overhead for logical conflict checking based on the predicates of temporal transactions instead of the overhead for physical conflict checking based on all record identifiers accessed by temporal transactions. From the logical conflict checking, if a conflict is detected, the physical conflict checking is done against only data records which are included in the conflict space of logical conflict instead of all data records accessed by the conflicting transactions.

As an example, consider five transactions T_0, T_1, T_2, T_3 and T_4 of Figure 4. For simplicity, we choose a data set which has the same data key values and transaction time values but different valid time values. Hence the access spaces of transactions are represented as lines in Figure 5.

T_0: range of *t_v* is *r*
retrieve *t_v.all*
where *t_v.key* = k
when (not (t_v precede v_2) and (t_v precede v_0)

T_1: range of *t_v* is *r*
delete from *t_v*
where *t_v.key* = k
when (not (t_v precede v_4) and (t_v precede v_1)

T_2: range of *t_v* is *r*
retrieve *t_v.all*
where *t_v.key* = k
when (not (t_v precede v_5) and (t_v precede v_3)

T_3: range of *t_v* is *r*
delete from *t_v*
where *t_v.key* = k
when (not (t_v precede v_7) and (t_v precede v_6)

T_4: range of *t_v* is *r*
retrieve *t_v.all*
where *t_v.key* = k
when (not (t_v precede v_5) and (t_v precede v_3)
as of January 1, 1990 through December 31, 1990

Figure 4 Example of concurrent temporal transactions

First, consider T_4 with other concurrent transactions. Since T_4 reads a data set of historical versions of between January 1, 1990 and December 31, 1990 of the system clock while other concurrent transactions T_0, T_1, T_2 and T_3 access data sets of current versions which have the unspecified value as the end of transaction time, no conflict occurs between T_4 and other concurrent temporal transactions. Then, let us consider T_2 and T_3. Since there is no conflict space between T_2 and T_3, T_2 and T_3 can be executed concurrently after logical conflict checking with only the cost for logical conflict checking. However, since there are conflict spaces CS_{01} and CS_{12} between T_0 and T_1, and T_1 and T_2, respectively, physical conflict checking must be done against data records which are included in CS_{01} and CS_{12}, respectively. From the physical conflict

checking, since T_0 and T_1 have an actual conflict for data record 2, they cannot be executed concurrently. However, since there is no actual conflicted data record in CS_{12}, T_1 and T_2 can be executed concurrently. That is, the conflict of CS_{12} detected by logical conflict checking is a false conflict.

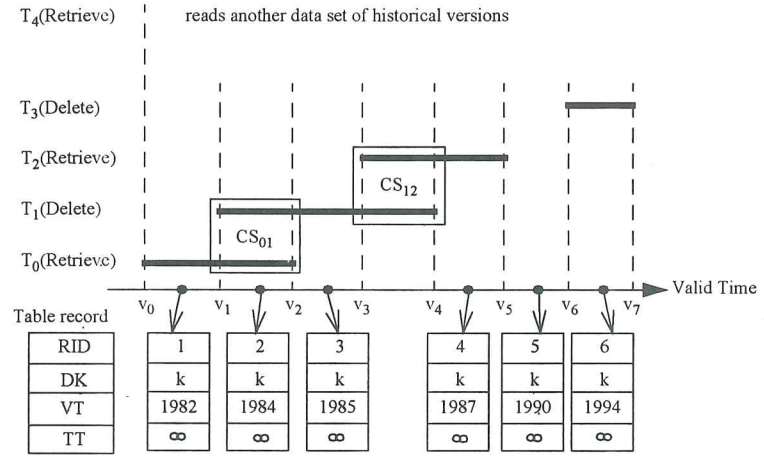


Figure 5 Temporal Transactions and a Temporal Relation

4. Conclusions

We proposed a two-level conflict detection scheme for concurrency control in temporal database systems. By two-level, we mean that the proposed scheme maintains the concurrency control information at two levels; logical level and physical level and detects conflicts at these two levels. That is, to reduce the overhead for conflict detection, first checking conflicts between concurrent temporal transactions is performed at the logical level based on their predicates. And if a conflict is detected from the logical level checking, in order to exactly differentiate actual conflict from false conflicts, the physical conflict checking is done on the basis of the record identifiers belonging to the intersected conflict spaces between the predicates. Hence, the proposed scheme is able to enlarge the number of transactions which are able to be executed concurrently while the overhead for conflict detection can be reduced.

Acknowledgment

This work was partially supported by the 1995 Grant of INHA University.

References

[1] D. Lomet and B. Salzberg, "Transaction-Time Databases", In [5], pp. 388-417, 1993.

[2] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Searching", *The Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 47-57, 1984.

[3] I. Ahn and R. Snodgrass, "Partitioned Storage for Temporal Databases Information Systems", *Information Systems*, Vol. 13, No. 4, pp. 369-391, 1988.

[4] R. Elmasri, G. T. J. Wuu, and V. Kouramajian, "The Time Index and the Monotonic B+-tree", In [5], pp. 433-456, 1993.

[5] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, *Temporal Databases*, The Benjamin/Cummings, 1993.

[6] S. H. Jeong, Y. T. Kim, G. H. Ryu, and Y. S. Kim, "The Design and Implementation of a Time Index Scheme Using Modified MBT(MMBT)", *The Proceedings of Workshop of SIGDB of Korea Information Science Society*, 1995.

[7] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[8] K. P. Eswaran, N. Gray, A. Lorie, and I. L. Traiger, "The Notions of Consistency and Predicate Locks in Database System", *Communication of ACM*, Vol. 19, No. 11, pp. 624-633, 1976.

[9] J. G. Ahn, Y. S. Lee, D. C. Shin, Y. S. Kim, J. T. Lim, W. Y. Kim, and S. Moon, "Development of a Prototypical Relational Database Management System: IM", *Journal of Korea Information Science Society*, Vol. 18, No. 4, pp. 399-412, 1991.

[10] H. Korth and A. Silberschatz, *Database System Concepts*, 2nd Edition, McGraw-Hill, 1991.

Multiple Query Optimization in Advanced Database Systems *

Zbyszko Krolikowski, Juliusz Jezierski

Institute of Computing Sciences, Poznan University of Technology
60-965 Poznan, POLAND
e-mail: krolikzb@pozn1v.put.poznan.edu.pl

Abstract

Conventional database management systems (*DBMS*) do not adequately meet the requirements of non-traditional, advanced applications, for example in data warehousing and on-line analytical processing (*OLAP*) systems. In order to support these applications, the functionality of conventional *DBMS* is being extended in several ways, between others by introducing some extensions to conventional query processing techniques. Traditional query optimization techniques are not sufficient, when used for queries in advanced database applications. In this paper we analyze multiquery optimization problem in advanced database systems (*DBS*). First we discuss variety of multiquery application scenarios for advanced *DBSs* and present workable example that shows opportunities for using of multiquery optimization approach in data warehousing and *OLAP* systems. Finally, we introduce two global alternative multiquery optimization methods that can be used in advanced *DBSs*. Results achieved from a simulation experiments show that these methods can be very useful in advanced database environment.

1. Introduction

The importance of databases in non-traditional applications is well understood and needs no explanation. Conventional database management systems do not adequately meet the requirements of non-traditional, advanced applications [2, 4]. In order to support these applications, the functionality of conventional *DBS* is being extended in several ways. At least four types of *DBSs* (termed *advanced DBSs* in the rest of this paper), which overlap in their functionality, can be identified readily [2]:

* [1] This research is supported in part by Polish State Committee for Scientific Research Grant 43-0329 (KBN2).

- *Data Warehousing and OLAP Systems* - to integrate decision support systems with relational databases,
- *Deductive Databases Systems* - to integrate large rule bases and their processing with relational database systems,
- *Active Database Systems* - to support situation monitoring, timing constraints, and host of related features,
- *Object-Oriented Database Systems* - to support new object classes, their operations, and complex objects in an extensible manner.

Several research prototypes that are being pursued actively, such as Maripose, Starburst, HiPAC and Postgres, can be viewed as advanced database systems as they fall into one or more of the categories described above. As many authors pointed out, efficient realization of these *DBSs* requires extensions to conventional query processing techniques [2, 4, 13].

Most of the current query processors optimize queries *one at a time* (such an approach is termed in the rest of this paper, *single query optimization - SQO*). However, significant performance improvements can be achieved by grouping several queries and optimizing the group as a whole [2, 4, 6, 8, 11-13, 15]. The idea behind such a way of query optimization (termed *multiple query optimization - MQO*) is to minimize the cost of evaluation of a set of related queries according to some criteria by evaluating them commonly. Characteristics of queries that can be exploited for minimizing the total cost of evaluation include [8, 11, 12, 15]: identical subexpressions or even subexpressions that subsume one another, common scan of relations, using cached results from intermediate computation, etc. Taking advantage of these common tasks, mainly by avoiding redundant data accesses, may prove to have a considerable effect on execution time.

The paper is organized as follows. In the next Section, the problem of multiple query optimization in advanced database systems is formulated. In Section 3 we discuss variety of multiquery application scenarios for advanced database systems. In Section 4 we present workable example that shows opportunities and benefits of using of *MQO* techniques in advanced database systems, namely, in data warehousing and *OLAP* systems. In Section 5 we present two global alternative multiquery optimization

methods that can be used in advanced *DBSs*. In the next Section we present experimental analysis of the effectiveness of the global optimization methods proposed in Section 5. Section 7 concludes this paper.

2. MQO Problem Statement

The execution cost of a relational query, e.g. time necessary to process the query, depends on the *query execution plan (QEP)*, i.e. on the particular sequence of relational operations that is selected to evaluate the query. For a given query there might be many different *QEP*s. The general goal of query optimization is to find *QEP* with the lowest cost. As in most of the literature we restrict our work to queries involving only join operations [14].

More formally, according to the previous assumption, a query Q is represented by a query graph $G_Q=(V_Q, E_Q)$, where the vertices $V_Q = \{ R_i \}$ represent base relations in the database, and each edge $\{ c_{ij} \} \in E_Q$ represent a join condition between the relations R_i and R_j.

The execution of a query Q can be expressed by a query execution plan P_Q which is a tree representing the order in which join operations are performed during the query evaluation process. The leaves of P_Q are the base relations in the query graph and the internal nodes represent join operations together with the specification of the way the joins are performed (e.g. join method, argument order, use of indices etc.). Any subtree of P_Q represents a partial execution of Q and will be called a *Partial Query Execution Plan (PQEP)*. As most systems implement the join operation as a 2-way join, we shall assume that P_Q is a binary tree.

Given a cost function *cost(P)* that associates an execution cost to a (partial) query execution plan P_Q, the *Single Query Optimization (SQO)* problem consists in selecting among all possible valid *QEP*s for a given query Q the one with the minimum cost [6].

Assume now that a set of queries $S=\{Q_1, ..., Q_m\}$ that should be processed together is given. We shall call it a *Multiple-Query Set (MQS)}*. In order to reduce the execution cost, the common parts of queries are outlined and executed only once in an integrated process. Therefore, the execution of a *MQS* may be represented by a *Multiple-Query Execution Plan (MQEP)* $M = \{C_1, ... ,C_k, R_1, ... ,R_m\}$, where $C_1, ... ,C_k$ are *PQEP*

representing the execution of common parts of queries, and $R_1, \ldots, +R_m$ are *PQEP* representing the residual parts of execution of the queries in the *MQS*. More precisely, each C_i and each R_i are *PQEP*s having as leaves either base relations or the results of the execution of some C_j. Therefore, in the *MQEP* a partial order is defined among the $C_1, \ldots, C_k$ and $R_1, \ldots, R_m$.

The cost function defined earlier for the *SQO* problem can be extended to the multiple-query environment to associate to each *MQEP* an execution cost corresponding to the sum of costs of all component *PQEPs*:

$$cost\ (M) = \sum_{i=1}^{k} Ci + \sum_{i=1}^{m} Ri$$

Given the cost function defined above, the *Multiple-Query Optimization (MQO)* problem can now be formulated as selecting among all possible *MQEPs* for a given multiple-query set *S* the one with the minimum cost [6].

3. MQO Application Scenarios

As we pointed above, significant performance improvements can be achieved in advanced database systems by grouping several queries and optimizing the group as a whole. In deed, the need for processing set of queries arises in natural way in many advanced applications. There are many applications where more than one query is submitted to the system in order to be executed. Below we identify a variety of application scenarios that require the capabilities of advanced *DBS*s and each case indicate opportunities for using *MQO* techniques [2].

First example is given by *deductive database systems*. A single query given to such a system may result in multiple queries that have to be run over the database [2]. This is because a deductive database may include more than one definition for the same predicate. For example, the following three rules [15]:

(R1) $A \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_l$

(R2) $A \leftarrow C_1 \wedge C_2 \wedge \ldots \wedge C_m$

(R3) $A \leftarrow D_1 \wedge D_2 \wedge \ldots \wedge D_n$

define the predicate *A*. A query on *A* would have to evaluate all three queries that correspond to the bodies of the above rules.

In *expert systems*, view definitions composed of union-compatible expressions or equivalently defined predicates are likely to have substantial overlap. In expert system, the matching phase of the recognize act cycle requires the issuing of the condition parts of rules as queries against the database. Evaluation of a query on such views (or predicates) can benefit from multiple query evaluation techniques by exploiting intra-query commonalties [2].

An essential characteristic of an *active database* [1] is the capability to monitor situations defined over the state of the database. Situations can be expressed using the data manipulation language, using rules, or using rules with matching patterns. Rules are of the form <*event, condition, action*>, where condition may reference data in the event and in the database. The set of all conditions of those rules sharing the same event, forms a potentially large set of predefined queries that have to be evaluated efficiently as an one, common unit [1, 2].

In *object-oriented database systems*, support for complex objects requires a capability for specifying the objects in the user's mental model and operations on the objects. Internally, a user object may be represented as sets of database view. Evaluation of these views is likely to generate overlapping queries, requiring *MQO* techniques [2].

In this paper, first of all we focused on a *data warehousing and on-line analytical processing* systems [3, 17] which are essential elements of decision support systems. In next section we present appropriate working example of using *MQO* technique in such environment.

4. MQO Optimization in Data Warehousing - working example

In the past three years data warehousing technologies have been successfully deployed in many industries: manufacturing (for order shipment and customer support), retail (for user profiling and inventory management), financial services (for claims analysis, risk analysis, and fraud detection), transportation (for fleet management), utilities (for power usage analysis), government (labour market and unemployment analysis) [3, 17]. Data warehousing and *OLAP* are essential elements of decision support systems. Data warehouse is a "subject-oriented, integrated, time-varying, non-

volatile collection of data that is used primarily in organizational decision making" [3]. Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports on-line analytical processing, the functional and performance requirements of which are quite different from those of the on-line transaction processing (*OLTP*) applications traditionally supported by the operational databases. Since data warehouses contain consolidated data, perhaps from several operational databases, over potentially long periods of time, they tend to be orders of magnitube larger than operational databases. Enterprise data warehouses are projected to be hundreds of gigabytes to terabytes in size. The workloads are query intensive with large, complex queries that can access millions of records and what is especially important, perform a lot of joins and aggregates [3, 17].

Thus, the volume and complexity of data as well as large set of join operations in data warehouse applications require extended and efficient query optimization methods. This is illustrated by the following example.

Example 4.1.

Let us consider a project of data warehouse and *OLAP* system, developed in Institute of Computing Sciences Poznań University of Technology by order of the Employment Agency. We assumed that data warehouse of the Employment Agency should contain following information:

- historical information concerning changes in personal data, addresses, education and professional qualification of unemployed, periods of unemployment for given person, unemployment benefits, etc.;
- statistical information concerning labour market in given region.

Appropriate fragment of the information model of the Employment Agency data warehouse is presented in Figure 4.1.

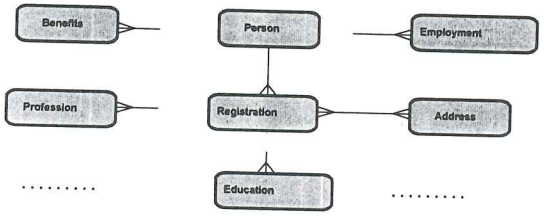


Figure 4.1. A piece of entity relationship model of the Employment Agency relational data warehouse.

Data stored in the warehouse are used to generate several statistical and analytical reports supporting decision making. For example, to generate reports about a how long on average, unemployed are taking benefits in relation to period of unemployment time, grouped by sex, age, education, etc., it is necessary to issue a following set of SQL queries.

select avg(sum(Benefits.to - Benefits.from)/sum(Registration.to-Registration.from)), Person.sex
from Benefits, Person, Registration
where Person.pesel=Benefits.pesel and Person.pesel=Registration.pesel *group by* Person.sex

select avg(sum(Benefits.to - Benefits.from)/sum(Registration.to-Registration.from)), Person.age
from Benefit, Person, Registration
where Person.pesel=Benefits.pesel and Person.pesel=Registration.pesel *group by* Person.age

select avg(sum(Benefits.to - Benefits.from)/sum(Registration.to-Registration.from)), Education.education_level
from Benefit, Person, Registration, Education
where Person.pesel=Benefits.pesel and Person.pesel=Registration.pesel and Education.pesel=Person.pesel *group by* Education.education_level

select avg(sum(Benefits.to - Benefits.from)/sum(Registration.to-Registration.from)), Adress.region
from Benefit, Person, Registration, Address
where Person.pesel=Benefits.pesel and Person.pesel=Registration.pesel and Adress.pesel=Registration.pesel *group by* Adress.region

Example of *MQEP* of the set of queries presented above is given in Figure 4.2.

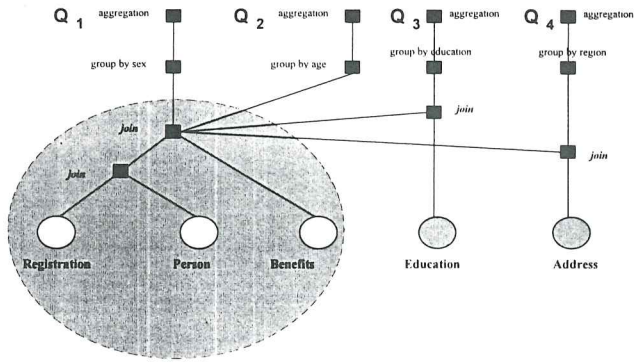


Figure 4.2. An example of the set of queries executed on data warehouse of the Employment Agency.

❑

As arise from Figure 4.2 queries Q_1 - Q_4 have common subexpressions. In our example there are following join operation: *Registration* ⋈ *Person* ⋈ *Benefits*. Therefore a significant performance improvement can be obtained by optimizing and executing the entire group of queries as whole, thus avoiding to duplicate the optimization and processing effort for common join operations.

5. Global Alternative MQO Optimization Methods

In this section, as a solution of the *MQO* problem in advanced database system, and especially in data warehouse systems, we propose two alternative global multiquery optimization methods [6].

Compared to the *SQO* problem, the major problem to be addressed in *MQO* is dealing with the large size of the search space of *MQEPs* that results from considering all possible overlappings among single query execution plans. Thus, it is necessary to introduce some simplifications to make the *MQO* problem tractable. Such simplifications we consider in the two alternative optimization methods presented below. The first one is based on the decomposition of the problem into a set of single query optimization problems; the second one is based on the integration of the query graphs and on solving the single query optimization problem for the integrated query graph.

The first approach that we shall call *Divide and Merge (D&M)* consists of the following steps:

- Solve separately the *SQO* problem for every query Q_i in the query set to get the optimal execution plan.
- Determine the common parts $C_1, \ldots C_k$ by intersecting all individual optimal query execution plans - *QEPs*.
- Build the multiquery execution plans - *MQEP* from $C_1, \ldots C_k$, and the residual parts $R_1, \ldots, R_m$ that are determined from the individual optimal *QEPs*.

The second approach that we shall call *Integrate and Split (I&S)* consists of the following steps:

- Merge the query graphs of all the queries Q_i to get a *global query graph* (that actually is a forest in a general case).

- Solve the *SQO* problem for the global query graph to get an optimal global *QEP*, which is a forest of trees with a separate root r_i for every query in the query set.
- Build the *MQEP* with $C_1, \ldots C_k$ being the subtrees in optimal global *QEP* common to more than one of the individual *QEPs*, and the residual parts $R_1, \ldots, R_m$ being the rest of the query trees.

The advantage of the *Divide and Merge* approach is clearly connected to the combinatorial structure of the search space, since it allows to reduce the large *MQO* problem to several *SQO* problems with considerably smaller search space sizes. The advantage of the *Integrate and Split* approach is due to the preliminary integration of the query graphs that allows to determine all the common subqueries before performing the optimization process and hence always leads to a set of execution plans where maximum overlapping is ensured. A further advantage is given, especially when the amount of overlapping among query graphs is large, by the possibility of concentrating the optimization effort on the common part. To illustrate both approaches let us consider the following example [6].

Example 5.1.

Consider a multiple query set $S = \{Q_1, Q_2\}$ with query graphs presented in Figure 5.1.

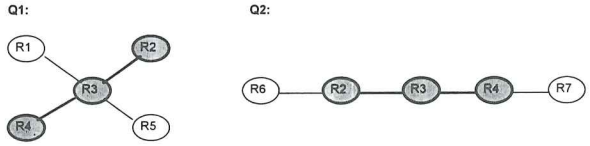


Figure 5.1. Query graphs of the set *S*.

Using to the *D&M* approach, both Q_1 and Q_2 are optimized separately. As the result, optimal query execution plans presented in Figure 5.2 are generated.

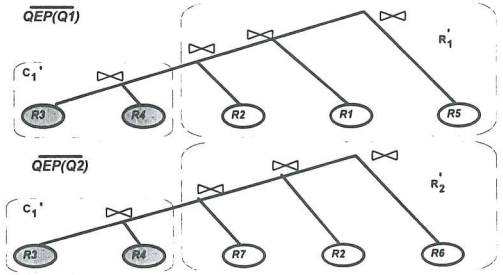


Figure 5.2. Optimal *QEPs* for Q_1 and Q_2.

The common part of both *QEPs* denoted C_1' contains the join of base relations R_3 and R_4. Residual parts of Q_1 and Q_2 are R_1' and R_2'. The *MQEP* for *S* produced by the *D&M* is given in Figure 5.3.

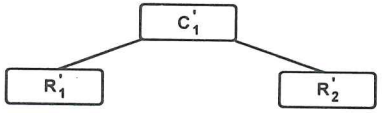


Figure 5.3. *MQEP* produced by the *D&M* approach.

Using to *I&S* approach, both Q_1 and Q_2 are optimized together. At the first step, the query graphs of the queries from *S* are merged to get a global query graph Γ (Figure 5.4).

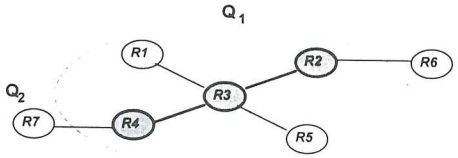


Figure 5.4. The integrated global query graph Γ.

The *SQO* problem for Γ is solved to get an optimal *QEP(Γ)* (Figure 5.5). The common part denoted C_1'' of *QEPs* for Q_1 and Q_2 consists now of joins of base relations R_2, R_3 and R_4. The *MQEP* for *S* produced by the *I&S* is given in Figure 5.6.

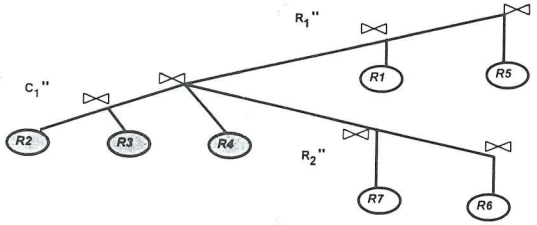


Figure 5.5. The optimal *QEP(Γ)*.

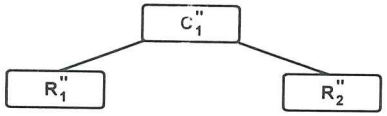


Figure 5.6. *MQEP* produced by *I&S* approach.

As we mentioned above, in comparison to the *SQO* problem the major problem to be addressed in *MQO* is dealing with the large size of the search space of multiquery execution plans (*MQEPs*) that results from considering all possible overlappings among single query execution plans. Additionally taking into account, that queries in non-traditional applications, like in data warehouse systems, deductive and active database systems can be very large, the fundamental problem with *MQO* is searching the large space of alternatives for finding the *MQEP* with the lowest cost. As the search space gets larger for large complex queries, the search strategy that investigates alternative *MQEP* is critical for the optimization cost [4].

In our opinion, a most appropriate approach to searching the large search space is to use random algorithms [5, 7, 9, 10, 16]. These algorithms permit to investigate the larger space of possible *MQEP* in comparison to other methods. Presented above global optimization methods are based on randomized search algorithms that ensure the tractability of the *MQO* problem in data warehouse systems and other new application environments. In *Divide&Merge* and *IntegrateSplit* multiquery optimization methods we adopted three combinatorial techniques: *Iterative Improvements* (*II*), *Simulated Annealing* (*SA*) and *Tabu Search* (*TS*).

6. Experimental Comparison of Global *MQO* Methods

In this section we will present analysis of the effectiveness of the global optimization methods presented above. An extensive parametric analysis has been performed by running a large set of experiments in a wide range of situations to understand how the global optimization methods and the combinatorial search algorithms are sensitive to the main parameters that characterize the *MQO* problem. In our simulation experiments we have applied the cost model commonly used in other papers [5, 10, 16]. Most interesting results of the experiments for "star" and "bushy" queries are presented in Figures 6.1 and 6.2.

The results of our experiments show the effectiveness of the global optimization methods and combinatorial algorithms to multiple-query optimization. All the algorithms we have tested behave quite well in a great variety of situations, but *Tabu*

Search as proved to outperforms *Iterative Improvements* and *Simulated Annealing* in most cases. As for the tradeoff between the two global optimization methods, our results show that it mainly depends on the amount of overlapping between queries, i.e. the fraction of each query that is common to all the queries in the set.

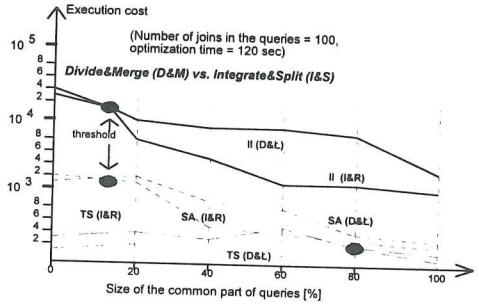


Fig. 6.1. Average execution cost versus size of the common parts for bushy queries

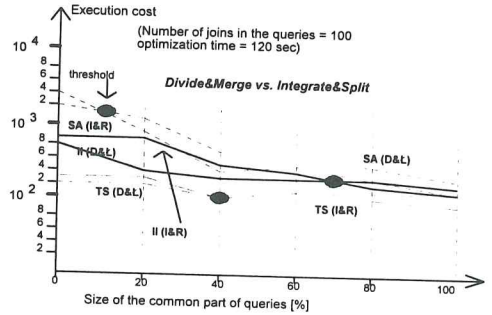


Fig. 6.2. Average execution cost versus size of the common parts for star queries

From the results of the experiments presented above, we can draw several interesting conclusions:

- The relative performance of the two global optimization strategies clearly depend on the *overlapping*, i.e. on the relative size of the common part between the queries in the query set. More precisely is possible to define, for every set of workload parameters, a threshold in the degree of overlapping, such that *Divide&Merge* performs better below the threshold and *Integrate&Split* above. This behavior can be explained since *Integrate&Split* exploits the overlapping by avoiding duplicating the optimization effort on the common part when this is large, and instead

Divide&Merge takes advantage in solving several problems of smaller size if they are largely independent.

- *Tabu Search* in a very large range of cases has a better performance. It shows a very stable and robust behavior, in the sense that it is fairly insensitive to variations of the workload parameters, and constantly approaches faster the solution.

7. Conclusions

In this paper we have studied multiple-query optimization problem in advanced database systems, especially in data warehouse and *OLAP* systems. Conventional query optimization methods are not sufficient, when used for queries in non-traditional database applications. First we have discussed a variety of multiquery application scenarios for advanced *DBSs* and have presented working example that showed benefits of using of *MQO* techniques in data warehousing and *OLAP* systems. Finally, we have introduced two global alternative multiquery optimization methods that could be used in advanced *DBSs*. In the paper, also results of the experimental analysis of the effectiveness of the global optimization methods were presented. Results achieved from our simulation experiments show that *Divide&Merge* and *Integrate &Split* methods can be very useful in advanced database environment.

References

1. Aiken A., et al, Behaviour of database production rules: termination, confluence, and observable determinism, in: Proc. of ACM-SIGMOD Int. Conf. on Manag. of Data' 92, 1992.
2. Chakravarthy S., Multiple Query Optimization: To Use or Not To Use, in: Proc. of the Intern. Workshop on Database Query Optimization, Portland, Oregon, 1989, pp.153 - 158.
3. Chaudhuri S., Dayal U., An overview of data warehousing and OLAP technology, in: Sigmod Record, 1997, pp. 65-75.
4. Graefe G., Research Problems in Database Query Optimization, in: Proc. of the Intern. Workshop on Database Query Optimization, Portland, Oregon, 1989.

5. Ioannidis E., Wong E., Query Optimization by Simulated Annealing, in: Proc. of ACM-SIGMOD Conference on Management of Data, 1987, pp. 9 - 22.
6. Królikowski Z., Matysiak M., Morzy M., Salza S., A Combinatorial approach to the multiple-query optimization problem, in: Proceedings of the Sixth Intern. Conf. On Management of Data - COMAD' 94, 1994, Bangalore, India (ed. McGraw - Hill Publishing Company).
7. Królikowski Z., Matysiak M., Multiple large query optimization using combinatorial methods, in: Proc. of the 9th Intern. Sympos. on Computer and Inform. Sciences ISCIS IX, Turkey, 1994.
8. Jarke M., Common subexpresion isolation in multiple query optimization, in: Query Processing in Database Systems, (ed: W. Kim, et al.), Springer-Verlag, Berlin, 1985, pp.191-206.
9. Matysiak M., Efficient optimization of Large Join Queries Using Tabu Search, in: Proc. of the Eighth Intern. Symp. on Computer and Inform. Sciences - ISCIS VIII, Istanbul, Turkey, 1993.
10. Morzy T., Salza S., Matysiak M., Tabu Search Optimization of Large Join Queries, in: Proc. of 4th Intern. Conf. EDBT'94, Cambridge, 1994, (ed. Springer Verlag), pp. 151-161.
11. Park J., Segev A., Using Common Subexpressions to Optimize Multiple Queries, in: Proc. of the 4th Intern. Conf. On Data Eng., Los Angeles, 1988, pp. 311 - 319.
12. Park J., Teorey T.J, Lafortune S., A knowledge based approach to multiple query processing, Data & Knowledge Engineering, vol. 3, No.4, 1989, pp. 261 -284.
13. Rosenthal A., Chakravarthy S., Anatomy of a Modular Multiple Query Optimizer, in: Proc. of the 14th VLDB Conf., Los Angeles, 1988, pp. 230 - 239.
14. Selinger A., Astrahan M., Chamberlin D., Lorie R.A., Price T.G., Access Path Selection in a Relational Database Management System, Proc. of ACM - SIGMOD, 1979, pp. 23-34.
15. Sellis T., Multiple-Query Optimization, ACM Trans. on Database Syst., 13(1), 1988, pp.23-52.
16. Swami, Gupta A., Optimization of Large Join Queries, Proc.of ACM-SIGMOD Conf.,1988, pp.8-17.

17. Widom J., Research problems in data warehousing, in: Proc. of 4th Int. Conf. on Inform. Knowledge Management, 1995.

Supporting Temporal Query Processing by Hash Filters

Holger Riedel *

Abstract

Hash filters are very useful for the optimization of query processing, especially in the case of join queries. In this paper, we investigate the possibilities how hash filters can be utilized in the field of temporal queries. We introduce a flexible kind of such hash filters and discuss different variations. Additionally, we analyze the usefulness of this concept for some well-known types of temporal queries.

1 Introduction

The processing of queries for temporal data significantly differs from well-known techniques in relational or object-oriented databases, because the time dimension has specific properties. For instance, most temporal queries can only inefficiently be handled, if the data is treated like relational data using operators of the relational algebra as query processing primitives.

A major problem is the representation of the history, if the whole temporal information of an object or tuple can consist of a union of different time intervals, like the employment information of part-time workers. This is supported by the TSQL2 proposal [13], but cannot be represented in a 1NF manner in a single tuple using start and stop times. Thus, either a specific "nested" representation or multiple "internal" tuples are necessary to represent a single "conceptual" tuple or object. This complicates the evaluation of queries (like testing predicates for selections or temporal joins) significantly.

Another problem is given by the specific temporal query types which must be supported. For instance, TSQL2 supports five different types of temporal conditions, as described in Figure 1. The TSQL2 data model is based on tuple timestamping. We call a chronon *active*, if it belongs to the history of the tuple or object.

Usually, the temporal data is stored in a similar way as relational data with specific extensions. As an example, Woo and Elmasri [18] propose a specific relational design where a specific attribute is used to mark the specific state of the tuple, which can be used to avoid updates of old tuples. Additionally, specific indices are proposed to support the query processing. An overview can be found in [10]. Often these temporal indices are based on spatial data structures [11] like the R-tree [3]. Although there are some similarities between the temporal and the spatial dimension

*adress: Fakultät für Mathematik und Informatik, Universität Konstanz, D-78457 Konstanz, Germany, email: Holger.Riedel@uni-konstanz.de

`valid(t1) = valid(t2)`	`t1` and `t2` have the same active chronons.
`valid(t1) OVERLAPS valid(t2)`	`t1` and `t2` have at least one active chronon in common.
`valid(t1) CONTAINS valid(t2)`	All active chronons of `t2` are also active for `t1`.
`valid(t1) PRECEDES valid(t2)`	`t1` is finished, before `t2` begins.
`valid(t1) MEETS valid(t2)`	The begin of `t2` is exactly one chronon after the end of `t1`.

Figure 1: Temporal conditions in TSQL2

(e.g. the mentioned condition predicates of TSQL2 also occur in spatial databases with a slightly different semantics), there is a major divergence that the data of a temporal object may change quite often or has indefinite length which can only inefficiently be incorporated into spatial data structures like the R-tree. A different index concept is the Time-Index [4], where all important instants (i.e., any instant where a "logical" update occurs) are held in a B-tree with references to the relevant database objects.

Another efficient method is hashing, which is widely explored for query processing, mostly for join queries [17, 1, 8]. Up to our knowledge, hashing for temporal query processing has only be discussed as a method for the efficient allocation of blocks within the query processing [2]. Nevertheless, the use of concepts like partitioned hashing [16] or encoding of the history similar to the the z-order encoding in spatial databases [7] seem to be useful to describe the histories of objects. Similar ideas of non-standard hashing are presented in the context of complex object databases in [15], where complex objects are encoded into hash values in order to simplify the equality test.

In our approach, temporal information of an object or a tuple is encoded in a hash value of a fixed length. The hash values can be used to check temporal predicates without accessing the primary data. Thus, these hash values can be used to implement query processing techniques which use hash values as filters to simplify the implementation of selections or joins. More specifically, the temporal hash value can be used in a flexible way:

- The basic granularity of the hash value can be different from the temporal granularity in the database. This is important, because the choice of the granularity in the conceptual model should not be interfered by physical design decisions like the parametrization of an index.
- Often, current data is more important than old data. This can be supported in our approach by using more bits for instants near to the referencing instant. Alternatively, time-intervals of the same size are supported, too.
- Because each bit of the hash value represents a certain time interval I, it has must be clarified whether a 1 in the hash value expresses the fact that the

object is valid at least for one or for all chronons of I. We support both alternatives.

In the next section, we describe the basic structure of the temporal hash values, and in Section 3, we give a detailed analysis how the condition predicates of TSQL2 can be evaluated using these concepts.

2 Temporal hash filters

2.1 The basic idea

We propose a storage structure for temporal data where the temporal information about an object or an attribute is encoded as a hash value. Each hash value is a representation of the complete history of the indexed attribute up to a certain *reference instant* r.

These hash values can be used in a hash-based query processing environment as reference values to check certain conditions. As an alternative, it may also be useful to generate physical addressses out of the hash values.

A temporal hash-index can be seen as a set of triples (o, h, r) where o is the referenced object or attribute value, h is the representation of the history of o, and r is the referencing instant. The hash values are variable in time, i.e. we store hash values with different reference instants in a single file and use them in parallel to process the queries. Therefore, it is necessary that such hash values can be compared using some shifting of a hash value to a different reference instant.

The detailed structure of the hash value h is as follows. Within the hash value, temporal information of a certain interval I is encoded into one bit, thus the temporal interval $(-\infty, r]$ is partitioned into $n = 2^m$ $(m = 1, 2, 3, \ldots)$ parts, each with a representing bit. In our approach, the hash values can be parametrized by the following dimensions:

- *linear* $\leftrightarrow$ *logarithmic:* In many time-oriented applications, the current data is more important than older parts of the database. Thus it may be useful to describe "data near to the referencing instant" by more bits than data of earlier times. We support two alternatives:
  - *linear:* All but the lowest bit represent a time interval of a fixed length, which is called the *granularity* of the index.
  - *logarithmic:* The most current index-granule is represented by $n/2$ bits, the next by $n/4$ bits, the next by $n/8$ bits and so on. The m-th index-granule is represented by one bit and all index-granules before are put together into one.
- *sparse* $\leftrightarrow$ *dense:* Because the granularity of the index is usually not the granularity of the represented data, a bit in the index usually refers to several instants. Thus, it has to be fixed whether a 1 represents that *at least one* or *all* instants are in the lifetime of the refered object. We support both alternatives where the first alternative is called *sparse*, and the second *dense*.

and	0	1	?
0	0	0	0
1	0	1	?
?	0	?	?

or	0	1	?
0	0	1	?
1	1	1	1
?	?	1	?

Figure 2: Three-valued and and or

Employees		
Employee	*employment*	*ssno*
Peter	[2/92 - 2/95; 5/95 - 9/96]	55446677
Paul	[2/92 - 12/94; 4/95-10/95]	55448867
Mary	[4/96 - 12/97]	55338967
Doris	[7/89 - 12/89; 7/90-12/90]	55338922

Figure 3: The relation *Employees*

It is important to crasp that the granularity of the database and the granularity of the hash values are two different concepts:

- the *database granularity* is a conceptual decision how temporal information is recorded, e.g., the employment time is recorded month by month.
- The *granularity of the hash values* fixes the distance between reference instants, and may be chosen upon physical database design decisions. Thus, it may be useful to use a representation year by year, where always the refencing instant is the 31st December.

In the following we use the notation *"*$\texttt{[11110010]}_{31.12.1996}$ *is an (8-bit,dense-log,1 year) hash value"* to describe that [11110001] is a hash value with referencing instant 31st December 1996, as a 8-bit hash value with granularity of one year, based on the dense and logarithmic dimensions. In order to simplify the notations of the examples below, we use $\texttt{[11110010]}_{1996}$ as a shorthand for $\texttt{[11110010]}_{31.12.1996}$. We use the abbreviations *lin* and *log* to refer to linear or logarithmic hash values. We enumerate the bits of a hash value from 1 to n, where the first bit is the *past-infinity bit*, while the bit n refers to the time interval directly before the referencing instant. Because the bits in the hash value directly correspond to the truth values *true* and *false*, we use the functions and and or to describe bitwise computations with the hash values. Later on, we need a three-valued approach. So we use and and or as three-valued operators as given in Figure 2.

Example 1 In order to describe the functionality of temporal hash values, we use a temporal relation *Employees* as given in Figure 3. Because we only use the attribute *employment* for temporal queries in the following examples, we do not add further temporal aspects to the example instance. Thus, the temporal information given through *employment* can be seen as the timestamp for the whole tuple in this example.

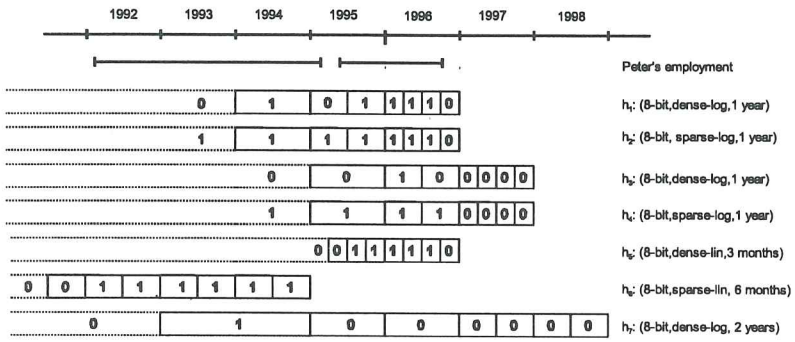


Figure 4: Representation of Peter's employment by 8-bit hash values.

In Figure 4 we use *Peter's* employment times as an example how temporal hash values can be built. In order to simplify the example we use 8 bits, although in practice bigger sizes are very reasonable.

h_1 is a (8-bit, dense-log, 1 year) hash value. Because it is logarithmic, the four highest bits are used for 1996, each for a quarter, 1995 is described in half-year terms and 1994 by a single bit. The complete history up to 1993 is represented by a 0, because there are instants in the past where Peter was not employed.

The only difference between h_1 and h_2 is that h_2 is sparse, thus the the third bit of h_2 is 1, because there are instants in the first half of 1995 where Peter was employed. h_3 and h_4 have the same dimensions as h_1 resp. h_2, but the referencing instant is 31st December 1997. Thus, the year 1995 is only represented by one bit in these cases. h_5 is a linear hash value, where each bit represents 3 months. Because the granularity is 3 months, the possible referencing instants are 31st March, 30th June, 30th September and 31st December. h_6 is a (8-bit, sparse-lin, 6 months) hash value referenced at the end of 1995. As the picture shows, h_6 is inappropiate to encode the history of Peter's employment, because its reference instant is not beyond the end of Peter's employment. h_7 is an example of an (8-bit, dense-log, 2 year) hash value. □

2.2 Temporal hash values and query processing

Temporal hash values used can be used to store information about temporal data either in an index or as adresses in hashed based storage environment.

A major reason for the invention of the temporal hash filters was that they can easily be compared with each other even when the referencing instants differ. Before we go into the details, we give an example to show the basic mechanism how queries can be evaluated.

Example 2 We like to check whether Paul was always employed when Peter was employed ($employment(Paul) \supseteq employment(Peter) \equiv Query\ 1$) or whether they have at least one month in common ($\equiv Query\ 2$). Using (8-bit, dense-log, 1 year) hash values, the hash values are $\texttt{[011110110]}_{1995}$ for Paul and $\texttt{[01011110]}_{1996}$ for

Peter. To use a bitwise comparison, Paul's hash value is shifted to 1996 by setting the four bits of 1996 to 0, and evaluating the other bits using the given hash value:

1993 and earlier:	0 and 1	⇒	0
1994:	1 and 1	⇒	1
1/1995:	0 and 1	⇒	0
2/1995:	1 and 0	⇒	0

The resulting hash value is $\texttt{[01000000]}_{1996}$. This can be compared with Peter's hash value $\texttt{[01011110]}_{1996}$ bit-by-bit. Bit 2 shows that both are employed for the whole year 1994 (⇒ $Query\ 2 = \texttt{true}$) and Peter is completely employed in the first three quarters of 1996, while Paul is not (⇒ $Query\ 1 = \texttt{false}$). □

The important point of the example is that it is necessary to compare index values of different referencing instants. This can be done by shifting one hash value to the other. Because we have a fixed granularity of the referencing instants this works well even in the logarithmic dimension.

The details about shifting hash values are explained in the next section, while the evaluation schema of different query types are discussed in Section 3.

2.3 Shifting hash values

Each hash value $hv_{old} = (o, h_{old}, r_{old})$ can be shifted to the next allowed referencing instant $r_{old} + 1$. All bits refering to instants between r_{old} and $r_{old} + 1$ are set to 0, and the others are newly calculated.

Definition 1 (Shifting of a hash value) Let $hv_{old} = (o, h_{old}, r_{old})$ be a hash value of length n.

hv_{old} can be shifted to $hv_{new} = (o, h_{new}, r_{old} + 1)$, where h_{new} is given by

- *in the linear case:*

$$h_{new}(j) = \begin{cases} 0 & : \ j = n \\ h_{old}(j+1) & : \ j \in \{2, \ldots, n-1\} \\ h_{old}(1) \text{ and } h_{old}(2) & : \ j = 1 \text{ and } h_{old} \text{ is dense} \\ h_{old}(1) \text{ or } h_{old}(2) & : \ j = 1 \text{ and } h_{old} \text{ is sparse} \end{cases}$$

- *in the logarithmic case:*

$$h_{new}(j) = \begin{cases} 0 & : \ j \in \{n/2+1, \ldots, n\} \\ h_{old}(2j-1) \text{ and } h_{old}(2j) & : \ j \in \{1, \ldots, n/2\} \text{ and } h_{old} \text{ is dense} \\ h_{old}(2j-1) \text{ or } h_{old}(2j) & : \ j \in \{1, \ldots, n/2\} \text{ and } h_{old} \text{ is sparse} \end{cases}$$

◇

Example 3 As explained in Example 1, h_3 is a shift of h_1 by one index-granule, in this case from 1996 to 1997. □

Shifting is always directed into the future, because shifting into the past is not possible:

- A basic assumption about a hash value is that it encodes the complete history. Thus, moving the reference instant backwards leads to some information loss, because then the new hash value does not describe some parts of the history which was present before, especially when some 1s get lost.

- It would be necessary to split an interval into two intervals (using linear hash values, this only occurs for the past-infinity bit). It is not clear how the new hash value should be built then. For instance, shifting h_3 from 1997 to 1996, we cannot obtain h_1, because $h_3(1), h_3(2)$, and $h_3(4)$ cannot be split, because a 0 in a dense hash value gives no information whether some of the referenced parts are completely active or not.

3 Query Processing with hash filters

3.1 Temporal queries

In the literature, there are many proposals for temporal query languages [9, 12] with a rather wide range of supported query types. Because we are mainly interested in index support for temporal queries, we focus on the specific temporal conditions which are present in temporal selections and temporal joins. More precisely, we choose the five condition types of TSQL2 as introduced in the Introduction, because they cover a broad range of temporal queries.

These condition predicates can be evaluated using the temporal hash filters by the following steps:

- express the condition on the level of hash values. Shift the hash value to a common referencing instant, if necessary.

- Check the specific condition. The result is usually three-valued:
  1 : the condition is fulfilled.
  0 : the condition is not fulfilled.
  ? : the condition cannot be decided using the hash values.

This can be used to realize usual hash-based query processing strategies with a probe and a test phase as discussed in Example 2. It turns out that =, `OVERLAPS`, and `CONTAINS` can be solved by the following strategy:

- compare the hash values bitwise. The bit operation depends on the query type and the index type and is three-valued:
  1 : this part of the history fulfills the condition.
  0 : this part of the history does not fulfill the condition.
  ? : the condition cannot be decided for this part of the history using the hash values. The details about this step are given in Figure 5.

- combine the bitwise results by a three-valued operator (and for = and `contains`, or for `overlaps` as given in Figure 2) and interprete the three-valued result as mentioned before.

dense: =	0	1
0	?	0
1	0	1

sparse: =	0	1
0	1	0
1	0	?

dense: OVERLAPS	0	1
0	?	?
1	?	1

sparse: OVERLAPS	0	1
0	0	0
1	0	?

dense: CONTAINS	0	1
0	?	0
1	1	1

sparse: CONTAINS	0	1
0	1	0
1	1	?

Figure 5: Bitwise operations on the hash values for temporal comparisons

(8-bit,dense-log,1 year)	
object	*hash value*
Peter	$[01011110]_{1996}$
Paul	$[01110110]_{1995}$
Mary	$[00011111]_{1997}$
Doris	$[00010011]_{1990}$

(8-bit,sparse-log,1 year)	
object	*hash value*
Peter	$[11111110]_{1996}$
Paul	$[11110110]_{1995}$
Mary	$[00111111]_{1997}$
Doris	$[00010011]_{1990}$

Figure 6: The hash values for employment

Example 4 Figure 6 shows an encoding of the employment information as given in Figure 3 into dense and sparse hash values. We are interested in the following query:

```
select e2.* from Employees e1, Employees e2
where e1.name='Peter' and valid(e2) CONTAINS valid(e1)
```

which selects all employees who were at least employed when Peter was employed. Figure 7 shows the steps of the evaluation for each object. At first, the hash values are shifted. Because the referencing instant of o_3 is beyond the referencing instant of o_1, we have to shift o_1 in this case. The fourth column shows the bit-wise comparison results and the fifth column the result. In this example, dense and sparse hash values have the same selectivity. An alternative is a kind of preprocessing where all values are shifted to the highest reference instant which is relevant for the specific query. This scenario is described in Figure 8.

A similar evaluation schema for the query

```
select e2.* from Employee e1, Employee e2
where e1.name='Peter' and valid(e2) OVERLAPS valid(e1)
```

which selects all employees who have at least one common chronon with Peter, is given in Figure 9. Here the selectivity of the dense index is better, because two objects are selected and the other two have to be checked, while the sparse index is no help for this query. □

using dense hash values				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (and)
Peter	[01011110]$_{1996}$	[01011110]$_{1996}$	[?1?1111?]	?
Paul	[01000000]$_{1996}$	[01011110]$_{1996}$	[?1?0000?]	0
Mary	[00011111]$_{1997}$	[00100000]$_{1997}$	[??011111]	0
Doris	[00000000]$_{1996}$	[01011110]$_{1996}$	[?0?0000?]	0

using sparse hash values				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (and)
Peter	[11111110]$_{1996}$	[11111110]$_{1996}$	[????????1]	?
Paul	[11110000]$_{1996}$	[11111110]$_{1996}$	[????0001]	0
Mary	[00111111]$_{1997}$	[11110000]$_{1997}$	[00??1111]	0
Doris	[10000000]$_{1996}$	[11111110]$_{1996}$	[?000000?]	0

Figure 7: Evaluation of the CONTAINS query

using dense hash values: shifting to 1997				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (and)
Peter	[00100000]$_{1997}$	[00100000]$_{1997}$	[????????]	?
Paul	[00000000]$_{1997}$	[00100000]$_{1997}$	[??0?????]	0
Mary	[00011111]$_{1997}$	[00100000]$_{1997}$	[??011111]	0
Doris	[00000000]$_{1997}$	[00100000]$_{1997}$	[??0?????]	0

using sparse hash values: shifting to 1997				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (and)
Peter	[11110000]$_{1997}$	[11110000]$_{1997}$	[????1111]	?
Paul	[11000000]$_{1997}$	[11110000]$_{1997}$	[??001111]	0
Mary	[00111111]$_{1997}$	[11110000]$_{1997}$	[00111111]	0
Doris	[10000000]$_{1997}$	[11110000]$_{1997}$	[?0001111]	0

Figure 8: An alternative evaluation of the CONTAINS query

using dense hash values				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (or)
Peter	[01011110]$_{1996}$	[01011110]$_{1996}$	[?1?1111?]	1
Paul	[01000000]$_{1996}$	[01011110]$_{1996}$	[?1??????]	1
Mary	[00011111]$_{1997}$	[00100000]$_{1997}$	[????????]	?
Doris	[00000000]$_{1996}$	[01011110]$_{1996}$	[????????]	?

using sparse hash values				
object	*hash value*	*Peter's hash value*	*comparison*	*result* (or)
Peter	[11111110]$_{1996}$	[11111110]$_{1996}$	[???????0]	?
Paul	[11110000]$_{1996}$	[11111110]$_{1996}$	[????0000]	?
Mary	[00111111]$_{1997}$	[11110000]$_{1997}$	[00??0000]	?
Doris	[10000000]$_{1996}$	[11111110]$_{1996}$	[?0000000]	?

Figure 9: Evaluation of the OVERLAPS query

A different evaluation schema is necessary to evaluate PRECEDES and MEETS predicates, because these operators are based on the order of the instants.

We introduce the following notions for a hash value h:

- $hbit_1(h)$ is the highest bit of h set to 1.
- $lbit_1(h)$ is the lowest bit of h set to 0.

Now the predicates PRECEDES and MEETS can be decided using the hash values as summarized in Figure 10. Depending on the *query type* and the type of the hash value, a *test condition* has to be checked, and if it succeeds, the mentioned *action* has to be performed with the same semantics as before (1: the condition is fulfilled; 0: the condition is not fulfilled; ?: the condition cannot be decided using the hash values). Usually, the result is two-valued for each query type, only PRECEDES queries used with sparse hash values can result in any of the three possible results. The last column shows the result, if X or Y have no 1-bits.

Example 5 Figure 11 shows the result of the four conditions

(1) `valid('Peter') MEETS valid(E)`
(2) `valid(E) MEETS valid('Peter')`
(3) `valid('Peter') PRECEDES valid(E)`
(4) `valid(E) PRECEDES valid('Peter')`

according to the given evaluation schema. □

As the examples show, the usefulness of the temporal hash values depends on different circumstances:

- The distribution of data over time: a wide-spread distribution of the data (many "0" in the hash values) yields better results. In our example, the data distribution is rather dense, i.e. there is a high degree of overlapping time intervals in the database. This significantly reduces the selectivity of the hash filters.

query type	*index type*	*test condition*	*action*	0?
X PRECEDES Y	dense	$hbit_1(X) \geq lbit_1(Y)$	0	?
X PRECEDES Y	dense	$hbit_1(X) < lbit_1(Y)$	?	?
X PRECEDES Y	sparse	$hbit_1(X) > lbit_1(Y)$	0	0
X PRECEDES Y	sparse	$hbit_1(X) = lbit_1(Y)$	?	0
X PRECEDES Y	sparse	$hbit_1(X) < lbit_1(Y)$	1	0
X MEETS Y	dense	$hbit_1(X) \geq lbit_1(Y)$	0	?
X MEETS Y	dense	$hbit_1(X) < lbit_1(Y)$	?	?
X MEETS Y	sparse	$hbit_1(X) \in \{lbit_1(Y)-1, lbit_1(Y)\}$	?	0
X MEETS Y	sparse	$hbit_1(X) \notin \{lbit_1(Y)-1, lbit_1(Y)\}$	0	0

Figure 10: Evaluation schema for PRECEDES and MEETS

using dense hash values						
E	*hash value of* E	*Peter's hash value*	(1)	(2)	(3)	(4)
Peter	$[01011110]_{1996}$	$[01011110]_{1996}$	0	0	0	0
Paul	$[01000000]_{1996}$	$[01011110]_{1996}$	0	0	0	0
Mary	$[00011111]_{1997}$	$[00100000]_{1997}$	?	0	?	0
Doris	$[00000000]_{1996}$	$[01011110]_{1996}$	?	?	?	?

using sparse hash values						
E	*hash value of* E	*Peter's hash value*	(1)	(2)	(3)	(4)
Peter	$[11111110]_{1996}$	$[11111110]_{1996}$	0	0	0	0
Paul	$[11110000]_{1996}$	$[11111110]_{1996}$	0	0	0	0
Mary	$[00111111]_{1997}$	$[11110000]_{1997}$	0	0	0	0
Doris	$[10000000]_{1996}$	$[11111110]_{1996}$	0	?	0	?

Figure 11: Evaluation of the MEETS and PRECEDES queries

- As usual for hash values, temporal hash filters simplify the exactness of the temporal data, which complicates the query processing of some queries. For instance, there can be "invisible" active chronons in "0"-intervals of dense hash values, which restricts the usefulness of dense indices for CONTAINS queries. In an sparse index, a "1" only signals one active chronon. Thus, finding two objects with the same sparse hash value does not help much for the evaluation of OVERLAPS queries, where the bitwise results are combined by or.

- The shifting of hash values leads to many 0s in the index, especially in the logarithmic dimension. This necessarily reduces the usefulness for some query types.

- the past-infinity bit behaves usually different than the others, because in many databases it will be 0 for dense indices and 1 for sparse indices, especially after shifting.

At last, we remark that the benefit of the temporal hash filters is higher than it may seem from some of the given examples because:

- in reality the length of the hash values will be between 32-128 bit, thus, the temporal information is more exactly described by the hash value and the past-infinity bit is not that important.

- in the paper we only discussed the details for the logarithmic dimension. In the linear dimension, usually bigger time intervals are supported and shifting only leads to a linear number of additional 0s, which extends the expressiveness of the hash values.

4 Conclusion

In this paper we presented a technique how temporal information can be encoded in hash values. We gave several alternatives and analyzed its usefulness for several temporal query types. It turns out that the proposed hash values allow a very flexible description of temporal data, because the representation within the hash value is independent of the granularity of the temporal data in the database. Thus, an index based on such hash values can be optimized w.r.t. the data load and the temporal queries which have to be supported.

The presented framework leaves many ways of future work. At first, it seems interesting to analyze how further kinds of temporal queries can efficiently be supported by the proposed hash values. It has to be worked out how hash values of different granularities can be processed simultaneously. This is necessary when we support optimization of queries with indices of different granularities and dimensions. It seems that this can be tackled by allowing only granularities with fixed fractions of a "global" granularity, similar to the structure of log-structured hash values, but here many details are open.

Another open question bothers how temporal information with null values should be treated. In our approach nulls occured within the scan process of index values and indicated that a certain condition could not be checked by the index, which enforces a lookup to the primary data. Adding nulls to the temporal information and to the temporal hash values may lead to some difficulties, because now nulls are used with two different semantics.

Another part of our future work is the implementation of the proposed index structure in our extensible OODBMS CROQUE [5], where we add temporal support on top of an ODMG-like OO model and queries.

References

[1] J.A. Blakeley and N.L. Martin. Join index, materialized view, and hybrid-hash join: A performance analysis. In *Proc. of the IEEE Intl. Conf. on Data Engineering*, pages 256–263, 1990.

[2] K. Bratbergsengen and K. Norvag. Improved and optimized partioning techniques in database query processing. In *British National Conference on Databases (BNCOD)*, pages 69–83, 1997.

[3] T. Brinkhoff, H. Kriegel, R. Schneider, and B. Seeger. Efficient processing of spatial joins using r-trees. In *Proc. ACM SIGMOD Conference on Management of Data*, pages 237–246, June 1993.

[4] R. Elmasri, G.T.J. Wuu, and V. Kouramajian. The Time Index and the Monotonic B$^+$-tree. *[14]*, pages 433–456, 1993.

[5] T. Grust, J. Kröger, D. Gluche, A. Heuer, and M. H. Scholl. Query evaluation in CROQUE – calculus and algebra coincide. In *British National Conference on Databases (BNCOD)*, pages 84–100, 1997.

[6] W. Kim, editor. *Modern Database Systems.* ACM Press, 1995.

[7] D. Lomet. A review of recent work on multi-attribute access methods. *ACM SIGMOD Record*, 21(4):56–63, 1992.

[8] R. Marek and E. Rahm. Tid hash joins. In *Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 42–49, 1994.

[9] G. Ozsoyoglu and R. Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):410–417, 1995.

[10] B. Salzberg and V. Tsotras. A comparison of access methods for time evolving data. Technical report, Northeastern University, 1994.

[11] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[12] R. Snodgrass. Temporal object-oriented databases: A critical comparison. In *[6]*, pages 386–408, 1995.

[13] R. Snodgrass, editor. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[14] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.

[15] V. Turau and H. Duchene. Equality testing for complex objects based on hashing. *Data and Knowledge Engineering*, 10:101–111, 1993.

[16] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Rockville, MD, 1988.

[17] P. Valduriez. Join indices. *ACM Transactions on Database Systems*, 12(2):218–246, 1987.

[18] J. Won and R. Elmasri. Representing retroactive and proactive versions in bi-temporal databases (2tdb). In *Proc. of the IEEE Intl. Conf. on Data Engineering*, pages 85–94, 1996.

On Effectiveness of Database Accessing Methods for Subset Searching

Maciej Zakrzewicz

Institute of Computing Science

Poznan University of Technology

ul. Piotrowo 3a, 60-965 Poznan, Poland

Tel.: (+48) 61 878 23 78

Fax: (+48) 61 877 15 25

mzakrz@cs.put.poznan.pl

Abstract

The field of relational database research has developed many database accessing methods for effective data retrieval, e.g. B$^+$ tree indexing, Bitmap indexing, hash-based join, sort-merge join. These methods are oriented on finding or joining single items (represented by records) that satisfy point or range conditions. However, in the area of data mining research, there is often the need to effectively retrieve the multi-item sets that contain a given multi-item subset. We will refer to this type of retrieval as Subset Search Problem.

In this paper we formally define the Subset Search Problem. We analyze and experimentally verify the usefulness and effectiveness of the most common database accessing methods applied to the Subset Search Problem. The results show that Bitmap indexing gives the best improvement to the implementation of the Subset Search Problem.

1. Introduction

Data Mining is a new area of database research that aims at finding previously unknown and potentially useful patterns in large databases [4]. The patterns are called knowledge and are usually represented by means of *knowledge rules*. There are many types of knowledge rules: classification, characteristic, discriminant, association etc., which are employed by different data mining tasks [11]. The most commonly sought patterns are *association rules* [1]. Intuitively, an association rule identifies a frequently occurring pattern of information in a database. Formally, by an association rule we mean a formula of the form $X \rightarrow Y$, where X and Y are two sets of items.

Association rules are discovered from database tables that store sets of items. Consider a supermarket database where the set of items purchased by a customer on a single visit to a store is recorded as a *transaction*. The supermarket managers might be interested in finding *associations* among the items purchased together in one transaction. An example of a supermarket database and the set of association rules derived from the database are presented in Figure 1. The example discovered rule: `bread ∧ butter ∧ milk → apples` states that a customer who purchases bread, butter and milk, probably also purchases apples. We refer to the left hand side of the rule as *body*, and to the right hand side as *head*. We also say, that the rule is *satisfied* by a given item set if $X \cup Y$ is contained in the set. We say, that the rule is *violated* by a given item set if the set contains X, but does not contain Y. Each rule has two measures of its statistical importance and strength: *support* and *confidence*. The support of the rule equals to the number of item sets that satisfy the rule divided by the number of all item sets. The rule confidence equals to the number of item sets that satisfy the rule divided by the number of item sets that contain X.

The main application area of association rules discovery is *basket analysis*. The basket analysis consists in finding the dependencies between products bought by customers in a supermarket. The results of the basket analysis can help in marketing, pricing, inventory planning or shelf planning to increase transactions and profit. For example, the observation that when customers buy milk, they also buy corn flakes, may lead to the price promotion of milk to increase the sale of corn flakes. Another popular data mining application is deviation detection that identifies deviations from established statistical norms in order to find suspicious data that may be indicative of fraudulent activity. Deviation detection consists mostly in finding the item sets that violate given set of rules.

Transaction_id	items
1	bread, butter
2	bread, butter, milk, apples
3	bread, butter, milk, apples

bread → butter

bread ∧ butter ∧ milk → apples

Figure 1. Example of a database and discovered rules

Usually, the discovered rules are stored in a database for future retrieval by users or decision support systems. Having stored the rules, users may search and analyze them to find more specific rules e.g. rules that relate bread and milk. Users may iteratively

penetrate the set of rules discovered from the given database from many points of view. Moreover, the users might be interested in finding customer transactions that e.g. violate the rule about bread and milk association.

In this paper we consider the problem of retrieval of item sets stored in a relational database, which we refer to as the Subset Search Problem. We propose a data structure for the item sets storage in a database and we give examples of practical applications of such structure in data mining area. Then, we analyze and experimentally compare the performance of the most popular database accessing methods applied to the Subset Search Problem.

1.1 Related Work

Database researchers have developed many database accessing methods for effective data retrieval. These methods can be classified into two groups: 1. without use of any additional permanent data structures (sort-merge join, hash-based join, nested-loops join) and 2. using additional permanent data structures (B^+ tree indexing, Bitmap indexing etc.).

The most popular methods of performing database joins on large database tables without use of indexes are sort-merge join and hash-based join. Both methods are based on full table scanning. To improve the performance of database join processing, sort-merge join method first sorts the joined tables and then performs the join [6]. The hash-based join method first builds a hash table for one of the joined database tables, and then performs the join [12].

Database indexes provided today by most database systems are B^+ tree indexes to retrieve records of a table with specified values involving one or more columns [3]. Another type of index that received recently significant attention and was implemented by most of commercial DBMSs is Bitmap index. Bitmap indexes were first developed for database use in the Model 204 product from Computer Corporation of America [9].

1.2 Outline

The paper is organized as follows. In Section 2 we introduce the Subset Search Problem, consider the data structures for item sets storage and explain the implementation of the Subset Search Problem in a relational, SQL-accessed database. In Section 3 we

describe the application of the popular database accessing methods to the Subset Search Problem. Section 4 contains our experimental results on synthetic data. Section 5 contains conclusions.

2. The Subset Search Problem

In this Section we define and analyze a generic structure for item sets storage in relational databases. Next, we introduce the Subset Search Problem and explain its implementation in a relational database.

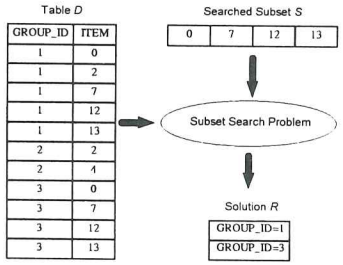


Figure 2: Example of the Subset Search Problem

We are given a large *database table of item sets D* in which each record *T* has two attributes: *GROUP_ID* and *ITEM*. The attribute *ITEM* represents an item and the attribute *GROUP_ID* represents a set that contains the item. For the sake of simplicity, we will alternatively refer to the attribute *GROUP_ID* as the *item set identifier*. In other words, each item set consists of records with the same value of *GROUP_ID* attribute. Let n_{trans} denote the *number of item sets* and n_{items} denote the *number of items*. We assume that both attributes are integer numbers, $GROUP\_ID \in \langle 0, n_{trans} \rangle$ and $ITEM \in \langle 0, n_{items} \rangle$. We are also given a *searched subset of items S* which is a collection of the form $\{item_1, item_2, ..., item_n\}$, where $item_i$ is an integer number, $item_i \in \langle 0, n_{items} \rangle$ and n is referred to as *searched subset size*.

Definition 1

The *Subset Search Problem* is to find in the database table *D* all item sets that contain all items from the subset *S*. The solution of the Problem consists of the set *R* of integer values *v* representing the identifiers of those item sets that contain the subset *S*:

$$R = \left\{ v: \forall_{i=1..n} \exists_{T \in D} S.item_i = T.ITEM \wedge v = T.GROUP\_ID \right\}$$

Let us consider the following example of the Subset Search Problem. Figure 2 presents the database table *D* of three item sets. Let the searched subset of items *S*={0, 7, 12, 13}. The solution of this Subset Search Problem will be *R*={1, 3} because only the item sets identified by the attribute *GROUP_ID*=1 and 3 contain all items from the set *S*.

Table *SHOPPING*

TRANSACTION_ID	ITEM
1	bread
1	butter
2	bread
2	butter
2	milk
2	apples
3	bread
3	butter
3	milk
3	apples

Figure 3: Example storage structure for purchase transactions

The Subset Search Problem can be found in several data mining tasks. Let us first show some examples of database tables of item sets. The model of item set organization presented in Figure 2 is commonly used for storage of both supermarket purchase transactions and association rules. Figure 3 illustrates an example database table *SHOPPING* that stores supermarket purchase data. The attribute *TRANSACTION_ID* identifies a purchase transaction and the attribute *PRODUCT* refers to a product purchased in the transaction. Figure 4 illustrates a data model and its example relational representation for association rules storage. Rule bodies and heads are placed in one database table, while the second table keeps specific rule parameters (e.g. support and confidence values). In this example, two rules from the Figure 1 are represented. Each item of the rule body or head is stored as a separate record in the table *ELEMENTS*. The attribute *RULE_ID* groups the rule items into rules and the attribute *ITEM* represents a rule body or head item.

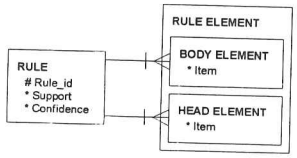
For the above form of item set organization we can distinguish two basic types of queries that are usually issued in searching purchase transactions or association rules:

1. Retrieve all purchase transactions that contain given subset of items

This type of retrieval can be used to determine the purchase transactions that satisfy or violate the specified rules. Finding the purchase transactions that violate strong rules allows deviation detection. This kind of queries could be also applied by a rule mining algorithm to discover association rules containing a specified set of items.

2. Retrieve all association rules that contain given subset of items in their bodies or heads

Queries for retrieval of association rules containing given items allow searching the database of rules for specific rules. Users impose the conditions on contents of rules that are to be retrieved from the database, e.g. a user may look for all rules that have the items (bread,butter) in their bodies.



RULES

rule_id	support	confidence
1	0.83	0.90
2	0.25	0.13

ELEMENTS

rule_id	item	type
1	bread	body
1	butter	head
2	bread	body
2	butter	body
2	milk	body
2	apples	head

Figure 4 Example data model and tables for association rule storage

Notice that both types of the queries are in fact instances of the Subset Search Problem - they consist in finding item sets that contain a given item subset.

Let us analyze physical properties of the database tables that are subject to the Subset Search Problem. In practical applications the number of item sets n_{trans} may be very large (supermarkets incrementally register thousands of transactions every day) and the average size of an item set is of the order of 10-100. Therefore, the selectivity of the attribute *GROUP_ID* will be very high because only 10-100 records from a large database table of even millions of records will have the same value of the attribute *GROUP_ID*. Moreover, since all items of an item set are usually being stored in a database at the same point of time, they are very likely to be located in one or very few adjacent disk blocks.

Let us now consider the properties of the attribute *ITEM*. In practical applications the number of all items n_{items} is relatively small (supermarkets sell a constant number of hundreds of different products), however, an individual item may occur in large number of item sets (e.g. milk is included in almost every transaction, beer may be included in 30% of transactions etc.). Therefore, the selectivity of the attribute *ITEM* is very low and its individual values are sparsely populated in a database.

2.1 Subset Searching Queries

The Subset Search Problem is not well supported by SQL query language and traditional query execution techniques. To illustrate the subset searching, we present an example of an SQL query retrieving from a database table *D* the identifiers of item sets containing the items 0, 7, 12 and 13. We will refer to this type of query as the *subset searching query*:

```
SELECT T1.GROUP_ID
FROM   D T1, D T2, D T3, D T4
WHERE  T1.GROUP_ID = T2.GROUP_ID
AND    T2.GROUP_ID = T3.GROUP_ID
AND    T3.GROUP_ID = T4.GROUP_ID
AND    T1.ITEM = 0
AND    T2.ITEM = 7
AND    T3.ITEM = 12
AND    T4.ITEM = 13;
```

To demonstrate the nature of the subset searching query, let us consider its execution plan, given in Figure 5. In fact, any execution plan for the query will have the similar structure. It will consist of four selection and three join operations. For the considered execution plan, at the first step, identifiers of all item sets containing items 0 and 7 are selected. These sets can be very large due to the low selectivity of the selection predicates - it is very likely that most of item sets (e.g. purchase transactions) will contain the items 0 and /or 7 (e.g. 0 may corresponds to "bread", etc.). Then, at the next step, the set of identifiers found in the previous step is joined with a set of identifiers of item sets containing the item 12. At the last step, it is joined with a set of identifiers of item sets containing the item 13. As the result, identifiers of all item sets containing items 0, 7, 12, and 13 are returned. There are two main disadvantages of these plans. First, as it can be seen, the number of joins in the query execution plan strictly depends on the size of the searched subset. Larger searched subset results in larger subset search query. Second, the

intermediate tables resulting from the selection operations may be very large due to the low selectivity of the selection predicates. The intermediate tables are joined to select those item set identifiers that are contained in all tables. This subset containment procedure is performed in step-by-step manner. So, it may occur that the system has to deal with very large intermediate tables at consecutive steps of the procedure. In fact, as we will show it later, sizes of intermediate tables have the main impact on the execution cost of this type of queries.

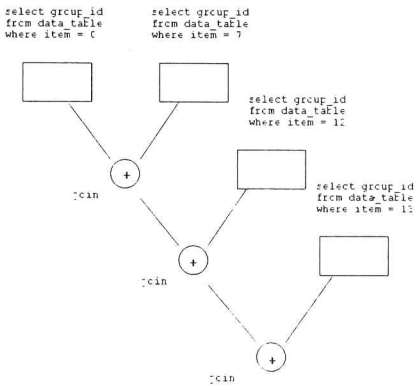


Figure 5: Execution plan for the example query

3. Database Accessing Methods Applied To The Subset Search Problem

A popular method of performing database joins without help of any indexes is sort-merge join method [6]. In this method, the database tables are first sorted according to the value of the join attribute. Then a pointer is associated with each table. The pointers point initially to the first record of the respective tables. As the sort-merge algorithm proceeds, the pointers move through the table. A group of records of one table with the same value of the join attribute is read. Then the corresponding tuples of the other table is read. Since the tables are sorted, records with the same value of the join attribute are in consecutive order.

Another method of performing database joins is hash-based join [12]. The basic approach of several hash-based join methods is to dynamically build a hash table on the

join attributes for one database table and then to probe this hash table using hash values on the join attributes of records from the other database table. The result of the join consists of the matching records. It is usually assumed that memory that is available is much smaller than the size of database tables to be joined, therefore it is impossible to build the hash table for the entire database table. The hash-join algorithms usually process the join in batches. In each batch, only a portion of a database table is read into memory and the corresponding hash table is built.

The role of indexing in query optimization is well understood in the database community. The purpose of an index is to provide pointers to the records in a table that contain a given key value, thus enabling efficient access to a subset of a database. The most popular database indexing techniques are based on B^+ trees. Figure 6 shows an example of the B^+ tree. Each non-leaf node contains entries of the form (v, p) where v is the separator value which is derived from the keys of the records and is used to tell which subtree holds the searched key, and p is the pointer to its child node. Each leaf node contains entries of the form (k, p), where p is the pointer to the record corresponding to the key k.

B^+ tree indexes are very effective in performing point and range queries for highly selective attributes. However, their performance significantly decreases for attributes that have a small number of key values compared to the number of records and that are uniformly spread over the database table. In the context of the Subset Search Problem, a B^+ tree index could be used for improving the joins on the attribute *GROUP_ID* of the database table *D*.

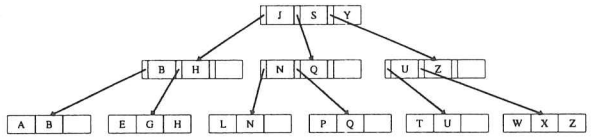


Figure 6: An example of the B^+ tree

Another type of index that received recently significant attention and was implemented by most of commercial DBMSs is Bitmap index. The Bitmap index is a collection of k 0-1 vectors, called bitmaps, where k is a number of indexed attribute's key values. Each bit in the bitmap corresponds to a record, and if the bit is set, it means that the corresponding record contains the key value. Figure 7 shows an example of a database table and its Bitmap index. The example Bitmap index consists of three bitmaps, one for each key value of the attribute *COL1*. The data retrieval by means of the Bitmap

index consists in fast scanning one or few of the bitmaps for non-zero elements. If the number of different key values is small, bitmaps are very space and time efficient. Moreover, Bitmap indexing efficiently merges indexes corresponding to several conditions in the WHERE clause. Records that satisfy some, but not all the conditions, are filtered out before the table itself is accessed.

In the context of the Subset Search Problem, a Bitmap index could be used for selection improvement of the attribute *ITEM* of the database table *D*. As we have already noticed, the attribute *ITEM* is used to filter the database table *D* for only those items that appear in the searched subset *S*. Due to the relatively small number of the key values, its selectivity is usually very low.

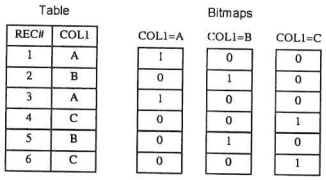


Figure 7: An example of the Bitmap index

4. Experimental Results

To assess the performance of the database accessing methods we performed several experiments on Oracle 7.3.1 DBMS, working on 2-processor Sun SPARCserver 630MP, with 128 MB of main memory. Experimental data sets were created by synthetic data generator *GEN* from Quest project [2]. *GEN* generates textual data files containing sets of numerical items. Several parameters affect the distribution of the synthetic data. These parameters are shown in Table 1.

To load the contents of the data files into the database, *Oracle SQL*Loader* program was used. The item sets were stored in database tables of the structure: (GROUP_ID NUMBER(6,0), ITEM NUMBER(6,0)).

parameter	value	parameter	value
n_{trans}	number of item sets, 50,000	n_{pats}	number of patterns, 500 and 10000
n_{items}	number of different items, 100 to 500	patlen	average length of maximal pattern, 4
t_{len}	average items per set, 15 to 30	corr	correlation between patterns, 0.25

Table 1 Synthetic data parameters

The main performance metric of the subset searching was mean execution time of test queries. *Oracle* SQL query tracing and *TKprof* profiler were used to measure precisely query execution time. Four types of test queries were issued: the queries that used sort-merge join method, the queries that used hash-based join method and the queries using B^+ tree index on the attribute *GROUP_ID* and Bitmap index on the attribute *ITEM*.

Figure 8 shows the performance of database accessing methods for different sizes of a searched subset. For increasing size of a searched subset, the number of database accesses also increases because of a greater number of query joins to be performed. Thus, the performance of all the methods is getting worse when the searched subset size is increasing. In general, as we expected, usage of database indexing results in shorter query execution times. Both B^+ tree and Bitmap indexing significantly outperformed sort-merge join and hash-based join methods. The fastest database accessing method appeared Bitmap indexing, which provided four times faster retrieval than sort-merge join and hash-based join methods.

Figure 9 demonstrates the effect of the average size of item sets on the performance of database accessing methods. For increasing size of item sets, the number of records in the database table also increases and the query execution time gets longer. For item sets of the size greater than 20 items, hash-based join method performs better than sort-merge join method. In general, when the item set size increases, the size of the database *D* increases but the number of key values of its attribute *GROUP_ID* remains constant, therefore B^+ tree index' performance is getting relatively worse. As we have observed, when the average size of item sets was greater than 25, then hash-based join method even outperformed B^+ tree indexing method. Again, the best database accessing method for all examined item sets sizes appeared Bitmap indexing.

Figure 10 shows the effect of the number of items stored in the database on the performance of database accessing methods. The number of items influences the size of intermediate join results. When the number of items is greater, the selectivity of the attribute *ITEM* is higher, thus the intermediate join results are smaller. The experiment showed, that this property does not significantly improve the performance of sort-merge join, hash-based join as well as B^+ tree indexing. We have observed a lowering improvement of Bitmap indexing performance for increasing number of items. Again, the best database accessing method for all examined numbers of items was Bitmap indexing.

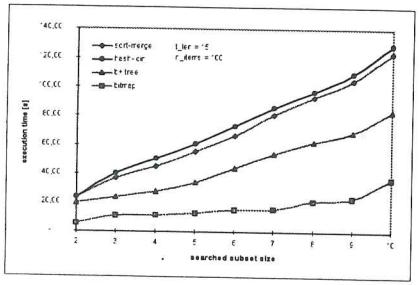


Fig.8: Query execution time vs. searched subset size

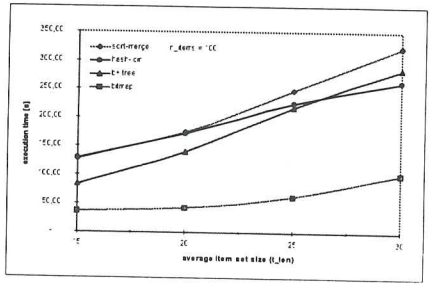


Fig.9: Query execution time vs. average item set size

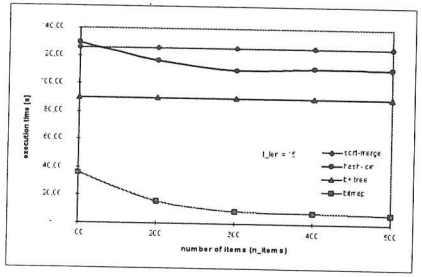


Fig.10: Query execution time vs. number of items

5. Concluding Remarks

In this paper we introduced the Subset Search Problem in relational databases which consists in retrieval of multi-item sets that contain a given multi-item subset. The Subset Search Problem has many applications in the field of data mining. We analyzed and compared the performance of the subset searching queries for different database accessing methods. We showed experimentally that Bitmap indexing is the best method of improving the subset searching queries performance. However, we realize that even Bitmap indexing results in hardly acceptable query execution times.

The results of this paper can be extended in several directions. First, we may study any alternative plans of executing the Subset Search queries. As we have noticed, the number of joins in the presented query execution plan strictly depends on the size of the searched subset. Larger searched subset results in larger subset search query. This property significantly reduces the performance of large subset searching queries. Second, new index structures could be invented to reduce time of large subset searching in large relational databases. We suspect that if an index structure was built over data sets instead

of items, the performance of the subset searching could be significantly better. Furthermore, we would like to analyze different data storage strategies (e.g. index clusters, hash clusters) for their application to the Subset Search Problem.

6. Bibliography

[1] Agrawal R., Imielinski T., Swami A., Mining Association Rules Between Sets of Items in Large Databases, Proc. of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, May 26-28 1993,

[2] Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T., The Quest Data Mining System,

[3] Comer D., The Ubiquitous B-tree, Comput. Surv. 11, 1979,

[4] Fayyad U., Piatetsky-Shapiro G., Smyth P., The KDD Process for Extracting Useful Knowledge from Volumes of Data, Comm. of the ACM, Vol. 39, No. 11, November 1996,

[5] Imielinski T., Manilla H., A Database Perspective on Knowledge Discovery, Comm. of the ACM, Vol. 39, No. 11, November 1996,

[6] Korth H. F., Silberschatz A., Database Systems Concepts, McGraw-Hill, 1986,

[7] Morzy T., Zakrzewicz M., SQL-Like Language For Database Mining, ADBIS'97 Symposium, St. Petersburg, September 1997,

[8] O'Neil P., Graefe G., Multi-Table Joins Through Bitmapped Join Indices, SIGMOD Record, September, 1995,

[9] O'Neil P., Model 204 Architecture and Performance, Springer-Verlag Lecture Notes in Computer Science 359, 2nd Int. Workshop on High Performance Transactions Systems (HTPS), Asilomar, CA, 1987,

[10] O'Neil P., Quass D., Improved Query Performance with Variant Indexes, Proc. Of the 1997 ACM SIGMOD, Tucson, Arizona, 1997,

[11] Piatetsky-Shapiro G., Frawley W.J., editors, Knowledge Discovery in Databases, MIT Press, 1991,

[12] Shapiro L. D., Join processing in database systems with large memories, ACM Trans. Database Syst., Vol. 11, No. 3, 1986

Domain Knowledge in Preferential Models

B.C.M. Wondergem, P. van Bommel, T.W.C. Huibers, Th.P. van der Weide

Computing Science Institute, University of Nijmegen
Toernooiveld 1, NL-6525 ED, Nijmgegen, The Netherlands
E-mail: bernd@cs.kun.nl

Keywords: Information Retrieval, Logical Relevance, Preferential Models, Domain Knowledge, User Preferences.

Abstract

In Information Retrieval, user preferences and domain knowledge play an important role. This article shows how to incorporate domain knowledge in a logical framework and provides a mechanism to exploit user preferences for personalizing domain knowledge, based on the inferences made in the matching functions. The matching functions are essentially symbolic logical inferences. The logic used in this article is that of Preferential Models, which is augmented with domain knowledge by providing an enriched aboutness relation. However, the techniques described in this article are applicable to other logics as well. A way to personalize the domain knowledge is given, which also gives the user insight into the workings of the matching functions. In addition, sound inference rules, which are tailor-made for the domain knowledge, are provided.

1 Introduction

Information Retrieval (IR) (see e.g. [8]) has gained growing attention over the past years, due to the enormous growth of online available information. The vast amount of information an IR system has to cope with nowadays requires the use of additional information besides queries and document characterizations. For instance, domain knowledge and user preferences should be used in the matching techniques.

User preferences are becoming increasingly important in selecting the tiny relevant part of the available information. User preferences come in a variety of sorts, e.g. browsing preferences (see e.g. [2]), preferences on keywords and descriptors (see e.g. [12]), preferences on documents (see e.g. [10]), and preferences on inference rules of the matching process (see e.g. [4]). We propose a mechanism to exploit user preferences on domain knowledge.

This mechanism allows users to have insight into the workings of the IR system, or rather, in the matching process. To this end, the matching process is presented to the user in detail so that it becomes clear which preferences and instances of domain knowledge were actually used. As a consequence, users will be able to formulate good queries more efficiently. Visualizing the matching process requires symbolic matching, such as stepwise logical inference.

The logic used in this article are Preferential Models, developed by Kraus *et al.* (see e.g. [5]), which is a well-defined framework capable of modeling and reasoning with user preferences on documents. These preferences, stemming from the user's information need, influence the inferences that can be made. However, these models have not been fully adapted for IR yet, i.e., not all the concepts required in IR are supported by Preferential Models yet.

Domain knowledge, for instance, which is highly beneficial for an increase in recall, has not been incorporated in these models yet. The main reason for this is the fact that the way in which the so-called *universe of reference* is traditionally defined by logicians is inappropriate for IR purposes. The traditional way discards of many documents a priori, thus probably losing relevant documents at an early stage.

In order to equip Preferential Models with domain knowledge, a different approach has to be taken. Preferential Models are augmented with domain knowledge by defining an enriched aboutness relation, the IR counterpart of the logical validity relation, which ensures no a priori loss of documents. Our approach treats the domain knowledge as an autonomous class of data, easing the way to personalize it without the need for costly recomputations.

Since domain knowledge is not generally applicable, but rather valid or invalid with respect to a certain information need, a way to personalize this knowledge is proposed. This ensures that user preferences on the domain knowledge are properly taken into account, i.e., that only the instances of the domain knowledge that are considered valid by the user are used in relevance proofs.

The outline of this paper is as follows: Section 2 gives a brief introduction to Preferential Models. Section 3 describes the exact nature of the domain knowledge considered, provides an appropriate way to incorporate the domain knowledge in Preferential Models, and describes a way of exploiting and personalizing it. Section 4 concludes with a brief summary and some directions for further research.

2 Introduction to Preferential Models

This section introduces the basics of Preferential Models, necessary for the rest of the article. Readers familiar with Preferential Models probably can skip most of this section. The original theory about Preferential Models can be found in [5]. In [3] and [10], Preferential Models are adapted for IR. This section mainly gives a brief summary of these articles.

The query and the characterization languages are based on a *descriptor language*. This descriptor language consists of a number of keywords and of composed descriptors, formed out of other descriptors with the information composition operator $\oplus$.

Definition 1 (Descriptor Language) For a given finite set of keywords K, the *descriptor language* L_K is defined to be the smallest superset of K such that if $i \in L_K$ and $j \in L_K$ then also $(i \oplus j) \in L_K$.

Queries are issued against a framework called a *retrieval model*, consisting of the set of documents to be queried, a characterization function for those documents which assigns a descriptor to each document, and a descriptor language from which the query and document characterizations are drawn.

Definition 2 (Retrieval Model) A *retrieval model* is a triple $M = \langle D, \chi, L \rangle$, where D is a finite set of documents, $\chi : D \mapsto \wp(L)$ is a characterization function for D and L is a descriptor language.

Example 2.1

Consider the set of documents $D = \{d_1, d_2, d_3, d_4\}$ *and a characterization function* $\chi : D \mapsto L_K$ *such that* $\chi(d_1) = \{\mathsf{wave}, (\mathsf{wind} \oplus \mathsf{surfing})\}$, $\chi(d_2) = \{(\mathsf{wave} \oplus \mathsf{surfing})\}$, *and* $\chi(d_3) = \{(\mathsf{internet} \oplus \mathsf{surfing})\}$, $\chi(d_4) = \{(\mathsf{surfing})\}$, *where* $\{\mathsf{surfing}, \mathsf{wind}, \mathsf{wave}, \mathsf{internet}\} \subseteq K$. *Then,* $M = \langle D, \chi, L_K \rangle$ *is a retrieval model.* □

The *aboutness relation* formally defines if a document is *about* a certain descriptor. The definition is relative to a retrieval model since it hinges on the characterization function. A document is about a descriptor if that descriptor is part of the document's characterization. Furthermore, the compositional structure of descriptors is taken into account. The operator $\oplus$ approximates informational composition by assigning it the properties of logical conjunction.

Definition 3 (Aboutness Relation) Let $M = \langle D, \chi, L \rangle$ be a retrieval model. Then the aboutness relation for M, denoted $\models_a^M \subseteq D \times L$, between a document and a descriptor is defined as:

- $d \models_a^M i \Leftarrow i \in \chi(d)$

- $d \models_a^M (i \oplus j) \Leftrightarrow d \models_a^M i$ and $d \models_a^M j$

User preferences on documents are denoted by a binary relation on documents, $\sqsubset_N \subseteq D \times D$, where N denotes that the relation is based on information need N. The expression $d \sqsubset_N d'$ means that, considering information need N, document d is preferred over document d'. *Preferential Models* consist of a retrieval model equipped with a preference relation on documents.

Definition 4 (Preferential Model) Let $M = \langle D, \chi, L \rangle$ be a retrieval model and $\sqsubset_N$ a preference relation on D. Then the tuple $P = \langle M, \sqsubset_N \rangle$ is called a *Preferential Model.*

A key notion in the underlying model, that of the set of documents together with the preferences on the documents, is that of *most preferred documents* about a certain descriptor, i.e., all documents about that descriptor for which there are no other documents about the same descriptor that are more preferred.

Definition 5 (Most Preferred Documents) Let $P = \langle M, \sqsubset_N \rangle$ be a Preferential Model, where $M = \langle D, \chi, L \rangle$. Let $[\![i]\!]_P$ denote all the documents of D that are about descriptor i in model P, i.e. $[\![i]\!]_P = \{d \in D | d \models_a^M i\}$. The set of *most preferred i-documents* in P, denoted $\mathcal{M}_P(i)$, is defined by:

- $\mathcal{M}_P(i) = \{d \in [\![i]\!]_P | \, \forall d' \in [\![i]\!]_P : d' \not\sqsubset_N d\}$

Two meta operators on descriptors are available in Preferential Models: $|\!\sim_N$ for *conditional assertions*, and $\perp_N$ for *preferential preclusions*. The expression $i|\!\sim_N j$ means that from i one can plausibly conclude j in the light of the information need N. It is valid iff in the underlying model all most preferred i-documents are also about j. Conditional assertions are also called *defaults*, since $i|\!\sim_N j$ means that the user, by default, expects delivered documents about i also to be about j.

The expression $i \perp_N j$ means that i contradicts j in the light of information need N and is valid iff in the underlying model none of the most preferred i-documents is also about j. A preferential preclusion $i \perp_N j$ means that the user is not interested in i-documents that are also about j.

Definition 6 (Validity in Preferential Models) Let P be a Preferential Model. Let $i, j \in L_K$. Furthermore, the language $\mathcal{L}_K$ is defined to be the set $\{i \alpha j\}$, where α is an element of the set $\{|\!\sim_N, |\!\not\sim_N, \perp_N, \not\perp_N\}$. Then, the relation $\models$ between P and an element of $\mathcal{L}_K$ is defined for defaults, preclusions, and their negations, respectively, by

$$
\begin{array}{lcl}
P \models i|\!\sim_N j & \Leftrightarrow & \mathcal{M}_P(i) \subseteq [\![j]\!]_P \\
P \models i \perp_N j & \Leftrightarrow & \mathcal{M}_P(i) \cap [\![j]\!]_P = \varnothing \\
P \models i \not\perp j & \Leftrightarrow & P \not\models i \alpha j
\end{array}
$$

Example 2.2

Consider a user in need of information about leisure forms of surfing. This means that the user is interested in forms of surfing like wind and wave surfing, but not in Internet surfing. This can be modeled by the following defaults and preclusions: surfing$|\sim_N$wave, surfing $\not\sim_N$ wind, *and* surfing $\bot_N$ internet. *A Preferential Model that satisfies these defaults and preclusions can be constructed by defining a suitable preference relation (see e.g. [10]). Consider the Retrieval Model M from example 2.1 and the preferential relation* $\sqsubset_N$ *for which* $d_1 \sqsubset_N d_2$, $d_1 \sqsubset_N d_3$, $d_1 \sqsubset d_4$, $d_2 \sqsubset_N d_3$, *and* $d_4 \sqsubset_N d_3$. *Note that the preferential relation satisfies the abovementioned defaults and preclusions. The Preferential Model* $P = \langle M, \sqsubset_N \rangle$ *is shown in Figure 1 where an arrow* $d \rightarrow d'$ *vizualizes* $d' \sqsubset_N d$. □

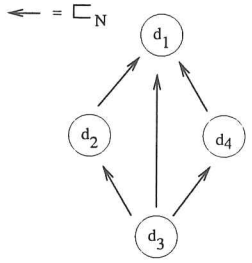


Figure 1: The Preferential Model of example 2.2.

If the preferential ordering of a Preferential Model $P = \langle M, \sqsubset_N \rangle$ is a strict partial order, i.e., if $\sqsubset_N$ is irreflexive, anti-symmetric, and transitive, the following inference rules are valid (for proofs, see [5]):

$$\frac{}{i|\sim_N i}\ \text{Reflexivity} \qquad \frac{(i \oplus j)|\sim_N k,\ i|\sim_N j}{i|\sim_N k}\ \text{Cut} \qquad \frac{i|\sim_N j,\ i|\sim_N k}{(i \oplus j)|\sim_N k}\ \text{Cautious Monotonicity} \qquad \frac{i_0|\sim_N i_1,\ \ldots,\ i_{k-1}|\sim_N i_k,\ i_k|\sim_N i_0}{i_0|\sim_N i_k}\ \text{Loop}$$

In addition, a number of derived rules are available:

$$\frac{i|\sim_N j,\ j|\sim_N i,\ i|\sim_N k}{j|\sim_N k}\ \text{Equivalence} \qquad \frac{i|\sim_N j,\ i|\sim_N k}{i|\sim_N (j \oplus k)}\ \text{Composition} \qquad \frac{i|\sim_N j,\ i \not\sim_N k}{(i \oplus k)|\sim_N j}\ \text{Rational Compositional Monotonicity}$$

In processing a query, those documents are returned, i.e. considered relevant, whose characterization plausibly implies the query. That is, for a query q in Preferential Model P, the result set is defined by:

$$\texttt{result}(q) = \{d \in D | \exists i \in \chi(d) : P \models i|\sim_N q\}$$

Example 2.3

Consider again the retrieval model from example 2.1 and the user preferences from example 2.2. Furthermore, suppose the user enters the query surfing. *Then,* result(surfing) $= \{d_1, d_2\}$. *Document* d_3 *is not considered relevant, since the query cannot be plausibly proven from* d_3*'s characterisation. Document* d_1*, for example, can be proven relevant with rule* Cautious Monotonicity *using* surfing$|\sim_N$surfing *(from the* Reflexivity *axiom) and* surfing$|\sim_N$wind *(from the user preferences) as antecedents. Document* d_2 *can be proven relevant using the rule* Rational Compositional Monotonicity. □

3 Domain Knowledge

In order to enhance the recall of an IR system equipped with Preferential Models, we augmented the model with domain knowledge. This will lead to a more realistic IR system capable of semantic expansion of descriptors.

First, the exact nature of the domain knowledge is identified. The second subsection describes our approach to equip Preferential Models with domain knowledge. The third subsection describes a way to personalize the domain knowledge that is argued to be beneficial for precision.

3.1 Types of Domain Knowledge

To incorporate domain knowledge in Preferential Models, it first has to be translated to or interpreted in the context of Preferential Models. In [5] two inference rules that express the influence of the set of underlying logical worlds on the notion of conditional assertions are given. In our case, this influence comes down to the influence of semantic relations between descriptors, which should hold in the underlying set of documents. The two rules exploit generally valid instances, i.e., valid in all possible worlds, of material equivalence (denoted by $\leftrightarrow$ in propositional logic) and material implication (denoted by $\rightarrow$). If the domain knowledge can be translated to instances of informational equivalence or information containment (as defined below) that are valid in all documents, it can be incorporated in Preferential Models. We first show the exact nature of the relations on descriptors that can be embodied.

Definition 7 (Domain Relations) Let $R : L \times L$ be a relation on descriptors. Then, R is called *an equivalent relation* iff for every iRj for every $d \in D : d \models_a i \Leftrightarrow d \models_a j$. Furthermore, relation R is called *a containment relation* iff for every iRj for every $d \in D : d \models_a i \Rightarrow d \models_a j$.

Example 3.1

An example of an equivalent relation on descriptors is the synonym relation, since every document about a certain descriptor is also about the synonyms of this descriptor. For example, a document about animals is also about beasts. Relations such as hypernyms (e.g., the **is-a** *relation) and holonyms (e.g., the* **part-of** *relation) are examples of containment relations. For instance, a document about cows is also about animals.* □

Note that the notion of an equivalent relation corresponds to the model-theoretical interpretation of material equivalence. The notion of a containment relation corresponds to material implication.

Domain knowledge expressed as either an equivalent or containment relation on descriptors can be incorporated in Preferential Models, as shown in the next section. The rest of this paper assumes that a knowledge base is available, denoted by K, containing domain knowledge in the form of instances of equivalent and containment relations on descriptors. Suitable domain knowledge can be obtained from thesauri and lexical databases, such as WordNet[1].

3.2 Augmenting Preferential Models

A correct exploitation of the knowledge base K in Preferential Models ensures the validity of all elements of K in all the documents considered. The set of all documents considered is called the *universe of reference*. Quoting [5] (page 172):

> Typical universes of reference are given by the set of all propositional worlds [in our case documents] that satisfy a given set of formulas [K].

Note that this notion of satisfaction finds its counterpart in IR in the notion of aboutness, i.e., logical worlds satisfying formulas correspond to documents about descriptors. Also note that aboutness is defined in terms of the characterization function. Because the characterization function most likely is not closed under the knowledge base (does not include all the information available in the knowledge base), the abovementioned notion of the universe of reference implies that only a part of the whole collection of documents would be used, i.e., would comprise the universe of reference. This set of 'useful' documents consists of exactly those documents that satisfy all formulae of the knowledge base. Because characterization functions only depict the actual content of documents rather than related domain knowledge, this set of documents will be very small.

[1] Created by Cognitive Science Laboratory, Princeton University, 221 Nassau St., Princeton, NJ 08542. WordNet is available for anonymous ftp from `clarity.Princeton.edu` and `ftp.ims.uni-Stuttgart.de`.

Example 3.2

To illustrate this, consider the expression dog **is-a** animal. *It is not likely that all documents that are characterized by* dog *are also a priori characterized by* animal. *For most current characterization algorithms, this would only be the case if* animal *actually is an explicit part of all the documents' content.* □

The example shows that the abovementioned notion of the universe of reference is not valid for IR, since a large number of possibly relevant documents would a priori be discarded of. The main idea now is that rather than restricting the set of documents to those that satisfy the knowledge base, it is actually necessary to impose the knowledge base on the complete set of documents.

The semantics of the formulae in the knowledge base suggest the use of a more powerful aboutness relation. Two major ways to accomplish this exist, i.e., ways to ensure that K is properly incorporated without a priori restricting the set of documents.

One approach does not change the actual definition of the aboutness relation (see definition 3). Rather, it augments the characterizations on the basis of the domain knowledge, thus indirectly obtaining a more powerful aboutness relation. The old characterizations are expanded with equivalent and containment relations. However, this approach has a number of disadvantages. First of all, the augmented characterizations are considerably larger than the original ones. Together with the large number of documents usually considered in IR applications, this can become costly in terms of space. Second, if the knowledge base changes, all characterizations have to be recomputed. Last, no distinction can be made anymore between the original characterization's content and the added descriptors from the knowledge base. This is necessary if clear insight into the workings of the matching process and the role of the knowledge base is required. Personalizing the knowledge base, as described in the next subsection, also becomes much more complicated.

A better approach to incorporating the knowledge base in Preferential Models augments the aboutness relation itself without altering the original characterizations. It does not combine the knowledge base with the characterizations, but treats the domain knowledge as an autonomous class of data. Inductively defined rules are constructed for the different types of relations in the knowledge base. The augmented aboutness relation includes the original one and is closed under the knowledge base.

Definition 8 (Augmented Aboutness Relation) Let M be a retrieval model. Furthermore, let $\models_a^M$ be the aboutness relation for M as defined in definition 3 and let K be the knowledge base. The *augmented aboutness relation*, denoted $\models_a^{M,K}$, is defined as follows:

- $d \models_a^{M,K} i$ if $d \models_a^M i$
- $d \models_a^{M,K} i$ if $d \models_a^{M,K} j$ and $iEj \in K$ or $jEi \in K$ where E is an equivalent relation

- $d \models_a^{M,K} i$ if $d \models_a^{M,K} j$ and $jCi \in K$ where C is a containment relation

This approach does not have the abovementioned disadvantages: the original characterization function remains unaltered, the knowledge base can easily be changed without the need for costly recomputations, and the aboutness relation clearly distinguishes the influence of the characterizations from the semantical relations in the knowledge base.

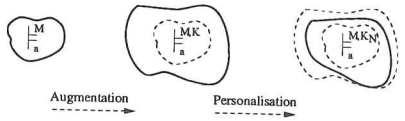


Figure 2: From the standard aboutness relation ($\models_a^M$) via augmentation with domain knowledge to the augmented aboutness relation ($\models_a^{M,K}$), and further to a personalized aboutness relation ($\models_a^{M,K_N}$). The bold black lines indicate the borders of the aboutness relations.

Domain knowledge has now been properly incorporated in Preferential Models. We will proceed and discuss how to actually *use* the domain knowledge at the level of inferences. Figure 2 clarifies the augmentation of the aboutness relation as well as the personalization of it, as described in the next subsection.

The two types of relations available in the knowledge base each have their own corresponding inference rules. For each equivalent relation, two inference rules are available: left and right replacement. For each containment relation, one inference rule is available: right weakening. Those rule schemes, which are sound with respect to the underlying model, are listed below. The symbols E and C are to be instantiated by an equivalent and a containment relation, respectively.

$$\frac{iEj, i|\sim_N k}{j|\sim_N k} \qquad \frac{iEj, k|\sim_N i}{k|\sim_N j} \qquad \frac{iCj, k|\sim_N i}{k|\sim_N j}$$

Left E Replacement Right E Replacement Right C Weakening

Example 3.3

Consider the syn *relation, which is an equivalent relation, and the* is-a *relation, which is a containment relation. If these relations are part of the knowledge base, the following inference rules are valid:*

$$\frac{i \text{ syn } j, i \,|\sim_N k}{j \,|\sim_N k} \qquad \frac{i \text{ syn } j, k \,|\sim_N i}{k \,|\sim_N j} \qquad \frac{i \text{ is-a } j, k|\sim_N i}{k \,|\sim_N j}$$

Left syn Replacement Right syn Replacement Right is-a Weakening

□

3.3 Personalizing Domain Knowledge

Although the domain knowledge is usually of a general nature, the elements of the knowledge base will not be valid in all contexts. To illustrate this, consider again the expression dog is-a animal, which is meaningful in most cases but not in the context of, say, hotdogs. We claim that the elements of the knowledge base are either valid or invalid given a single information need (context).

The use of invalid elements of the knowledge base will cause a drop in precision as documents will be retrieved on the basis of invalid information. Therfore, a way has to be found to select only the valid elements of the knowledge base, given a specific information need. This section shows how detailed relevance feedback can be used to accomplish this, i.e., to personalize the knowledge base.

The relevance feedback ultimately has to be given at the level of elements of the knowledge base. We envisage that, upon display of the relevant documents, the user can provide negative relevance feedback, i.e., identify which documents are not considered relevant by the user. Those documents might be proven relevant by the system on the basis of invalid elements of the knowledge base. In order to be able to check this and to enable the user to specify this explicitly, the notion of a relevance proof of a document is introduced.

Definition 9 (Relevance Proof) Let Δ be a set of conditional assertions and preferential preclusions upon which a Preferential Model $P = \langle M, \sqsubset_N \rangle$ is based. In addition, let K be a knowledge base consisting of a number of domain relations and let R be a set of sound inference rules for P. Given a document d and a query q, a *relevance proof* of d with respect to q is a finite sequence $\Phi(d,q) = \phi_1, \ldots, \phi_n$, where $\phi_n = i|\sim_N q$ such that $i \in \chi(d)$ and for every $\phi_i (1 \leq i \leq n)$ it holds that

- ϕ_i is one of the given conditional assertions or preferential preclusions, i.e. $\phi_i \in \Delta$, or
- ϕ_i belongs to the knowledge base, i.e. $\phi_i \in K$, or
- ϕ_i is the conclusion of an inference rule $r \in R$ and the antecedents of this rule appear in $\phi_1, \ldots, \phi_{i-1}$.

Furthermore, define the restriction of a relevance proof $\Phi(d,q)$ to the knowledge base K, denoted $\Phi_K(d,q)$, as $\{\phi \in \Phi | \phi \in K\}$.

Upon the user's indication that a document d is not relevant with respect to a query q, its relevance proof restricted to the knowledge base, i.e., $\Phi_K(d,q)$, is displayed. The user can indicate which elements of $\Phi_K(d,q)$ do not correspond to his information need. Those elements are deleted from the knowledge base, thus obtaining a personalized knowledge

base. This personalized knowledge base is denoted by K_N, where N indicates that this knowledge base corresponds to information need N. This plan of attack has been proven effective in expert systems (see e.g. [6]).

At later stages in the query session, the personalized knowledge base is used instead of the original one. This causes less documents to be proven relevant, exactly those that relied on invalid elements of the knowledge base. In an iterative process, this leads to a suitable knowledge base which gives better precision.

Relevance proofs can be described as proof trees. By exploiting this tree structure, the process of explaining the inferences made in relevance decisions can be refined. As a result, more detailed information about the preferences of the user can be obtained. For example, one could obtain user preferences on the inference rules or on the defaults and preclusions that were used to construct the Preferential Model.

3.4 Other linguistic relations

There are many other liguistic relations. Most logical models lack the expressive power, however, to include all those relations. This section describes a number of linguistic relations and shows if those can be included in Preferential Models.

The *entailment relation* (see [7]) defines semantically proximate descriptors. For instance, if the user is interested in cars, he is also likely to be interested in drivers, a semantically proximate concept. The entailment relation cannot, however, be modeled as a containment relation, since the entailments are not logically strict, i.e., there is too much vagueness in the entailment relation. A logic that includes (numerical) measures of certainty seems more suitable for this type of relation.

Because of the asymmetry in (material) implications, containment relations can also be used with preclusions. An instance of a containment relation, say dog is-a animal, means that all documents about dogs are also about animals. At the same time, it also states that all documents that are not about animals cannot be about dogs either. The way in which containment relations are incorporated in Preferential Models guarantees that for every containment relation C, we have for every document d: $d \not\models_a i \,\&\, j \; C \; i \Rightarrow d \not\models_a j$. This implies that for every containment relation C we have a sound inference rule:

$$\frac{i \perp_N j, k \; C \; j}{i \perp_N k}$$

Preclusion Weakening

We can thus use the 'inverted containment' relations, for instance hyponyms, in weakening preferential preclusions.

Stemming brings derived word forms back to the same stem, thus introducing a form of equivalence classes. Spelling mistakes can also be dealt with by introducing equivalence classes, i.e., equivalence classes of misspelt words. The equivalence classes obtained in

either way define an equivalent relation, which, as was shown earlier, can be exploited in Preferential Models. Thus, the technique described is also applicable to additional information different from domain knowledge.

To enrich the expressive power of Preferential Models used for Information Retrieval, the full underlying logic should be exploited. That is, the logical connectors available in Preferential Models, $\neg$ for negation and $\vee$ for logical disjunction, should also be used in a setting for IR.

4 Conclusions

In this article, we have shown how to incorporate domain knowledge in Preferential Models. The domain knowledge could consist of equivalent and containment relations. We also indicated how to personalize the domain knowledge, i.e., how to exploit user preferences on the domain knowledge. This was done by the symbolic inferences made in relevance proofs and negative relevance feedback from the user. The visualization of the relevance proofs gives the user insight into the workings of the matching function.

The techniques described can also be applied in the context of Information Filtering (see e.g. [1]), and Information Discovery (see e.g. [11]). The advantage in those settings is that a user profile, capturing long-term user interests with their corresponding user preferences, is maintained. The domain knowledge can then be naturally personalized with respect to the user profile. In the PROFILE information filtering project of the University of Nijmegen, a lot of attention is paid to user modeling. The resulting user profiles can serve as a proper basis for personally adapted Preferential Models. For more information about the user modeling component of the PROFILE system, see for instance [9].

Further research has to be conducted into different domain relations. More complicated semantic relations, like conditional relations, should be incorporated. For example, one can think of a relation that specifies 'soft synonyms', i.e., synonyms within a certain context. Since the limits of pure Preferential Models have nearly been reached, research must be conducted into possible augmentations that lead to more powerful Preferential Models.

References

[1] N. Belkin and W. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, Dec. 1992.

[2] F. Berger and P. van Bommel. Personalized Search Support for Networked Document Retrieval Using Link Inference. In R. Wagner and H. Thoma, editors, *Proceedings of*

the 7th International Conference DEXA'96 on Data Base and Expert System Applications, volume 1134 of *Lecture Notes in Computer Science*, pages 802–811, Zurich, Switzerland, Sept. 1996. Springer-Verlag.

[3] P. Bruza and B. van Linder. Preferential Models of Refinement Paths. Technical report, Queensland University of Technology, Brisbane, Australia, 1996. To appear.

[4] T. Huibers and N. Denos. A qualitative ranking method for logical information retrieval models. Technical Report RAP95-005, Groupe MRIM of the Laboratoire de Génie Informatique, Grenoble, France, Aug. 1995.

[5] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44:167–207, 1990.

[6] P. Lucas and L. v. d. Gaag. *Principles of Expert Systems.* Addison-Wesley, Reading, Massachusetts, 1991.

[7] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five Papers on WordNet. Technical report, Cognitive Science Laboratory, Princeton University, August 1993.

[8] C. v. Rijsbergen. *Information Retrieval.* Butterworths, London, United Kingdom, 2nd edition, 1979.

[9] J. Simons. Using a semantic user model to filter the world wide web proactively. In A. Jameson, C. Paris, and C. Tasso, editors, *Proceedings of the sixth international Conference UM97.* SpringerWienNewYork, 1997.

[10] B. Wondergem. Preferential Structures for Information Retrieval. Master's Thesis INF-SCR-96-21, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1996.

[11] B. Wondergem, P. v. Bommel, T. Huibers, and T. v. d. Weide. Towards an Agent-Based Retrieval Engine. In J. Furner and D. Harper, editors, *Proceedings of the 19th BCS-IRSG Colloquium on IR research*, pages 126–144, Aberdeen, Scotland, April 1997.

[12] B. Wondergem, W. v. d. Hoek, T. Huibers, and C. Witteveen. Preferential Semantics for Query by Navigation. In K. van der Meer, editor, *Informatiewetenschap 1996*, pages 153–168, Delft, the Netherlands, December 1996.

Integration of Aggregate Approach in Knowledge Representation of the Multimodal Transport Evaluation System

Dale DZEMYDIENE
Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania
E-mail: daledz@ktl.mii.lt

Henrikas PRANEVICIUS
Kaunas University of Technology
Studentu 50, 3028 Kaunas, Lithuania
E-mail: hepran@if.ktu.lt

Abstract. The aims of information systems to aid in organisational tactic and strategic management processes encourage additional methods and techniques for decision support. Multimodal transportation has been considered as a rapidly changing and complex technology. Some issues of requirement specification for decision support systems of such a type are considered. Some possibilities of integrating a semantic model for structural information representation and an aggregate approach to functional representation of processes are presented. The created simulation model of multimodal transportation makes it possible to evaluate competitive features of specified routes between western and eastern directions of Baltic countries and also to investigate the Lithuanian position in multimodal transportation.

1. Introduction

The technology for building information systems (assistants of organisational tactic and strategic management processes) must provide methods for acquisition, structural representation of many types of knowledge in respect of large, shared and distributed data bases. A rapidly changing information environment may require for additional techniques for strategic planning, operation control and decision support.

The essential feature of multimodal transportation is inclusion of several types of transport into the process of freight transportation. A separate cycle of transportation of dispatch can include road haulage, rail freightage, as well as water transportation. The

the 7th International Conference DEXA'96 on Data Base and Expert System Applications, volume 1134 of *Lecture Notes in Computer Science*, pages 802–811, Zurich, Switzerland, Sept. 1996. Springer-Verlag.

[3] P. Bruza and B. van Linder. Preferential Models of Refinement Paths. Technical report, Queensland University of Technology, Brisbane, Australia, 1996. To appear.

[4] T. Huibers and N. Denos. A qualitative ranking method for logical information retrieval models. Technical Report RAP95-005, Groupe MRIM of the Laboratoire de Génie Informatique, Grenoble, France, Aug. 1995.

[5] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44:167–207, 1990.

[6] P. Lucas and L. v. d. Gaag. *Principles of Expert Systems.* Addison-Wesley, Reading, Massachusetts, 1991.

[7] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five Papers on WordNet. Technical report, Cognitive Science Laboratory, Princeton University, August 1993.

[8] C. v. Rijsbergen. *Information Retrieval.* Butterworths, London, United Kingdom, 2nd edition, 1979.

[9] J. Simons: Using a semantic user model to filter the world wide web proactively. In A. Jameson, C. Paris, and C. Tasso, editors, *Proceedings of the sixth international Conference UM97.* SpringerWienNewYork, 1997.

[10] B. Wondergem. Preferential Structures for Information Retrieval. Master's Thesis INF-SCR-96-21, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1996.

[11] B. Wondergem, P. v. Bommel, T. Huibers, and T. v. d. Weide. Towards an Agent-Based Retrieval Engine. In J. Furner and D. Harper, editors, *Proceedings of the 19th BCS-IRSG Colloquium on IR research*, pages 126–144, Aberdeen, Scotland, April 1997.

[12] B. Wondergem, W. v. d. Hoek, T. Huibers, and C. Witteveen. Preferential Semantics for Query by Navigation. In K. van der Meer, editor, *Informatiewetenschap 1996*, pages 153–168, Delft, the Netherlands, December 1996.

Integration of Aggregate Approach in Knowledge Representation of the Multimodal Transport Evaluation System

Dale DZEMYDIENE
Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania
E-mail: daledz@ktl.mii.lt

Henrikas PRANEVICIUS
Kaunas University of Technology
Studentu 50, 3028 Kaunas, Lithuania
E-mail: hepran@if.ktu.lt

Abstract. The aims of information systems to aid in organisational tactic and strategic management processes encourage additional methods and techniques for decision support. Multimodal transportation has been considered as a rapidly changing and complex technology. Some issues of requirement specification for decision support systems of such a type are considered. Some possibilities of integrating a semantic model for structural information representation and an aggregate approach to functional representation of processes are presented. The created simulation model of multimodal transportation makes it possible to evaluate competitive features of specified routes between western and eastern directions of Baltic countries and also to investigate the Lithuanian position in multimodal transportation.

1. Introduction

The technology for building information systems (assistants of organisational tactic and strategic management processes) must provide methods for acquisition, structural representation of many types of knowledge in respect of large, shared and distributed data bases. A rapidly changing information environment may require for additional techniques for strategic planning, operation control and decision support.

The essential feature of multimodal transportation is inclusion of several types of transport into the process of freight transportation. A separate cycle of transportation of dispatch can include road haulage, rail freightage, as well as water transportation. The

multimodal transportation may be considered as a dynamically changing and complex technology.

The problem of expression of behavioural aspects (temporal relationship of process interaction and their determination in time, synchronisation of decision making and information processing, communication between objects, etc.) arose in a multimodal transportation domain. The representation of causal interaction of processes could relate the use and analysis of information structures and simultaneously express mutual object communication aspects in time. In many conceptual models the aspects of static and dynamic representation are being considered now (methodologies like REMORA [11], TEMPORA [13], BIER[7]). Some recent models [1], [10], [12] are integrated object-oriented concepts in conceptual design and reduce the complexity of design tasks by specifying object classes, inheritance links and actions in these classes. Different mathematical schemes are used for creating formal descriptions of dynamic systems, such as: data flow and state transition diagrams, temporal logic technique, Petri-net classes, abstract communicating methods.

The aim of this study is to analyse integration possibilities of the semantic model for structural information representation [5] and the aggregate approach [8] for functional representation of processes in the requirements specification. Such a type of multimodal transportation can be organised much more efficiently using the possibilities of information system and simulation results. It is of importance that the early stage of software development, called as requirements engineering, should play a significant role in development of an accurate system with the required level of quality. A formal specification based on piecewise-linear aggregates (PLA) [9] makes it possible to represent relationships between processes.

2. Peculiarities of the Multimodal Transportation Technology

The multimodal transportation may be considered as a dynamically changing and complex technology. Decision making is performed considering a lot of various factors: evaluating the technical infrastructure of multimodal transportation and organisational aspects, comparing reports with the real situation.

Our consideration is focused on the representation of several features of multimodal transportation technology:

- a cargo being carried by the same transport unit using several different transport kinds;
- a long conveyance cycle and international character of transportation;
- complex relationships between separate conveyance links, time and geographic dependencies in process relationships;
- multiple subsystems with their own complex mechanisms interacting as internal or external parts;

The results of analysis of attractiveness of the transport system between forwarding agents showed that the most important evaluation criteria are: cost, reliability and time of transportation. The weight of these three factors is varying among different respondents. It is linked to the nature of cargo being carried and depends on special requirements of senders and receivers.

The interaction of objects in road, rail, and water transportation is considered. The structure of transportation objects that can interact in such processes represented by three levels. Each level reflects some closely interrelated functions of the activities. The physical-technical system consists of network of roads, sea routes, and railways; infrastructure of terminals involved in the organisation of reloading. Industrial activities and transportation functions are safeguarded by the transport agents. A system of co-ordination of activities is implemented by a synchronous work of senders, forwarding agents, terminal operators, sea ferry bridges, ships, trains and motor transport drivers.

The objective of our multimodal transport evaluation system under development is to analyse the multimodal transport situation between Western and Eastern Europe and evaluate Lithuania as a transit country.

3. Knowledge Representation Techniques

We have distinguished two levels of knowledge representation: a semantic model of specification of static aspects of the target system and a model of behavioural analysis of the target system based on aggregate approach. The model allows us to represent the domain specificity more expressively and to choose the main, first used

principals and rules. A deep representation of knowledge provided by the semantic model assures an implicit representation of domain specificity and a choice of priorities of the principles and rules provided by specialists-experts of the target area [2,5]. The adequate imitation model of the behavioural analysis allows us to predict further evolution of the target system and to increase the quality of decision making.

3.1. Components of the Semantic Model

Three types of abstractions of the chosen entities (generalisation, decomposition, and transformation) are used in constructing a semantic model of static components. following by [6]. The structure of a concept ties the characteristics of the concept together. It is based on the relationship of *intentional containment*. The intentional representation of concepts following by [12] is defined as follows. Let X_{ext} denote the extension of a concept X_{con} with respect to its extensionality and X_{int} denote the extension of concept X_{con} with respect to its intensionality. Then, defined $(\forall x/X_{ext})(\exists pp/X_{int*})(\forall p/pp)p(x)$ is the intentional representation of the concept X_{con}.

Generalisation abstraction defines the type of intention "*concept-concept*", where one concept of a set of objects is generalised by another. The following expression symbolises the generalisation relationship between concepts: X_j ***IS_A*** X_i.

Decomposition abstraction helps us to construct a concept by other concepts depending on their functional dependence. The next expression will symbolise the decomposition relationship between concepts: X_i ***PART_OF*** X_j

The description of *transformation* is very similar to that of the decomposition, except that it contains a specification of the transformation function, too.

3.2. Aggregate approach

The aggregate approach and its possibilities for specification and analysis of real time systems are presented in [8, 9]. The theoretical basis of the aggregate approach is a piecewise-linear aggregate (PLA) for formal specification of systems. The advantage of the aggregate approach is a permission to create models both for the analysis of correctness of specifications and simulation.

In the aggregate approach a system is represented as a set of interacting PLA. The piecewise-linear aggregates belong to the class-time automaton. The essence of PLA is that the state of an aggregate has a special structure. The state of an aggregate consists of two components: a discrete state component taking a countable set of values; and a continuous component with co-ordinates. The state of an aggregate can change in two cases only: when an input signal arrives at the aggregate or when the continuous component acquires a definite value. The theoretical basis of PLA is their representation as a piecewise-linear Markoff processes.

4. Representation of Multimodal Road Network Topology

Each level of the information system has to ensure a certain operativity of information. The first level of information consists of the data on the route network technical characteristics: lengths and the technical state of routes, limits of route permeability and speed, etc. Conceptual schemes of topology for the road network are proposed using the constructions of the semantic model. The example of structure of route network technical characteristic is shown in Fig. 4.1.

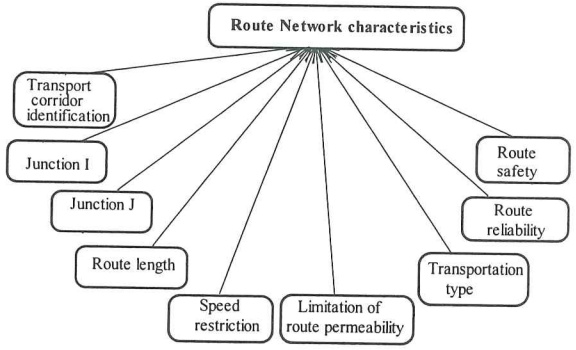


Fig. 4.1. The conceptual scheme of road network technical characteristics.

Technical characteristics of the multimodal transportation system combine freightage junctions such as: border crossing points, main cities, ports, etc. These intersection points are described as 'junction-nodes' in Fig. 4.2. The junctions of multimodal transportation (i.e. reloading terminals) can ensure delivery of freight by certain means of transportation, reloading of freight into another means of transportation and its forwarding to another transport junction-node.

The evaluation and selection of route also depend on the type of loads and on the desirable duration of transportation. The price and reliability of transportation play an important role in selecting the route.

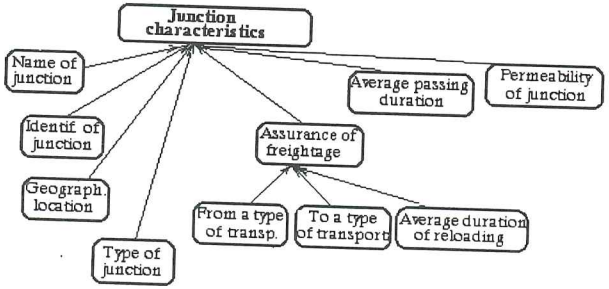


Fig. 4.2. The conceptual scheme of multimodal freightage junction characteristics.

The data accumulated in the information system should help to determine the technical state and reliability of routes, transportation duration, etc. The reliability of the route is a complex evaluation and it should reflect assurance of load safety, possibility of assault, assurance of freight delivery within the limits of fixed terms.

5. The Aggregate Model of Multimodal Transportation

The whole modelling system of multimodal transportation was divided into three subsystems: "Environment", "Chain" and "Node". Functioning of these subsystems is formally described by PLA. The example of structural scheme of the aggregate model of the route 'Paris - Moscow' is shown in Fig. 5.1.

The subsystem "Environment" represents the real environment of multimodal

transportation. It has two output channels with the first and the last "Node" subsystem, through which the "Environment" passes the output signal about the order to start goods transportation accordingly in the West-East or East-West direction.

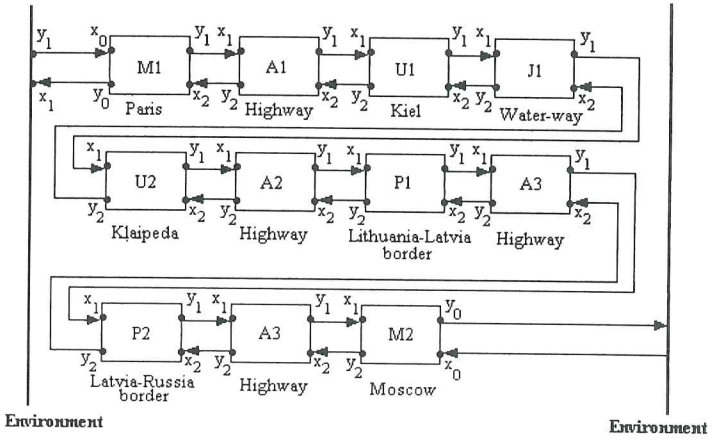


Fig. 5.1. The example of aggregate scheme of the multimodal transport route 'Paris-Moscow'.

Also, the "Environment" has to incoming channels with the first and the last "Node" subsystem, through which the "Environment" gets the incoming signal about the complete transportation of certain goods. The "Environment" has a counter that indicates how many transportation cycles were carried out in the West-East or East-West direction.

The subsystem "Chain" is dedicated to the model of a certain chain of transportation routes. When the goods arrive at the "Chain", they have either to wait or not for the next transport means time schedule. The data on chains of the route are the goods code and the list of route chains. Modelling characteristics of the model are certain route transportation time, price, and confident (i.e. how long the goods can maximally be late with respect to the medium transportation time).

The subsystem "Node" is dedicated to model a certain junction of the transportation route, i.e. a port, a city, a border crossing, etc. Each "Node" subsystem has outcoming channels with "Chain" subsystems to the right and to the left side through which the "Node" passes the outcoming signal about the finished goods

transportation through the "Node" (goods loading, warehousing, customs). Also, each "Node" subsystem has incoming channels with "Chain" subsystems to the right and to the left side (except the first and the last "Nodes" which have channels with the "Environment"). The "Node" gets the incoming signal about the complete transportation of goods through the "Chain" (waiting for a transport time schedule, stops for the driver relaxation).

The formal description of the aggregate "Node" is follows:

1. The set of input signals: $X = \{x_0, x_1, x_2\}$, where x_0 is a signal stating that a load order for transportation has arrived to a node; x_j is a signal stating that the load has arrived to a node from one or another direction (destination/source); j=1,2; $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4})$; x_{i1} is a code of the load; x_{i2} is a time moment when the load has arrived to the route; x_{i3} is a time interval during which the load is transported through the route; x_{i4} is transportation expenses (evaluating a passing part of the route).

2. The set of output signals: $Y=\{y_0, y_1, y_2\}$, where y_0 is a signal stating that the transportation of load has finished through this route (only for source or destination nodes); y_j is a signal stating that load has left a node to one or another direction (destination or source); j=1,2; $y_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4})$, where: y_{i1} is the code of the load; y_{i2} is a time moment when the load has arrived to the system; y_{i3} is a time interval during which load is transported through route; y_{i4} is a transportation cost evaluating passing part of the route.

3. The set of external events: $E'=\{e'_0, e'_1, e'_2\}$, where e'_0 implies the arrival of the signal x_0; e'_j implies the arrival of the signal x_j; j=1,2.

4. The set of internal events: $E''=\{e''_0, e''_1, e''_2\}$, where e''_0 - transportation of load has finished; e''_j - service of load in node has finished and load has left the node; j=1,2.

5. Controlling sequences: $e_i'' \Rightarrow \left\{\mu_j\right\}_{j=1,}^{\text{Constant}} \; i = 1,2;$ where μ_j is duration of service in a node; j corresponds to direction (forward/backward).

6. Aggregate state: $Z_v(t_m) = \{W(t_m), \sqrt{}(t_m)\}$, where $W(t_m) = \{w(e''_1, t_m), w(e''_2, t_m)\}$ is a continuous aggregate state component; $w(e''_i, t_m)$ is a time instant when the event e''_i has to occur; i=1, 2; $\sqrt{}(t_m) = \{\nu_1, \nu_2(t_m)\}$ is discrete aggregate state component;

ν_1 is expenses of load transportation in analysed node; $\nu_2(t_m) = \{0, 1\}$; $\nu_2(t_m) = 0$, if a node has no load; $\nu_2(t_m) = 1$, if node has a load that is served.

7. Initial state: $w(e''_i, t_0) := \infty$, i=1, 2; $t_0 = 0$; $\nu_2(t_0) := 0$.

8. System behaviour:

$H(e'_i)$:i = 0, 1, 2; j = 1, …, Constant;

$w(e''_i, t_{m+1}) := t_m + \mu_j$; $\nu_2(t_{m+1}) := 1$; $t_{tr} := x_{i3} + \mu_j$; $k_{tr} := x_{i4} + \nu_1$;

STD(μ_j);

$G(e'_i)$:i = 0, 1, 2; $Y := \varnothing$;

$H(e''_i)$: i = 0, 1, 2;

$w(e''_i, t_{m+1}) := \infty$; $\nu_2(t_{m+1}) := 0$;

$G(e''_i)$:i = 0, 1, 2;

$y_i := (y_{i1}, y_{i2}, t_{tr}, k_{tr})$.

6. Evaluation Results of Multimodal Transportation Routes

Data for the model were collected from two sources. The first source is the Finnish scientists' report [14]. The second source was data from the Lithuanian transportation companies. Five competitive routes were selected:

- **The Baltic route** is the water-way through the Baltic sea from the main European cities to one of the three independent Baltic states;
- **The land-way through Belarus** is the railway or the highway that crosses at least three borders: Germany-Poland, Poland-Belarus, and Belarus-Russia, reloading the goods at Belarusian border because of different European and Russian rail track width;
- **The Finland route** is the water-way through Baltic sea from the main European cities to Helsinki and further it's usually the railway to Russia;
- **The Klaipeda route** is the water-way through the Baltic sea from the main European cities to Klaipeda and further, it's the railway or the highway to Russia.
- **The land-way through Lithuania** is the railway or the highway that crosses the following borders: Germany-Poland, Poland-Lithuania, Lithuania-Belarus, and Belarus-Russia when going to Moscow, or Germany-Poland, Poland-Lithuania, Lithuania-Latvia, and Latvia-Russia when going to St. Petersburg.

Our mathematical model was used for the comparative analysis of competitive routes between Western Europe and Russia. Two cycles of modelling were performed. The simulation model evaluates the time schedules of transport means and the fact, that the transportation time duration via a certain chain or the time duration at the borders are random variables with the known law of distribution.

An experiment was carried out to evaluate possibilities of improving the situation of the Baltic route via the Klaipeda port. For this purpose the prices of Klaipeda port services were reduced by 5%; duration of every Klaipeda port service was reduced by 5%. The percentile differences of competitive routes by evaluating criteria are given in Table 6.1. and Table 6.2.

Percentage differences of concurrence routes by evaluating criteria (modelling cycle I)

Table 6.1.

The name of the route	Criteria		
	Price	Time	Reliability
Finland	100%	87.73% (12.27%)	67.61% (32.39%)
Baltic	72.87% (27.13%)	85% (15%)	67.61% (32.39%)
Land-way	70.04% (29.96%)	100%	100%

The percentile difference was calculated, where the last place was stated as 100% according to some evaluation criteria and the percentile difference is given in practice, to show how much such a competitive route is better than the worse route.

Percentage differences of routes after reducing times of border crossing (modelling cycle II)

Table 6.2.

The name of the route	Criteria		
	Price	Time	Reliability
Finland	100%	50% (50%)	71.43% (28.57%)
Klaipeda	53.33% (46.67%)	31.25% (68.75%)	50% (50%)
Land-way via Lithuania	26.67% (73.33%)	68.75% (21.25%)	64.29% (35.71%)
Land-way via Belarus	86.67% (13.33%)	100%	100%

The situation of Lithuania in the multimodal transportation market has been defined and also several offers how to improve this situation have been made, according to the modelling characteristics, which were gained in simulating a certain situation (for example, to increase the effectiveness of work at the Klaipeda port or to increase the penetrability of border posts).

Conclusions

The approach for representing static and dynamic aspects of a decision support system and reflecting them using knowledge representation of multiple objective decision making has been developed.

The integration possibilities of two knowledge representation techniques: semantic model constructions and piecewise-linear aggregates are considered. This integration allows us to construct a compatible scheme of specifying the information system for the evaluation of dynamic environment. The model based on the PLA aggregate approach makes it possible: to express the communication between informational objects of dynamic environment; to validate the semantic model of static data structures, and relate the use and analysis of information structures with the processes modelling.

The created simulation model of multimodal transportation enables us to evaluate competitive features of a specified route between Western and Eastern directions of Europe, and also to establish the position of the Lithuanian multimodal transportation.

References

1. Brunet J., Cauvet C., Lasoudris L. Why using events in high-level specification. In H.Kangassalo (Ed.) Entity-relationship Approach: the Core of Conceptual Modelling. North-Holland. 1991, pp. 211-223.

2. Dzemydiene D. On Solving the Problem of Representation of Static and Dynamic Aspects of the Information System. In P. Rumsas, G. Masiuliene (Eds.) *Computer Programming*. Vilnius 1986. Issue 10, 1986, pp. 70-88 (in Russian).

3. Dzemydiene D. Representation of Decision Making Processes for the Ecological Evaluation System. *Annals of Operation Research*. 1994.Vol.51, pp. 349-366.

4. Dzemydiene D., Pranevicius H. Description of a Dynamically Changing Environment in a Decision Support System. In Proc. of the Baltic Workshop on National Infrastructure Databases: Problems, Methods, Experiences. Vilnius. 1994. Vol. 2, pp. 102-111.

5. Dzemydiene D. Application of E-nets in Conceptual Modelling of Dynamic Environment for Decision Support Information Systems. Abstract of the Thesis for Doctor Degree. Vilnius, MII. 1995, pp. -24.

6. Kangassalo H. On the Concept of Concept in Conceptual Schema. In H. Kangassalo (Ed.) First Scandinavian Research Seminar on Information Modelling and Data Base Management. Tampere. 1982. Ser. B, Vol. 17, pp. 129-172.

7. Kappel G., Schrefl M. A Behaviour Integrated Entity Relationship Approach for the Design of Object-Oriented Databases. In C.Batani (Ed.) Proc. of 7-th Int. Conf. on ER Approach. North-Holland, 1989, pp. 311-328.

8. Pranevicius H. Aggregate approach for specification, validation, simulation and implementation of computer network protocols. In "Lectures Notes in Computer Sciences". No 502. Springer-Verlag. Berlin. 1991, pp. 433-477.

9. Pranevicius H. Specification and Analysis of Real-Time Systems by Means of Aggregate Approach. In Proc. of the 3rd Baltic Summer School on "Information technology and System Engineering". Kaunas. Technologija. 1995, pp. 60-87.

10. Pernici B. Objects with Roles. In Proc. of the ACM/IEEE Conf. "Office Information System", Boston , 1990.

11. Rolland C. Event Driven Synchronisation in REMORA. In H. Kangassalo (Ed.) Third Scandinavian Research Seminar on Information Modelling and Data Base Management. Tampere. 1984. Ser. B. Vol. 22, pp. 245-275.

12. Schreft M. Behaviour Modelling by Stepwise Refining Behaviour Diagrams. In H.Kangassalo (Ed.) Entity-Relationship Approach: the Core of Conceptual Modelling. North-Holland. 1991, pp. 119-134.

13. TEMPORA Esprit 2 Project. Concepts Manual. 1993.

14. Transport Routes between Western Europe and Russia. VTT Communities and Infrastructure Transport Research. NEA Transport Research and Training. Espoo. 1994.

THE EXTENSION OF STRUCTURAL MODELLING APPROACH FOR PROCEDURAL KNOWLEDGE REPRESENTATION

Janis Grundspenkis
Department of Systems Theory and Design
Riga Technical University
1 Kalku Street, LV 1658 Riga, Latvia
E-mail:jgrun@itl.rtu.lv

ABSTRACT

The aim of structural modelling is to support a systematic and causal domain model driven knowledge acquisition process. This process results with the development of the knowledge base which consists from two parts: a topological knowledge base and a deep knowledge rule base. Both parts capture only the static view about a given system. The objective of this study is the integration of the knowledge base used so far with methods based on the well known object oriented approach of the frame-based system design. The basic principles of the extension of the frame structure using facet methods is discussed. Temporal relations are defined and added to event trees which are derived from the model of a functional structure.

Keywords: knowledge representation, structural modelling, temporal relations, hybrid knowledge-based systems.

1. INTRODUCTION

Structural modelling may be considered as a tool for causal domain model driven knowledge acquisition to support a construction of intelligent diagnosis systems [1]. The main application domain of structural modelling is complex technical systems with heterogeneous physical elements, for example, carburated gasoline engines or high pressure blowers.

The essenc of structural modelling is the systematic procedure for the construction of three kinds of structural models reflecting the morphology, functions and behavior of the given system. One of the most relevant advantages of this approach is the ability to automate the knowledge base building, i.e., it includes formal procedures both for the construction of functional models from the morphological structure, and for the derivation of deep knowledge rules as well [2].

Up to this moment structural modelling helps to create only a static view of a system under investigation. To overcome this drawback we need to encapsulate the procedural knowledge into the knowledge base. The aim of this study is to integrate the static frame based knowledge representation scheme used in structural modeling with the facet methods of the object oriented approach. The result is a hybrid knowledge based system in which a frame hierarchy is used together with rules. The addition of a temporal dimension to functional structures is proposed as an important step towards the implementation of temporal reasoning in expert diagnosis systems.

This paper has three main sections, the introduction and the conclusion. The introduction points to advantages and drawbacks of the current state of the art in the structural modelling field. The architecture of the knowledge base used to represent a static knowledge is described in the second section. The extension of the knowledge base to represent a dynamic (procedural) knowledge is discussed in the third section. The fourth section is devoted to the addition of the temporal dimension to event trees derived from the model of functional structure in the space of parameters. Temporal relations of connected and disconnected events are proposed in this section too. In the conclusion some aspects of ongoing research and future plans are described.

2. THE ARCHITECTURE OF THE KNOWLEDGE BASE IN STRUCTURAL MODELLING

The abstract causal domain model built within the framework of structural modelling consists from three models, namely, the model of morphological structure (MSM) and two kinds of functional structures. The transformation of the MSM into the functional model in a space of functions (FSM FS) as well as the transformation of the FSM FS into the functional model in a space of parameters (FSM PS) are executed automatically. The representation language used throughout the structural modelling framework is the same for all models.

The frame hierarchy is chosen as the most appropriate for the diagnosis problem domain, and all three structures are visualized as digraphs [1]. Figure 1 shows two object frames from the morphological structure of engine's cooling system shown in Figure 2a. Figure 2b represents the corresponding model of the functional structure in a space of functions (FSM FS) where the arcs represent causal relations between given functions.

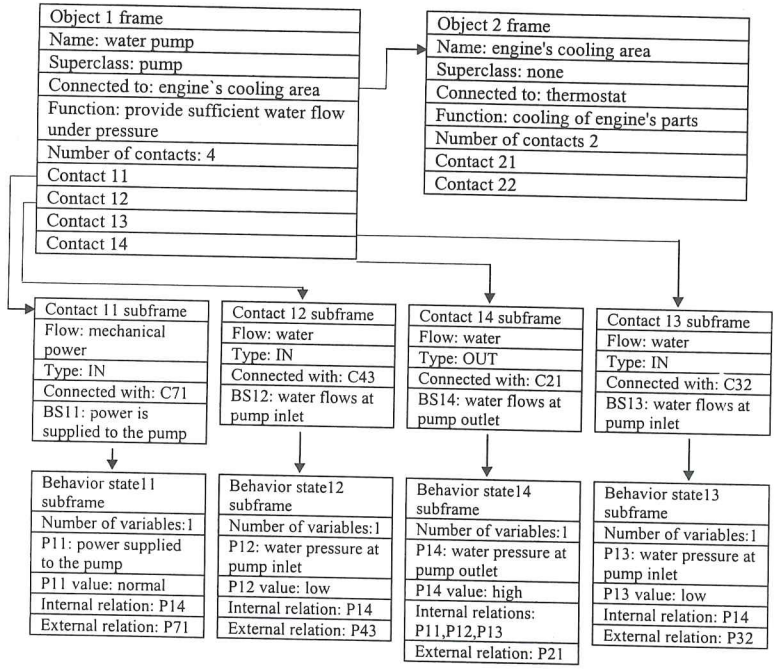


Figure 1. A fragment of a frame hierarchy for knowledge representation .

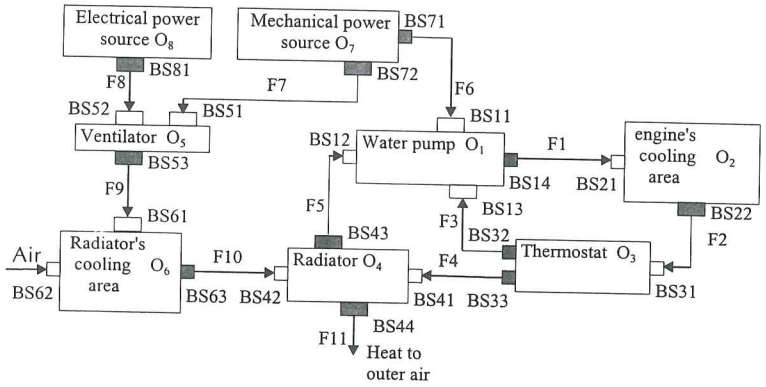


Figure 2a. A representation of the morphological structure of a cooling system.

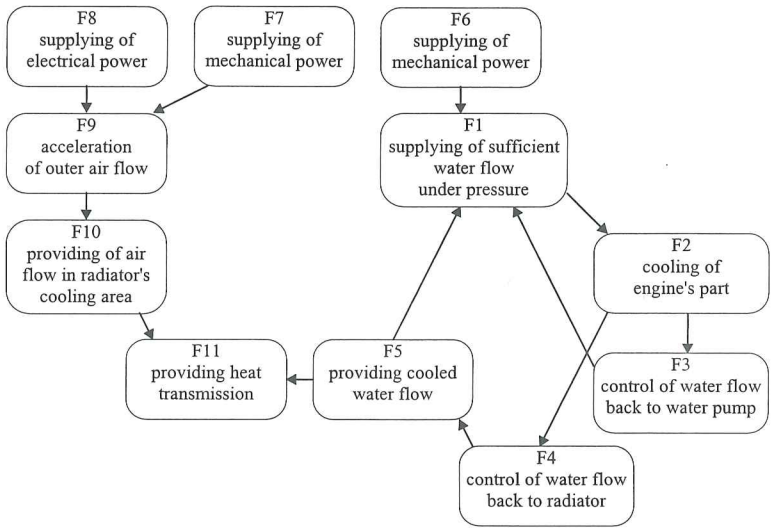


Figure 2b. A representation of the functional structure in a space of functions.

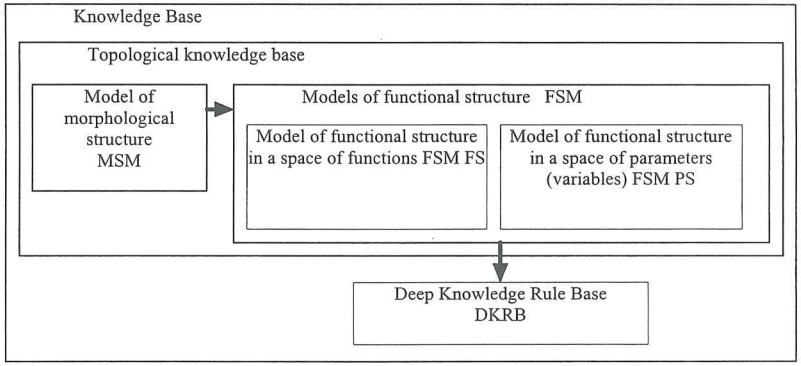


Figure 3. The structure of the knowledge base used in the structural modeling

The examination of the contents of the frames helps to determine the relations between all three model types and to interpret their semantics. Notice that "morphological structure" represents the physical formation of the system, while notions of "space of functions" and "space of parameters" are used to represent the expert's knowledge of how the component actually performs. It is worth to stress that models of functional structure, namely, the FSM FS and the FSM PS are isomorphic, and the only

difference between them are the semantics. That is why the visualized example of the FSM PS is omitted because it is easy to replace the name of the function in Figure 2b by the corresponding name of the parameter from the behavior state subframe. More details are in the third section of this paper, and also may be found in [1] and [2].

The acquired knowledge about morphological and functional structures is captured in a topological knowledge base (TKB). The TKB is used to obtain a deep knowledge rule base (DKRB) which represents the knowledge of how a system under diagnosis actually works. To achieve the transformation from TKB to DKRB the model of functional structure in a parameter space (FSM PS) is decomposed to substructures, i.e., event trees [1]. Each event tree reflects the cause-consequence relations between possible faults and changes of variables (parameter values). Cause-consequence relations that are captured in each event tree can be easily transformed into a set of rules. The set of cause-consequence (C-C) rules that compiles the DKRB is a deep causal domain model of the system under diagnosis.

The high level structure of the knowledge base used in the structural modelling is displayed in Figure 3.

3. THE EXTENSION OF THE KNOWLEDGE BASE

The information gathered in the topological knowledge base provides users with characteristics of the system's objects, but at this point we have only a static view of a given system. We need to study the diagnosis problem further to see how we can bring this information to life. The analysis of the set of possible faults gives us a primary information to extend the frame hierarchy.

The specific feature of systems with physically heterogeneous elements (except those with redundant elements) is that these systems usually have not similar types of objects, i.e., each component is unique in sense of its construction and purpose of its use in the system. Therefore, a definition of classes have different aims in comparison with the object-oriented approach to the frame-based system design. In structural modelling classes may play three roles: 1) common role to the class is to describe general objects and their interconnections with specific objects that define ''kind of'' relationship (generalization); 2) classes may be used as identifiers in a repository that captures a set of previously built structural models of similar objects (all similar objects belong to one

class, for example, water pump, oil pump etc. belong to the subclass PUMPS); 3) classes may be used to define graphical objects - templates of frames and subframes, i.e. the user's interface allows to generate empty shells for all types of used frames.

Now let us consider how to define events that may happen in a given system. A model of a functional structure in a space of parameters (FSM PS) is used to solve diagnosis tasks based on the structural modelling approach. When the FSM PS describes a normal operation of real technical system, its nodes represent variables (parameter values) and arcs correspond to cause-consequence relations between variables. When used to support diagnosis, the FSM PS is extended by adding the set of possible (observed) faults [1]. A fault is defined as a structural change of a component. For example, a leakage in a radiator, that is, a fault of the radiator is caused by a structural change, namely, the appearance of holes in it.

Obviously, if the given system has a fault, then we can observe changes in its behavior, i.e., the system changes its state. There may be at least three different kinds of causes why the system's behavior has changed. Firstly, a behavior state of an element may change. As a result, the normal relationship between input and output of an element is interrupted. For example, if there is not a water flow at the inlet of the water pump (BS12 or BS13 has been changed in Figure 1), then there is not any action of the pump (no pumping), and as a consequence, there will not be a water flow at the outlet of the pump. In fact, the cause-consequence chain is interrupted due to the fault of some ancestor component. In the case of water pump this situation will occur if the cooling system will lose water. It may happen for several reasons: the engine's casing has a hole, the piping has bad connections etc. Secondly, there may be a fault, i.e., a structural change of an element. For example, there is not a fuel flow at the outlet of the fuel pump because the membrane is destroyed. Again the cause-consequence chain is interrupted but the component itself is ''responsible'' for this because it has the fault. At last, there may be a wide range of deflections of parameter values caused by faults which are more or less incipient. The latest situation is the most difficult and, at the same time, the most interesting diagnosis problem. In this case faults become time dependent and during some interval of time faults ''grow up'' from incipient to more and more serious. Let us consider two examples. If the bearing of the pump is damaged, then relatively quickly we can observe that the pump bearing is warm and the pump does not supply sufficient water pressure. As a different case we can imagine that at the very beginning the radiator

have a very small halls and a leakage of water is practically unobservable. It will take relatively long time before we can detect this type of a fault.

To represent the set of possible faults we need to define a new subframes, namely, fault subframes. The extension of behavior state subframes is also needed to model the propagation of consequences of fault appearance in the system. The frame-based expert system design combined with the object-oriented programming gives several techniques to accomplish this task.

There are two ways to enable objects to communicate with one another [3]. The first technique relies on IF-NEEDED or IF-CHANGED facets. The second technique involves messages being sent between objects. Both techniques rely on procedures (methods) being written and attached to the frame. Thus, declarative descriptions of objects captured in the TKB may be extended as follows.

A fault subframe have the following slots: names, which are the identifiers of corresponding faults, connections with parameters and time which reflects the moment (the time point) when certain fault appear, or time interval, while a fault exists in the system. A subframe for the fault ''damage of water pump bearing" is shown in Figure 4a. Using facet approach, the written method is attached to the IF-NEEDED facet of a given fault subframe's property ''time''. The IF-NEEDED facet method is executed whenever the property is needed. Consider for example that we want to predict ''what and when happens if there is a certain fault in the system''. To accomplish this type of predictive reasoning, first of all we need the information about ''the starting time of the fault'', as well as about ''the time interval of fault existence''.

Now let us suppose that we have defined the relations between corresponding object parameters in our knowledge base, and that we also have the information about propagation time of consequences caused by certain fault, i.e., we have time intervals between the moments when parameter value changes take place in the chain connecting the fault and symptoms to be observed.

So, using forward chaining, it is possible to calculate, for example, when we may expect to observe that heating system in our car does not operate efficiently enough and that ventilator is turned on more frequently as usual. These, in turn, may be considered as symptoms that the water level in our car cooling system is below the safety level. We shall discuss the formal aspects of calculations in the next section.

Figure 4b shows the subframe of the fault: ''damage of water pump bearing" with

a)

Fault subframe
Name: damage of water pump bearing
Connected with: P14 water pressure at pump outlet
Time: unknown

b)

Fault subframe	
Name: damage of water pump bearing	
Connected with: P14 water pressure at pump outlet	Time IF-NEEDED method
Time: unknown	GET TIME (damage of water pump bearing)

Figure 4. An example of a fault subframe.

IF-NEEDED method added and written in a pseudocode. In this example we suppose that the IF-NEEDED method will ask the user to enter a time value.

In diagnosis we have the opposite situation, i.e., we can fix time when certain symptom appears and then try to estimate ''why the behavior of the system changes'', as well as ''when the cause of a faulty behavior may happen''. We hope that future studies of the latest problem may lead to the solution of detecting incipient faults, or more generally, will allow to separate various degrees of the same fault. As a consequence, that may bring us to better understanding of what really has happened in the system, and to predict the state of the system in the given time horizon.

Thus, in diagnosis systems we need to implement a backward chaining based on relationships between objects (frames) representing parameters in the FSM PS. It is clear, that existence of a fault will entail series of parameter value changes, and our system must be supplied with ability to reason about this issue. In fact, we need to add more details to the variable (parameter) slots of behavior state subframe.

First of all, the expert must evaluate causal relationships between parameter values. The evaluation process starts from a primary event, i.e., from a fault, and has two stages. At the first stage the expert must answer the question: ''if there is a fault, the corresponding parameter value increases or decreases''. For our example displayed in Figure 4 the expert will be asked the question: ''if there is a damage of the water pump bearing, the water pressure at the pump outlet decreases or increases?'' The relationship between the fault and the parameter value is evaluated by introducing a label, namely, the sign ''+'' if the fault will cause the increase of the parameter value, and ''-'' if the

fault will cause the decrease of the parameter value. The second stage is foreseen to get more information about the same relationship. The expert is asked to add a quantitative estimation of the relationship. At the present moment in structural modelling we use only one kind of a quantitative space, namely, [-10,-1,0,+1,+10] where ±1 denotes moderate changes and ±10 denotes significant changes of corresponding parameter values [2]. The research is going to find effective procedures for the evaluation based on fuzzy logic.

To implement the proposed evaluation we need to add a new property - ''caused change'' - to the fault subframe. We also need to define IF-NEEDED and IF-CHANGED facets. As in previous situation, the IF-NEEDED method will be used to ask the expert to define needed values of parameter changes. Then the IF-CHANGED method will transfer these values to the corresponding parameter value property in the behavior state subframe.

Let us return to our example. The fault subframe displayed in Figure 4b is extended as follows (see Figure 5).

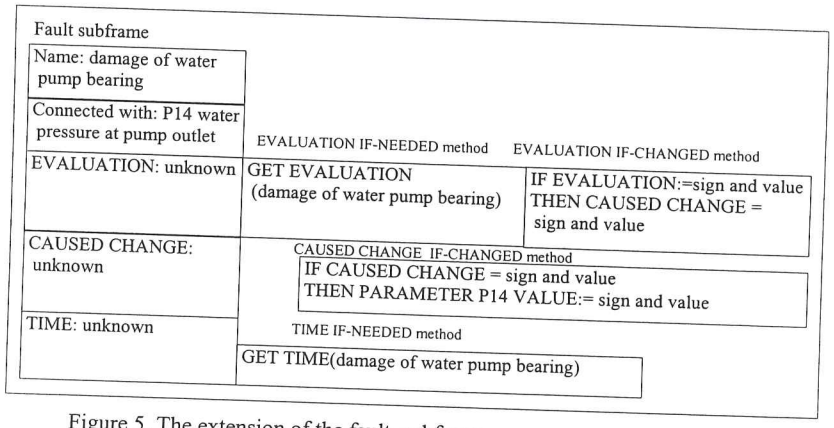


Figure 5. The extension of the fault subframe.

To get the evaluation of the relationship ''damage of water pump bearing → P14 water pressure at the pump outlet" we have IF-NEEDED method. This method executes the function ''GET EVALUATION'' which will call the corresponding procedure, i.e., the dialogue with expert.

This function will return the sign and the value of the relationship and will bind them to the value of the property ''CAUSED CHANGE''. In our example the returned result will be ''-10'' because damage of the water pump bearing will significantly decrease the water pressure at the pump outlet. This, in turn, will bind returned sign and value to the parameter value in the linked behavior state subframe, i.e., the property ''P14 value'' will be changed from ''high'' to ''very low'' (or ''zero''). This new binding is defined by the mapping of quantitative value scale [-10,-1,0,+1,+10] to the set of linguistic variables.

The propagation of parameter value changes may be modeled if we have the IF-CHANGED method in each behavior state subframe. For example of the cooling system (see Figure 2a), a very low water pressure at the outlet of the water pump will cause a very low water pressure at the engines cooling area inlet, and at the engines cooling area outlet. Finally, the engine will be overheated. To model the cause-consequence chain, the described above procedure of the evaluation of parameter values and addition of appropriate IF-NEEDED and IF-CHANGED methods must be completed.

4. THE TIME DIMENSION AND THE TOPOLOGICAL KNOWLEDGE BASE

It is clear that if we want to get deep enough knowledge about processes taking place in a system when it has faults (at least one) we must consider a failure propagation in our system. In this case time plays a fundamental role because we need to represent not only a static knowledge (a snapshot of the system) but we need also to represent a dynamic knowledge. A lot of work has been done to create various models of time and to implement these models in the temporal data bases or to use them to support temporal reasoning (see, for example [4,5,6,7, 8]). The notion of "event" has an important role in all models of time [7]. In structural modelling the notion of "event" is used as a synonym of the change of parameter value. The background of introducing time in structural models is as follows [1]. As described in previous section, the FSM PS reflects causal relations between variables (parameter values), i.e. it reflects successions of events. Now let us consider these events in more details. Three types of events are

considered: a final event, a primary event and an intermediate event. A final event is equal with the observed (measured) change of a parameter value. A cause of a final event is a structural change, that is, a fault of the corresponding component. The structural change is called a primary event. Each primary event causes one or more series of intermediate events. Thereby, by identifying the succession (chain) of events that causes a definite final event it is possible to explain the cause of this particular final event. A causal ordering of two events is interpreted as a causal relationship or a cause-consequence relation. Causal event chains allow to generate a tree structure from causal relationships. Structural modelling supports formal procedures of decomposition of the FSM PS. A set of event trees is the result of decomposition of the extended FSM PS where for each fixed final event a set of primary events is found. From the discussion above it is evident that in structural modelling we can use event trees to support the reasoning in time.

Now let us discuss basic principles of adding the temporal dimension to the event trees. All examples used in further discussions are borrowed from the MSc. thesis of Zhanis Vushkans supervised by the author of this paper [9]. For better understanding of temporal relations and their representation we have chosen abstract event trees which are not connected with real examples discussed in previous sections.

Temporal relations between events are defined as follows:

n_i TO n_j where n_i and n_j are events, and TO is a time operator.

We define two categories of temporal relations:

1) n_i <<.TO.>> n_j is a temporal relation when both events n_i and n_j belong to the same event tree, and n_j is reachable from n_i . These events are called "connected events".

2) n_i <. TO .> n_j is a temporal relation when both events n_i and n_j belong to the same event tree, and n_j is not reachable from n_i or vice versa. These events are called "disconnected events".

Figure 6 displays an abstract event tree. We can see that there are not direct connections between primary events, i.e. these events are disconnected. Primary events are connected with intermediate events or they may be connected directly with final events. Intermediate events are connected with other intermediate events or intermediate events are connected with final events. It is worth to stress that in this study we do not consider temporal relations in situations when one primary event causes another primary event or when an intermediate event causes some primary event.

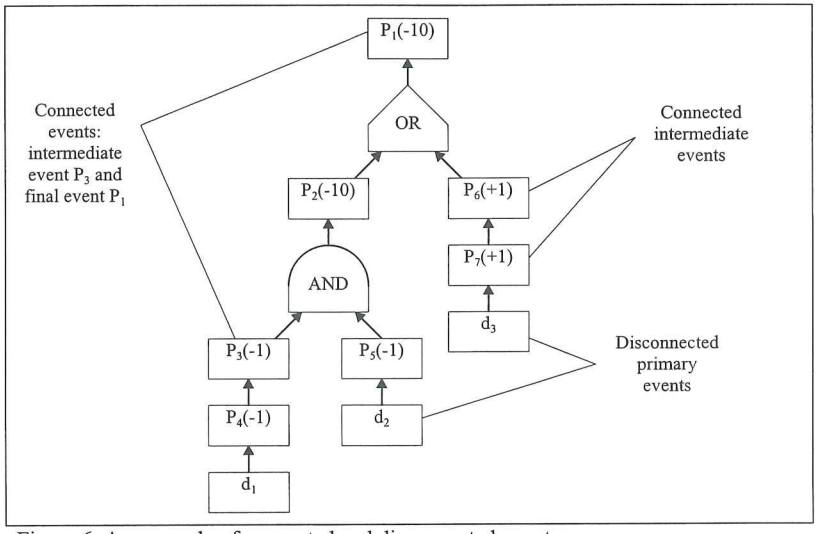


Figure 6. An example of connected and disconnected events.

In general, temporal relations between events may be metrical, nonmetrical and periodical. There may be two kinds of events, namely, momentary (point like) and interval events. In this paper we shall consider only point like events. More details and further discussion may be found in [9].

The kinds of temporal relations between connected point like events are as follows:

1. A nonmetrical temporal relation n_i<<. .>>n_j describes that an event n_i occurs before an event n_j happens The nonmetrical temporal relations are widely used in the case of structural modelling That is because these are qualitative relations between events, and structural modelling first of all is the qualitative approach to systems analysis.

2. A metrical temporal relation n_i<<. t*.>> describes that an event n_i occurs at a time moment t*.

3. A metrical temporal relation n_i<<. m .>>n_j describes that an event n_i occurs m time units before an event n_j happens.

4. A periodical temporal relation n_i<<. dT .>>n_i describes that an event n_i occurs periodically after dT time units.

Two kinds of nonmetrical, five kinds of metrical and one periodical temporal relation may be defined for the connected interval events [9].

For the disconnected point like events there may be the following temporal relations:

5. A nonmetrical temporal relation $n_i<. <.>n_j$ describes that an event n_i occurs before an event n_j happens.

6. A nonmetrical temporal relation $n_i<. >.>n_j$ describes that an event n_i occurs after an event n_j happens.

7. A nonmetrical temporal relation $n_i<. =.>n_j$ describes that an event n_i occurs simultaneously with an event n_j.

8. A metrical temporal relation $n_i<. <m .>n_j$ describes that an event n_i occurs m time units before an event n_j happens.

9. A metrical temporal relation $n_i<. >m .>n_j$ describes that an event n_i occurs m time units after an event n_j happens.

10. A metrical temporal relation $n_i<. t^*.>$ describes that an event n_i occurs at a time moment t*.

11. A periodical temporal relation $n_i<. dT .>n_i$ describes that an event n_i occurs periodically after dT time units.

It is easy to see that the first and the fifth, the second and the eighth, the third and the tenth, as well as the fourth and the eleventh temporal relations are equivalent.

Eight nonmetrical, four metrical and one periodical temporal relation may be defined between disconnected events. Thus the total number of temporal relations used in structural modelling is 32 [9]. In the event trees considered in structural modelling only a subset of temporal relations are used, namely, for connected events the second and the third temporal relations are used, as well as for disconnected events the seventh, the eighth, the ninth and the tenth temporal relations are used. The problem domain experts are asked to provide the values defined in the given temporal relation when the corresponding IF-NEEDED method is executed. These values are used as labels of arcs as it is shown in Figure 7. The duration of the event is represented by the number put into parentheses and located besides the corresponding event. The labeled event tree is used to calculate the moment when the changes of the parameter value of the final event may be measured (this moment is depicted as μ_s in the Figure 7) if the temporal relations between connected and disconnected events are known. These calculations support the

reasoning about the time relations taking into consideration the appearance of a fault in the system, the failure propagation time and the moment when changes of system behavior may be detected. Two algorithms have been worked out. The first algorithm supports the estimation when systems deviation from normal operation can be observed for each particular fault. The second algorithm has been developed to achieve the same goal but in situations when time relations between multiple faults are known. These algorithms are not discussed in this paper due to its limited scope.

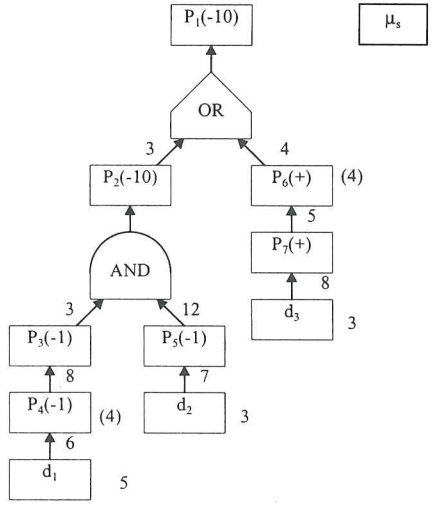


Figure 7. A representation of temporal relations in the event tree.

5. CONCLUSION

In this paper we discuss the basic principles how to capture the procedural knowledge within the structural modelling framework. The integration of structural models with the well known facet methods of the object-oriented approach to the frame-based system development have been proposed. The extended knowledge base includes the frame hierarchy and the rules. Thus, the result of integration is the hybrid knowledge based system. The addition of the temporal dimension to the model of a functional structure in the space of parameters will allow to develop more sophisticated temporal reasoning procedures, and thus to give life to static models used in structural modelling

at this moment. We hope that this will widen the applications of structural models and will improve the explanation facilities of expert diagnosis systems.

The ongoing research is carried out to implement the extended structural modelling schema. Two alternatives are checked, namely, the development of the expert system from scratch using one of the object oriented programming languages, and to integrate structural modelling system with one of the frame based expert system shells. In future the research and implementation of temporal reasoning methods are planned.

REFERENCES

[1.] Grundspenkis J., Causal Domain Model Driven Knowledge Acquisition for Expert Diagnosis System Development. Lecture Notes of the Nordic-Baltic Summer School (K. Wang and H. Pranevicius Eds.), Kaunas University of Technology Press, Kaunas,.Lithuania, 1997, pp. 251 - 268.

[2.] Grundspenkis J., Automation of Knowledge Base Development Using Model Supported Knowledge Acquisition. Proceedings of the Second International Baltic Workshop on Databases and Information Systems, Tallinn, June 12-14, 1996 (Hele-Mai Haav and Bernhard Thalheim Eds.), Vol.1, Institute of Cybernetics, Tallinn, Estonia, 1996, pp. 224 - 233.

[3.] Durkin J., Expert Systems. Design and Development, Macmillan Publishing Company, New York, 1994.

[4.] Even A.,. Temporal Logic. Nonclassical Logic (in Russian), Moscow, Nauka, 1970, pp. 124 - 190.

[5.] Kondrashina E., Litvinceva L,, Pospelov D., Representation of Knowledge about Time and Space in Intelligent Systems (in Russian). Moscow, Nauka, 1989.

[6.] Orchi T. Lecture Notes in Temporal and Deductive Databases, Royal Institute of Technology and Stockholm University, Sweden, 1995.

[7.] Prior A., Past, Present and Future, Oxford, Cleredon, 1967.

[8.] Tichy P. The Logic of Temporal Discourse. Linguistics and Philosophy, N.3,1980, pp. 343 - 369.

[9.] Vushkans Z. Development of Method for Addition of Temporal Dimension to the Functional Model, MSc. Thesis, Riga Technical University, 1997.

USING NATURAL LANGUAGE PROCESSING TOOLS FOR READING TEXTS IN A FOREIGN LANGUAGE

Tiit Roosmaa

Institute of Computer Science, University of Tartu

2 J. Liivi Tartu , EE2400 ESTONIA

Tiit.Roosmaa@ut.ee

Abstract

This paper describes results obtained within the EC Copernicus language technology project GLOSSER. GLOSSER aims to apply natural language processing techniques, especially morphological analysis, dictionary and corpora processing, to technology for computer-assisted language learning. Created by the GLOSSER team the linguistic components (Analyzer, bilingual dictionaries and bilingual corporas) and developed technologies - tested on three languages, can be used in a several information systems, where natural language analyze is needed.

Introduction

The GLOSSER prototype will help many people who know a bit of English but cannot read it quickly or with full understanding. It will support their knowledge of basic grammar (i.e., morphology) and remove the tedious task of thumbing through a dictionary.

GLOSSER's aim has been to implement bilingual dictionary (English-Estonian, English-Hungarian, English-Bulgarian) and bilingual corpora for the same languages as on-line context-sensitive translation support. It presupposes that a user has a text in his word-processor that he wants to read. Clicking on a sequence of words will display