A Consensus Test Suite of Temporal Database Queries*

Christian S. Jensen (editor) James Clifford Shashi K. Gadia Fabio Grandi
Patrick P. Kalua Nick Kline Nikos Lorentzos Yannis Mitsopoulos
Angelo Montanari Sunil S. Nair Elisa Peressi Barbara Pernici
Edward L. Robertson John F. Roddick Nandlal L. Sarda Maria Rita Scalas
Arie Segev Richard T. Snodgrass Abdullah Tansel Paolo Tiberio
Alexander Tuzhilin Gene T. J. Wuu

Abstract

This document presents the temporal database community with an sizable consensus test suite of temporal database queries. The test suite is intended to be helpful when evaluating the user-friendliness of temporal relational query languages.

The test suite consists of a database schema, an instance for the schema, and a set of approximately 150 queries on this database. The queries are classified according to a taxonomy, which is also included in the document.

1 Introduction

The central goal of this document is to provide the temporal database community with an *extensive consensus test suite* for temporal relational query languages that is *independent* of any existing language proposal. The test suite is not related to performance issues, but has a *semantic* focus and is intended to be an aid in evaluating the user-friendliness of temporal query languages. Thus, temporal query languages should ideally be able to express the included queries both conveniently and naturally. However, no definition of user-friendliness is included—this aspect is left to the individual users of the test suite.

The work that lead to this document was initiated in early 1993 when all researchers in temporal databases were invited to participate in creating an unbiased consensus test suite. An electronic mail distribution, `tdbbenchmark@cs.arizona.edu`, was used as the medium for the work on the test suite, and an initial working document ("The TSQL Benchmark") was constructed by a total of 20 researchers. That document was presented at the ARPA/NSF International Workshop on an Infrastructure for Temporal Databases, held in Arlington, TX, June 1993, and was subsequently discussed among the 40 invited temporal-database researchers that attended the workshop. The present document is the result of the initial efforts and the efforts of the workshop participants, and as such it represents a consensus among a large fraction of the temporal database community.

The test suite consists of a database schema, an instance for the schema, and a set of queries on this database. The queries are classified according to a taxonomy, which is also part of the document. As a consequence of the central goal above, no existing temporal data models are used or mentioned. The database schema of the test suite is described using the ER model. The presented ER schema may be mapped to a set of relation schemas that fits a particular data model. With the exception of attributes illustrating user-defined time, the underlying temporal aspects are implicit in the description of the database schema. Of course, specific temporal data models might add explicit temporal attributes. The contents of the relations are described in natural language. The actual queries are also given only in natural language. The taxonomy is independent of any particular temporal query language.

The test suite is not intended as a substitute to other means of studying query languages, such as laboratory experiments with users or orthogonality studies. Rather, the test suite is intended as a complementary addition to the existing repertoire of query language evaluation techniques. It is emphasized that the test suite is not intended to constitute a metric for query language completeness, and as such it is not a substitute for a rigorous *theoretical* study of expressive powers of various temporal query languages. Such

*Correspondence may be directed to the TSQL electronic mail distribution, `tdbbenchmark@cs.arizona.edu`, or to the editor at Aalborg University, Datalogi, Fr. Bajers Vej 7E, DK–9220 Aalborg Ø, Denmark, `csj@iesd.auc.dk`. This document was prepared by multiple contributors. The names, affiliations, and e-mail addresses of the contributors may be found in a separate section at the end of the document.

studies are still needed. While a sizable, or extensive, test suite was purposely constructed to ensure that a wide range of query language design aspects were covered, there is no formal basis for claiming that the list of queries is complete, or comprehensive. No such claim is made! It it emphasized that using the test suite as an advanced, quantitative scoring system for comparing languages makes little sense. Thus, one language is not necessarily superior to another just because one is capable of expressing more queries than the other.

In summary, the test suite may be understood as simply an unbiased list of queries. The queries are intended to aid in evaluating the user-friendliness of individual temporal relational query languages.

The presentation is structured as follows. Below, the intended scope of the test suite is defined. Sections 3, 4, and 5 are structured similarly. Each first presents design criteria, then presents a specific design. Section 3 concerns the database schema. The next section covers the database instance, and Section 5 concerns the classification scheme. The main body of the document is Section 6, which presents, using the classification scheme, approximately 150 queries.

2 Scope

The test suite has been designed to provide a "dense" coverage of a restricted range of queries rather than a "sparse" coverage of wide range of queries. Additional queries that cover more types of queries may be added later. This section characterizes the types of queries that are, and are not, covered by the test suite.

Temporal query languages may loosely be categorized as relational or object-oriented. The test suite was designed with only relational temporal query languages in mind that may be perceived as temporal extensions of the SQL query language.

The intention is to provide a foundation for comparing the descriptive and operational characteristics and capabilities of temporal data models and query languages. The test suite is not aimed at performance comparisons. Properly extended with additional relation schemas and a variety of large instances, a performance benchmark may be constructed from the test suite.

A number of restrictions are imposed on which types of queries are intended to be included in the current test suite, including the following.

- Queries are restricted to valid time only. Transaction-time related queries are not explored.
- Schema evolution and versioning are not considered.
- Incompleteness is not considered.
- Recursive queries are not included.
- General temporal reasoning is beyond the scope of this version of the test suite.
- Queries involving aggregation facilities are not considered.
- Only queries are included—insertions, deletions, and updates are not considered.
- Continuous attributes such as time are not included.
- The querying of data valid in the future is not explored.

These advanced aspects are excluded solely for pragmatic reasons, and the exclusion is not meant to imply in any way that the aspects are not important. The restrictions simply represent an attempt to reduce the size of the initial test suite to manageable proportions. Specifically, the next version of the test suite is intended to include queries that involve aggregation.

3 A Database Schema for the Test Suite

3.1 Criteria

A suitable database schema for a test suite satisfies four criteria.

- The schema should be natural. That is, it should correspond to a reasonable, though possibly greatly simplified, segment of the real world. This both reduces the need to explain the model and enhances the ability to recognize verball pitfalls in the path to the query instances.
- The schema should be simple. This will aid in making the test suite easy to understand. This criterion indicates that a database schema consisting of relatively few relation schemas, with relatively few attributes, is desirable. Additionally, the names of the relations and of the attributes should be short, as they will be referenced repeatedly.

  When an expansion is proposed, the benefits should be carefully compared with the added complexity.

- The schema should allow for comprehensiveness within the chosen scope. Using the schema, it should be possible formulate queries of all the types that appear reasonable.

  This indicates a need for at least two related relation schemas (for natural-join queries).

- A schema that has already been used frequently is preferred over a new schema. This guarantees that many existing queries can be adapted easily to the test suite.

- For clarity, schema and attribute names must start with capital letters.

3.2 The Schema

Rather than defining one particular temporal relational database schema, we define instead a database schema using the ER Model. The advantage of this approach is that the ER schema described here may subsequently be mapped to a specific relational database schema in a way that is appropriate for the particular data model at hand. No single relational version of the ER schema fits all temporal relational data models.

The database schema is defined by the ER schema in Figure 1, containing three entity sets, namely `Emp`, `Skill`, and `Dept`, describing employees, skills, and departments, respectively. The attributes of the entity sets, and their interrelationships, are described next.

Entities in entity set `Emp` are represented by the attributes `Name` and `Salary` which record the names and salaries of employees. In addition, attributes `Gender` and `D-birth` indicate the gender and date of birth of employees. While the name and salary of an employee vary over time, both the gender and the date of birth are assumed to be time-invariant.

Skills, in entity set `Skill`, are represend by a single attribute, `Name`, which records the names of individual skills. The name of a skill is time-invariant. Entity sets `Skill` and `Emp` are connected via an n–m relationship set, `EmpSkills`. The skills of an employee may vary over time. For example, employees are considered to have the skill "driving" only during those interval(s) when they hold valid licenses.

The entity set `Dept` represents departments and is described by the attributes `Name` and `Budget` which record the names and budgets of departments, respectively. While the budget of a department varies over time, the name is assumed to be time-invariant. Employees are associated with departments by means of two relationship sets. First `EmpDept` records which employees work in which departments. This is a time-varying n–1 relationship set. Second `MgrDept`, also time-varying, is a 1–1 relationship set associating those employees that are managers with the departments they manage.

Attributes `Name` (of `Emp`, `Skill`, and `Dept`) are of type `textString`; attribute `Gender` is one of `F` (female) and `M` (male); `Salary` and `Budget` are of type `integer`; and `D-birth` is a user-defined time value which may be compared with valid times.

The entity sets obey the following *snapshot* functional dependencies:

For `Emp`:
- `Name` → `Salary`
- `Name` → `Gender`
- `Name` → `D-birth`

For `Dept`:
- `Name` → `Budget`

Note that `Name` is the primary key of `Emp` (it is the only candidate key). For `Skill`, `Name` is the only attribute and is thus the key. For `Dept`, `Name` is the primary key.

It is emphasized that the notion of key does not capture correspondence between attribute values and the real-world objects they represent. As one consequence, it is possible in this ER schema, e.g., for an employee to change `Name` attribute value over time.

This concludes the description of the ER schema. Next, we exemplify how the ER schema may be mapped to a relational database schema. Figure 2 shows one such possible schema. As for the ER schema, the valid-time aspect of this schema is implicit.

```
Emp = (Id, Name, Salary, Gender, D-birth,
          DeptName)
Skill = (EmpId, Name)
Dept = (Name, Budget, EmpId)
```

Figure 2: Sample Relational Database Schema

Relation `Emp` models the entity set `Emp` and the relationship set `EmpDept`. Attribute `Id` is a time-invariant key (i.e., values of this attribute identify employees). Relation `Skill` models the relationship set `EmpSkills` and the entity set `Skill`. Values of attribute `EmpId` in `Skill`, as well as in `Dept`, identify employees. Finally, relation `Dept` models relationship set `MgrDept` and entity set `Dept`.

In this design, `Name` is the snapshot primary key of `Emp` (it is the only candidate key). For `Skill`, there is no non-trivial key. For `Dept`, each of `Name` and `EmpId`

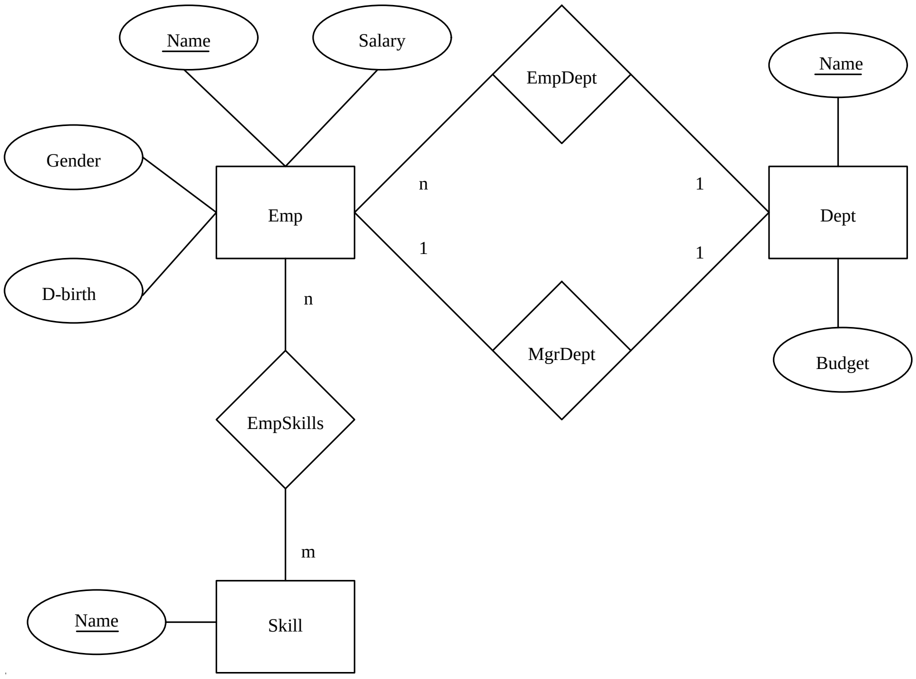

Figure 1: ER Diagram of Database Design

is a candidate key, and `Name` is selected as the primary key.

Each of the relation schemas are in snapshot Boyce-Codd normal form.

4 The Test Suite Data

4.1 Criteria

- For clarity, the database instance should ideally accord with *all and only* those constraints which are explicitly stated in the definition of the database schema.

- For simplicity and ease of typing, attribute values should be short and salary values should be multiples of $10,000.

- Transitions (i.e., timestamp values) occur only at the beginning of the month, and all dates should be in the time interval from 1/1/81 to 12/31/88 (because the digits 8 and 9 are relatively hard to distinguish). Time intervals are all specified by the inclusive starting and ending events. Also for clarity, relation instance names should start with lowercase letters.

- The data should include a "hole in the history" of some entity. For example, the database may be designed to contain a whole in the employment of some employee.

- The data should include asynchronous behavior of attributes. For example, the department and salary of employees may change independently.

4.2 The Data

Three instances, `emp`, `skill`, and `dept`, are defined over the `Emp`, `Skill`, and `Dept` relation schemas, respectively. The contents of these instances is described below.

There are two employees, identified by *ED* and *DI* in the following.

ED worked in the Toy department from 2/1/82 to 1/31/87, and in the Book department from 4/1/87 to the present. From 4/1/87 to the present, he managed the Book department. The budget of that department has been $50K since *ED* became its manager. His name was Ed from 2/1/82 to 12/31/87, and Edward from 1/1/88 to the present. His salary was $20K from 2/1/82 to 5/31/82, then $30K from 6/1/82 to 1/31/85, then $40K from 2/1/85 to 1/31/87 and 4/1/87 to the present. *ED* is male and was born on 7/1/55. Several skills are recorded for *ED*. He has been qualified for typing since 4/1/82 and qualified for filing since 1/1/85. He was qualified for driving from 1/1/82 to 5/1/82 and from 6/1/84 to 5/31/88.

DI worked in and managed the Toy department from 1/1/82 to the present. Her name is Di throughout her employment. The budget of the Toy department was \$150K from 1/1/82 to 7/31/84, \$200K from 8/1/84 to 12/31/86, and \$100K from 1/1/87 to the present. Her salary was \$30K from 1/1/82 to 7/31/84, \$40K from 8/1/84 to 8/31/86, then \$50K from 9/1/86 to the present. *DI* is female and was born on 10/1/60. *DI* has been qualified for directing from 1/1/82 to the present.

The present time (i.e., the value of `now`) is 1/1/90.

5 Classification of Queries

A classification of the test suite queries will be based on a comprehensive taxonomy of queries. First, criteria for such a taxonomy are outlined. Next, the taxonomy itself is presented. As the taxonomy is too fine-grained, categories are then merged into an adequate number of groups which can subsequently be used for classification.

5.1 Criteria

Three criteria for an appropriate taxonomy of queries are suggested.

- The taxonomy should be schema and instance independent. This criterion helps ensure that the taxonomy will persist when the database schema evolves as new versions of the test suite appear. Ideally, this will allow for an incremental mode of work, where only new queries need to be categorized and existing queries do not need recategorization.

- The taxonomy should provide extensive coverage of queries. Extensiveness is desirable to avoid holes and point to many categories of queries.

- The taxonomy should be useful when structuring the presentation of the test suite. Most importantly, it should provide sufficient structure. Thus, taxonomies that have only few categories and that map many queries to single categories are problematic. If the number of categories is excessive for presentation purposes, classes of categories may be identified with individual sections.

5.2 The Taxonomy

The taxonomy is characterized as having a projection (output) and a selection component, much like SQL. Then each component is covered in turn. Finally, the full taxonomy is summarized and a notation for naming individual categories is defined.

5.2.1 Top-level Taxonomy

At the top level, the taxonomy is divided into two orthogonal parts, namely a part where queries are categorized according to their *output component* and a part where the categorization is based on the *selection component*. Thus, a category is described by two components, as illustrated in Figure 3.

$$\{< \text{output component} >\} \times \{< \text{selection component} >\}$$

Figure 3: Top-level Description of Test-suite Taxonomy

This top-level design reflects the SQL template (i.e., `SELECT` ... `FROM` ... `WHERE` ...). The first component categorizes the contents of the `SELECT` clause, and the second component categorizes the contents of the `WHERE` clause. No component is needed to reflect the `FROM` clause where tuple variables are defined. The two components are orthogonal only in the same sense that the `WHERE` and `SELECT` clauses of a particular query are orthogonal.

5.2.2 Output-based Taxonomy

The output-based taxonomy is intended to reflect the part of queries where the format of the resulting tuples is specified. The taxonomy is described in Figure 4 and is explained in the following.

The idea is to distinguish between queries based on the format of the result tuples. A tuple may include an explicit-attribute component and a valid-time component, each of which are considered next.

If present, the explicit-attribute component may contain all attributes in the argument relation (multiple relations are discussed below) or it may contain a subset of the attributes in the argument relation. In the first case, the explicit attribute component is "complete," and in the second, it is "projected."

To exemplify, consider a tuple telling that Ed is in the Book department from 1/1/82 to 12/31/84. Here "Ed" and "Book" constitute the explicit-attribute component, and "1/1/82" and "12/31/84" is the valid-time component. If the argument relation contained

$$\left\{\begin{array}{c}\underline{\text{explicit-attribute component}}\\ \text{none}\\ \text{projected}\\ \text{complete}\end{array}\right\} \times \left\{\begin{array}{c}\underline{\text{valid-time component}}\\ \text{none}\\ \left\{\begin{array}{c}\underline{\text{type}}\\ \text{event}\\ \text{interval}\\ \text{element}\end{array}\right\} \times \left\{\begin{array}{c}\underline{\text{value}}\\ \text{intersection}\\ \text{union}\\ \text{derived}\\ \text{imposed}\end{array}\right\}\end{array}\right\}$$

Figure 4: Output-based Taxonomy

an attribute "Salary" in addition to the EmpName and DeptName attributes, this result is projected.

If several relations are used in a query, the argument relation is the Cartesian product of these, i.e., the schema is the concatenation of the schemas of the relations used in the query.

The valid-time component of a tuple may be of three types. First, it may be an event, i.e., a single time value (e.g., 3/1/83). Second, it may be an interval, i.e., a sequence of consecutive time values (e.g., as above). Third, it may be an element, i.e., a set of time values which may be described by a set of intervals (e.g., 1/1/82 to 12/31/84, 2/1/85 to 3/31/85, and 5/1/86 to 5/31/86).

Orthogonally, the value of a valid-time component may be characterized by "intersection," "union," "derived," or "imposed." A value of a valid-time component of a tuple is characterized by intersection if it is computed by intersecting the valid-time components of argument tuples (cf., intersection join, valid-time natural join, etc.), and if it is computed as the union of argument valid times, it is characterized by union. A derived value is also computed solely in terms of the valid-time components of the tuples in the argument relation, but in some way that is different from intersection and union. An imposed value is computed by explicit assignment in the query.

Note that at least one of the two components must be present in the result of a query. This part of the taxonomy results in 38 mutually exclusive categories.

The distinctions above are based on the schema of result relations. It is possible also to distinguish between the cardinalities of result relations, e.g., between set-valued and single-tuple valued results.

5.2.3 Selection-based Taxonomy

The selection component is divided into two parts, one for valid-time selection and one for selection not involving valid time. See Figure 5.

$$\{<\text{valid-time selection}>\}^* \times \{<\text{non-temporal selection}>\}^*$$

Figure 5: Top-level Selection-based Taxonomy

Both parts are based on the same observation. In general, a selection predicate is built from atomic selection predicates using logical operators (e.g., `and`, `or`, and `implies`) and parenthesis. Both parts categorize queries based on the atomic predicates used in the queries. As several types of atomic predicates may be used in the same query, queries generally fall into multiple categories (as indicated in Figure 5 by the Kleene star, "*"). We examine each part of the selection-based taxonomy in turn.

Atomic valid-time selection predicates are assumed to be of the form

$$arg_1 \ \texttt{op} \ arg_2$$

where `op` is a some comparison operator (e.g., `precedes`, and `contains`). It is assumed that arg_1 is the valid time of the data, and restrictions are imposed based on the type of the comparison operator, on the origin of arg_2, and on the type of arg_2. Figure 6 outlines the categories.

Three types of comparison operators are identified. First, a comparison operator may be duration-based. For example the operator `spanExceeds` returns true if the duration of the first argument is equal to or larger than the duration of the second argument. Second, comparison operators may be based on ordering. Operators in this category include `precedes` and `meets`. The first applies to all timestamps and evaluates to true if the largest time in the first argument is smaller than the smallest times in the second argument. Operator `meets` appears to be useful only for events and intervals. Two timestamps meet if they are not separated by any event (i.e., may be coalesced). Operators based on containment concern overlap, or lack thereof,

$$\left\{\begin{array}{c}\underline{\text{type of comparison operator}}\\ \text{duration-based}\\ \text{ordering-based}\\ \text{containment-based}\end{array}\right\} \times \left\{\begin{array}{c}\underline{\text{type of } arg_2}\\ \text{event}\\ \text{interval}\\ \text{element}\end{array}\right\} \times \left\{\begin{array}{c}\underline{\text{origin of } arg_2}\\ \text{explicitly supplied in query}\\ \text{user-defined attribute value}\\ \text{computed from other valid times}\end{array}\right\}$$

Figure 6: Valid-time Selection-based Taxonomy

of the arguments and include `=` (identity), `overlaps`, and `contains`.

The second argument (arg_2) may be an event, an interval, or an element. Also, it may come from three sources. First, it may be supplied directly in the query, as a constant. Second, it may be the value of a user-defined time attribute in an argument tuple. Note that this is only possible for events if first normal form is required. Third, like the first argument, the second argument may be computed from valid times and/or user-defined attribute values in the argument tuples.

If the three types of categories are completely orthogonal, this part of the taxonomy will contribute with a total of 27 categories. However, it may be debated whether intervals and elements may be used as values of user-defined attributes (resulting in non-1NF relations).

The final part of the selection-based taxonomy categorizes queries based solely on the part of the selection component that involves only ordinary, non-temporal selection.

Many possibilities for categorization exist. Below, in Figure 7, we distinguish between four significant types of atomic selection predicates. First, an attribute may be compared with a constant, supplied by the user. Second, attribute values, both in the same relation, may be compared. Third, a primary key value may be compared with a matching foreign key value. Fourth, arbitrary attributes of possibly distinct relations may be compared. In the figure, $\theta ::= < \mid > \mid \leq \mid \geq \mid =$, i.e., a combination of equality and/or the one of the two inequality operators. If we distinguish between situations where only equality is involved and situations where inequality is involved, this give 8 categories.

5.3 Overview and Naming of Categories

Each query has a single output component, zero or more valid-time selection components (one per such operator), and zero or more non-temporal selection-based components (one per such operator). The taxonomy is summarized in Figure 8. There, the names introduced in the taxonomy are used along with punctuation in order to name a category.

To exemplify the use of Figure 8 for naming categories, consider the query "When was *ED* manager of the Toy department?" Using the schema of Figure 2 (this schema will be used in the remainder of the document unless explicitly stated otherwise), this query is in the category shown next (with no valid-time selection).

(None, Element, Union) // (=, Constant) (=, Constant)

It may be observed that the taxonomy gives rise to a large number of categories. For example, assuming a single non-temporal operator and no valid-time operators, there are $38 \times 8 = 304$ categories. Adding a single valid-time operator while assuming orthogonality yields 8208 categories!

As a result, it becomes necessary to create classes of categories which then may be used for classifying the queries.

One approach would be to name a *class* of categories of queries, by simply replacing one or more of the entries with the Kleene star ("*"), e.g.,

(None, Element, Union) / (*,*,*) / (=, Constant) (=, Constant)

The above query category would be in this class. In the next section, we define the classes to be used in the test suite.

5.4 Forming Classes from Categories

The idea is to remove distinctions from the comprehensive taxonomy until a suitable number of classes is obtained. Figure 9 is thus a reduced version of Figure 8.

The second and third lines concern output. Only the prescence or absence of explicit attributes and

$$\left\{\begin{array}{c}\underline{\text{non-temporal attribute value selection}}\\ att\ \theta\ Constant\\ att_1\ \theta\ att_2\\ att_k\ \theta\ att_{fk}\\ att(rel_1)\ \theta\ att(rel_2)\end{array}\right\}\times\left\{\begin{array}{c}\underline{\text{comparison operator, }\theta}\\ \text{only equality }(=)\\ \text{inequality }(<>)\end{array}\right\}$$

Figure 7: Non-temporal Selection-based Taxonomy

```
<category>        ::= <output> '/' {<v-t selection> }* '/' {<non-t selection> }*
<output>          ::= '(' {None | Projected | Complete } ','                  /* explicit-attribute component
                      {None |                                                 /* no valid-time attribute
                       {Event | Interval | Element } ','                      /* type of valid-time attribute
                       {Intersection | Union | Derived | Imposed } } ')'      /* value of valid-time attribute
<v-t selection>   ::= '(' {Duration | Ordering | Containment } ','            /* operator type
                      {Event | Interval | Element } ','                       /* argument type
                      {Explicit | User-defined | Computed } ')'               /* argument origin
<non-t selection> ::= '(' {'=' | '<>' } ','                                   /* operator type
                      {Constant | Single | Foreign | Arbitrary } ')'          /* argument types
```

Figure 8: Overview of the Taxonomy used for Naming Categories

```
<class-name>             ::= <reduced output> '/' {<reduced v-t selection> }*
<reduced output>         ::= '(' {None | Proj/Comp } ','                /* explicit-attribute component
                             {None | Not empty } ')'                    /* valid-time attribute component
<reduced v-t selection>  ::= '(' {Duration | Other } ','                /* comparison operator type
                             {Event | Interval | Element } ','          /* argument type
                             {Computed | Other } ')'                    /* argument origin
```

Figure 9: Overview of the Classification of Queries

timestamps are distinguished, leading to three categories. The last three lines concern valid-time selection (non-temporal selection is disregarded). Comparison operators may be duration-based or not; arguments may be of either event, interval, or element type; and the arguments may or may not derive from valid times of tuples.

6 The Test-suite Queries

6.1 Overview

The structure of this section is based on Figure 9. There are three sections, each of which contains ten sections. The three top-level sections classify queries according to the output. Thus, the output from a query may have either only an explicit-attribute value component (O1), only a valid-time component (O2), or it may have both (O3).

Each top-level section contains the same ten sections. These divide queries based on a single, distinguished valid-time selection predicate[1]. The predicate is of the format arg_1 `op` arg_2 where `op` is a comparison operator which may (or may not) be duration-based. One argument is the valid time of tuples in the argument relation. The other argument may be of event, interval, or element type; orthogonally, it may (or may not) be computed from existing valid times in the argument relation. This results in a total of ten classes (the combination of duration-based predicates and event arguments is omitted).

(S1) (Duration, Interval, Computed)
(S2) (Duration, Interval, Other)
(S3) (Duration, Element, Computed)
(S4) (Duration, Element, Other)
(S5) (Other, Event, Computed)
(S6) (Other, Event, Other)
(S7) (Other, Interval, Computed)
(S8) (Other, Interval, Other)
(S9) (Other, Element, Computed)
(S10) (Other, Element, Other)

Finally, it is emphasized that the classification of natural-language queries according to a taxonomy based on a formal query language template (e.g., SQL and Quel) is necessarily approximate and, at times, ambiguous. Classification is also dependent on the specific relational database schema that is adopted (for the remainder of the paper, assume the schema of Figure 2).

[1] A query may contain several selection predicates, but it is classified according to a single predicate.

6.2 Explicit-attribute Output

This section involves queries which return only explicit-attribute values—no valid-time values are present in the result.

6.2.1 Class O1.S1 (Duration, Interval, Computed)

QUERY Q 2.1.1: Which departments had managers who served for the shortest continuous period?
ANSWER: "Book"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) /

QUERY Q 2.1.2: Who worked continuously in the Book department for at least as long as Di did?
ANSWER: "*ED*"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.1.3: Who worked continuously in the Toy department for at least as long as Di did?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.1.4: Who worked continuously in a department longer than their current manager worked in that department?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) (Containment, Event, Explicit) /

QUERY Q 2.1.5: Who had the same salary for the longest continuous time period?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) /

QUERY Q 2.1.6: Who worked for a manager in a department for a period as long as that manager managed the department?
ANSWER: "*DI*" and "*ED*"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) /

QUERY Q 2.1.7: Which managers served continuously longer than some other manager?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Computed) /

6.2.2 Class O1.S2 (Duration, Interval, Other)

QUERY Q 2.2.1: Which employees had the same salary for a single period of at least three years?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) /

QUERY Q 2.2.2: Who worked for the same manager for at least five years continuously?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) /

QUERY Q 2.2.3: Which employees have stayed in the same department throughout the past 5 years?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) (Containment, Event, Explicit) /

QUERY Q 2.2.4: For each department which has had the same managers and budget for the last 18 months, list its current name, manager, and budget.
ANSWER: "(Toy, *DI*, 100K)" and "(Book, *ED*, 50K)"
CATEGORY: (Complete, None) / (Duration, Interval, Explicit) (Containment, Event, Explicit) /

QUERY Q 2.2.5: Who has worked in the Toy department and has earned at least 40K throughout the last two years?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) / (=, Constant) (<>, Constant)

QUERY Q 2.2.6: Who had at least three raises in a continuous five-year period?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) /

QUERY Q 2.2.7: Who had the most raises in a continuous five-year period?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Interval, Explicit) /

6.2.3 Class O1.S3 (Duration, Element, Computed)

QUERY Q 2.3.1: Who worked in the Toy department for at least as long as *DI* worked there?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.3.2: Who worked in a department longer than their current manager worked in that department?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Element, Computed) (Containment, Event, Explicit) /

QUERY Q 2.3.3: Which managers managed which departments, longer than Di managed the Toy department?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.3.4: Who had the same salary for the longest total time?
ANSWER: "*ED*"
CATEGORY: (Projected, None) / (Duration, Element, Computed) /

QUERY Q 2.3.5: Which departments had managers who served for the shortest total time?
ANSWER: "Book"
CATEGORY: (Projected, None) / (Duration, Element, Computed) /

QUERY Q 2.3.6: List all employees currently in the Book department who received salaries of over 40K longer than *ED* did.
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Element, Computed) / (=, Constant) (<>, Constant) (=, Constant)

QUERY Q 2.3.7: Who worked in the Toy department for at least as long as the total time that the Toy department was NOT managed by *ED*?
ANSWER: "*DI*"
CATEGORY: (Projected, None) / (Duration, Element, Computed) / (=, Constant) (<>, Constant)

QUERY Q 2.3.8: Find the names of employees that have been in a department named Toy for a shorter period than has *DI*.
ANSWER: "Ed" and "Edward."
CATEGORY: (Projected, None) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.3.9: Find the current name and department for the employees which made $40K for a longer period than *DI* did.
ANSWER: "(Edward, Book)."
CATEGORY: (Projected, None) / (Duration, Element, Computed) (Containment, Event, Explicit) / (=, Constant)

6.2.4 Class O1.S4 (Duration, Element, Other)

QUERY Q 2.4.1: Who managed the Book department for at least two years?
ANSWER: "*ED*"
CATEGORY: (Projected, None) / (Duration, Element, Explicit) / (=, Constant)

QUERY Q 2.4.2: Which employees had the same salary for at least three years?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Element, Explicit) /

QUERY Q 2.4.3: Who worked for the same manager for at least five years?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Element, Explicit) /

QUERY Q 2.4.4: Who worked in a department for less than 6 months total?
ANSWER: "None"
CATEGORY: (Projected, None) / (Duration, Element, Explicit) /

QUERY Q 2.4.5: Who worked for the same manager for total time of at least five years?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Duration, Element, Explicit) /

6.2.5 Class O1.S5 (Other, Event, Computed)

QUERY Q 2.5.1: Find *ED*'s skills when he joined the Book department
ANSWER: "typing," "filing" and "driving."
CATEGORY: (Projected, None) / (Containment, Event, Computed) / (=, Constant)

QUERY Q 2.5.2: Find the name and the budget of *ED*'s departments when he joined them.
ANSWER: "(Toy, \$150K)" and "(Book, \$50K)."
CATEGORY: (Projected, None) / (Containment, Event, Computed) / (=, Constant) (=, Foreign)

QUERY Q 2.5.3: For each employee who was in the Toy department when it opened, find all data and skills that were valid at the time.
ANSWER: "(Di, \$30K, Toy, F, 10/1/60, directing)."
CATEGORY: (Complete, None) / (Containment, Event, Computed) / (=, Constant) (=, Foreign)

QUERY Q 2.5.4: Find the names valid when the budget of the Toy department was decreased of the employees who had been working in the Toy department before the budget was decreased.
ANSWER: "Ed" and "Di"
CATEGORY: (Projected, None) / (Ordering, Event, Computed) (Ordering, Interval, Computed) / (=, Constant) (<>, Constant)

QUERY Q 2.5.5: Find *ED*'s skills when his salary increased from \$30K to \$40K.
ANSWER: "typing," "filing" and "driving."
CATEGORY: (Projected, None) / (Containment, Event, Computed) (Ordering, Interval, Computed) / (=, Constant)

6.2.6 Class O1.S6 (Other, Event, Other)

QUERY Q 2.6.1: Find the name, current budget, and manager of the Toy department.
ANSWER: "(Toy, *DI*, \$100K)."
CATEGORY: (Complete, None) / (Containment, Event, Explicit) / (=, Constant)

QUERY Q 2.6.2: Find the skills for which *ED* became qualified after 1/1/83.
ANSWER: "filing" and "driving."
CATEGORY: (Projected, None) / (Ordering, Event, Explicit) / (=, Constant)

QUERY Q 2.6.3: Find *DI*'s salary on her 25^{th} birthday.
ANSWER: "\$40K."
CATEGORY: (Projected, None) / (Containment, Event, User-defined) / (=, Constant)

QUERY Q 2.6.4: Find the departments *ED* worked in before and not after 1/1/88.
ANSWER: "Toy."
CATEGORY: (Projected, None) / (Ordering, Event, Explicit) / (=, Single)(=, Constant)

QUERY Q 2.6.5: Find the date of birth and current name of the women who were working in the Toy department on 1/1/83.
ANSWER: "(Di, 10/1/60)."
CATEGORY: (Projected, None) / (Containment, Event, Explicit) / (=, Constant)

QUERY Q 2.6.6: Who worked in their current department for a longer time than their current manager worked in that department?
ANSWER: "None."
CATEGORY: (Projected, None) / (Containment, Event, Explicit) /

6.2.7 Class O1.S7 (Other, Interval, Computed)

QUERY Q 2.7.1: Find the names of all employees that changed department while *DI* was working in a department called Toy.
ANSWER: "Ed" and "Edward."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.7.2: Which of all skills ever recorded did *ED* not acquire while working in the Book department.
ANSWER: "Driving," "Directing," "Filing," and "Typing."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (<>, Constant)

QUERY Q 2.7.3: Of the skills at some time possessed by *ED*, list those he did not acquire while he was working in the Book department.
ANSWER: "Driving," "Filing," and "Typing."
CATEGORY: (Projected, None) / (Containment, Interval, Explicit) (Containment, Interval, Explicit) / (<>, Constant)

QUERY Q 2.7.4: For any employee who reacquired a skill, find the name of the employee when a skill was reacquired.
ANSWER: "Ed."
CATEGORY: (Projected, None) / (Ordering, Interval, Computed) (Ordering, Interval, Computed) / (<>, Constant)

QUERY Q 2.7.5: Find the gender and name at the time of all employees that started working in some department before *ED* acquired the skill Driving for the second time.
ANSWER: "(Ed, M) and (Di, F)."
CATEGORY: (Projected, None) / (Ordering, Interval, Computed) (Ordering, Event, Computed) / (Sequence, Constant)

QUERY Q 2.7.6: Who worked in a department for a given manager for at least the period when that manager managed that department?
ANSWER: "*ED*" and "*DI*."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) /

QUERY Q 2.7.7: What was the highest salary earned by *ED* before changing his name to Edward?
ANSWER: "40K."
CATEGORY: (Projected, None) / (Ordering, Interval, Computed) / (=, Constant) (=, Constant)

6.2.8 Class O1.S8 (Other, Interval, Other)

QUERY Q 2.8.1: Find the name and skill pairs sometime possessed by people who worked in the Book or Toy department last year.
ANSWER: "(Edward, Typing), (Edward, Filing), (Edward, Driving), (Ed, Typing), (Ed, Filing), (Ed, Driving), and (Di, Directing)."
CATEGORY: (Complete, None) / (Containment, Interval, Explicit) / (=, Constant) (=, Constant)

QUERY Q 2.8.2: Find the current name and skills of all people who worked for the Book or Toy department last year.
ANSWER: "(Edward, Typing), (Edward, Filing), and (Di, Directing)."
CATEGORY: (Complete, None) / (Containment, Interval, Computed) (Containment, Interval, Explicit) / (=, Constant) (=, Constant)

QUERY Q 2.8.3: Find their names (when they reported to *DI*) for all people who reported to *DI* before last year.
ANSWER: "Ed" and "Di."
CATEGORY: (Complete, None) / (Ordering, Interval, Explicit) / (<, Constant)

QUERY Q 2.8.4: Find the current manager of anyone who acquired a skill between 1983 and 1987 inclusive.
ANSWER: "*ED*."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 2.8.5: Find the name current of anyone who lost a skill in the last four years.
ANSWER: "Edward."
CATEGORY: (Projected, None) / (Containment, Interval, Explicit) / (=, Constant)

QUERY Q 2.8.6: Find the current name and department of anyone who changed their name or salary between July 1987 and June 1988 inclusive.
ANSWER: "(Edward, Book)."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 2.8.7: Which employees stayed at their first salary for less than one year?
ANSWER: "*ED*"
CATEGORY: (Projected, None) / (Containment, Interval, Explicit) /

QUERY Q 2.8.8: List the names, and current managers and budgets of all departments with budgets

of less than 200K during any period between January 1, 1985 and December 31, 1989.
ANSWER: "(Toy, *DI*, 100K)" and "(Book, *ED*, 50K)."
CATEGORY: (Complete, None) / (Containment, Interval, Explicit) / (<>, Constant)

QUERY Q 2.8.9: Who worked in the Toy department at some point and earned at least 40K throughout the last two years?
ANSWER: "*ED*" and "*DI*"
CATEGORY: (Projected, None) / (Containment, Element, Explicit) / (=, Constant) (<>, Constant)

6.2.9 Class O1.S9 (Other, Element, Computed)

QUERY Q 2.9.1: Find the names of departments that always had a budget greater than $90K during the times when managed by someone named Di.
ANSWER: "Toy."
CATEGORY: (Projected, None) / (Containment, Element, Computed) / (=, Constant) (=, Constant) (=, Foreign)

QUERY Q 2.9.2: Find *ED*'s salaries when he worked in the same department as *DI*.
ANSWER: "$20K, $30K, $40K."
CATEGORY: (Projected, None) / (Containment, Element, Computed) / (=, Constant) (=, Single) (=, Constant)

QUERY Q 2.9.3: Find the names of the departments that *ED* worked in while earning $40K.
ANSWER: "Toy" and "Book."
CATEGORY: (Projected, None) / (Containment, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.9.4: Find *ED*'s names after he left the Toy department.
ANSWER: "Ed" and "Edward."
CATEGORY: (Projected, None) / (Ordering, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.9.5: Find the skills that *ED* possessed sometime when he worked in the Toy department.
ANSWER: "Driving," "Filling," and "Typing."
CATEGORY: (Projected, None) / (Containment, Element, Computed) / (=, Constant) (=, Foreign) (=, Constant)

QUERY Q 2.9.6: What new skills did *ED* obtain after he had changed his name to Edward?
ANSWER: "None."
CATEGORY: (Projected, None) / (Ordering, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 2.9.7: What were Toy's departmental budgets when it had a manager named Di?
ANSWER: "150K," "200K," and "100K."
CATEGORY: (Projected, None) / (Containment, Element, Computed) / (=, Constant) (=, Constant)

6.2.10 Class O1.S10 (Other, Element, Other)

QUERY Q 2.10.1: Which managers managed which departments between January 1, 1982 and December 31, 1989?
ANSWER: "(`DI`, Toy)" and "(*ED*, Book)."
CATEGORY: (Projected, None) / (Containment, Element, Explicit) /

6.3 Valid-time Output

This section involves queries which return only valid-time values—no explicit-attribute values are present in the result.

6.3.1 Class O2.S1 (Duration, Interval, Computed)

QUERY Q 3.1.1: Find the times when persons with a shorter employment in the Toy department than *DI* were employed in the Book department.
ANSWER: "{ [4/1/87 - `now`]}"
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) / (=, Constant)

QUERY Q 3.1.2: Find the employment periods of persons that made 40K for a longer time than *DI* made 40K.
ANSWER: "{ [2/1/82 - 1/31/87], [4/1/87 - `now`]}"
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) / (=, Constant)

QUERY Q 3.1.3: Find the starting times in the Book department of persons which possessed the Filing skill for a longer time than *DI*.
ANSWER: "{ 4/1/87}"
CATEGORY: (None, Element, Derived) / (Duration, Element, Computed) / (=, Constant)

QUERY Q 3.1.4: Return the times when persons employed a shorter time than *DI* acquired a skill.
ANSWER: "{ 1/1/82, 4/1/82, 6/1/84, 1/1/85, }"
CATEGORY: (None, Element, Derived) / (Duration, Element, Computed) / (=, Constant)

QUERY Q 3.1.5: Find the employment periods of persons employed shorter time than *DI*.
ANSWER: "{ [2/1/82 - 1/31/87], [4/1/87 - `now`]}"
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) /(=, Constant)

QUERY Q 3.1.6: When did someone get a raise more quickly than *DI* got her first raise?
ANSWER: "06/01/82" and "09/01/86."
CATEGORY: (None, Event, Derived) / (Duration, Interval, Computed) (Ordering, Interval, Computed) / (= ,Constant)

QUERY Q 3.1.7: What was the longest period when no one was hired or left employment?
ANSWER: "02/01/82–01/31/87"
CATEGORY: (None, Interval, Union) / (Duration, Interval, Computed) /

QUERY Q 3.1.8: What was the longest period when no one received a raise?
ANSWER: "09/01/86–`now`"
CATEGORY: (None, Interval, Union) / (Duration, Interval, Computed) /

QUERY Q 3.1.9: When was the longest period when a department was without a manager?
ANSWER: "Never."
CATEGORY: (None, Interval, Union) / (Duration, Interval, Computed) /

6.3.2 Class O2.S2 (Duration, Element, Other)

QUERY Q 3.2.1: Find employment periods in the Toy department for persons that have worked there for at least 8 years.
ANSWER: "{ [1/1/82 - `now`]}"
CATEGORY: (None, Element, Union) / (Duration, Element, Explicit) /(=, Constant)

QUERY Q 3.2.2: Find the starting times of managers which managed a department for at least 5 years.
ANSWER: "{ 1/1/82}"
CATEGORY: (None, Element, Derived) / (Duration, Element, Explicit) /(=, Constant)

QUERY Q 3.2.3: Find the rehiring dates of employees with a gap in employment that exceeds 1 month.
ANSWER: "{ 4/1/87}"
CATEGORY: (None, Element, Union) / (Duration, Element, Explicit) /(=, Constant)

QUERY Q 3.2.4: Find the times when persons possessed skills that they lost and regained more than 1 year later.
ANSWER: "{ [1/1/82 - 5/1/82], [6/1/84 - 5/31/88]}"
CATEGORY: (None, Element, Derived) / (Duration, Element, Explicit) /(=, Constant)

QUERY Q 3.2.5: Find budget periods that exceed 2 years.
ANSWER: "{ [1/1/87 - `now`], [4/1/87 - `now`]}"
CATEGORY: (None, Element, Derived) / (Duration, Element, Explicit) /(=, Constant)

QUERY Q 3.2.6: When did no one's salary change for at least six months?
ANSWER: "06/01/82–07/31/84," "08/01/84–01/31/85," "02/01/85–08/31/86," "09/01/86–`now`."
CATEGORY: (None, Interval, Derived) / (Duration, Interval, Explicit) (temporal aggregate) /

6.3.3 Class O2.S3 (Duration, Element, Computed)

QUERY Q 3.3.1: When did somebody have the same salary for the longest continuous period?
ANSWER: "09/01/86–01/01/90"
CATEGORY: (None, Element, Derived) / (Duration, Element, Computed) (temporal aggregate) /

QUERY Q 3.3.2: When did anybody work for a manager in a department for as long as that manager managed that department?
ANSWER: "1/1/82–1/1/90" and "4/1/87–1/1/90."
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) /

QUERY Q 3.3.3: When did someone manage the Toy department for longer than *DI* did?
ANSWER: "No time."
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 3.3.4: When did anyone have a skill longer than *ED* had Driving?
ANSWER: "{ [1/1/82 - `now`], [4/1/82 - `now`], [1/1/85 - `now`] }"
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

6.3.4 Class O2.S4 (Duration, Element, Other)

6.3.5 Class O2.S5 (Other, Event, Computed)

6.3.6 Class O2.S6 (Other, Event, Other)

QUERY Q 3.6.1: When did anybody have at least the skills that *DI* currently has?
ANSWER: "No time."
CATEGORY: (None, Element, Intersection) / (Containment, Event, Explicit) / (=, Constant)

6.3.7 Class O2.S7 (Other, Interval, Computed)

QUERY Q 3.7.1: When did the Toy budget decrease?

ANSWER: "1/1/87."
CATEGORY: (None, Event, Derived) / (Ordering, Interval, Computed) / (=, Constant) (<>, Single)

QUERY Q 3.7.2: When did an employee change name?
ANSWER: "1/1/88."
CATEGORY: (None, Event, Derived) / (Ordering, Interval, Computed) / (=, Single) (<>, Single)

QUERY Q 3.7.3: When did the salary of an employee increase while the employee was a manager?
ANSWER: "8/1/84" and "9/1/86."
CATEGORY: (None, Event, Derived) / (Ordering, Interval, Computed) (Containment, Element, Computed) / (=, Foreign) (=, Const) (<>, Single)

QUERY Q 3.7.4: Find the periods during which *DI* earned \$40K and was a manager of the Toy department.
ANSWER: "8/1/84 – 8/31/86."
CATEGORY: (None, Element, Intersection) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 3.7.5: Find the acquisition dates of the skills *ED* acquired before or during the year he joined the Toy department.
ANSWER: "1/1/82" and "4/1/82."
CATEGORY: (None, Event, Derived) / (Containment, Interval, Computed) / (=, Constant)

6.3.8 Class O2.S8 (Other, Interval, Other)

QUERY Q 3.8.1: Find the beginning of a continuous period in which *ED* was named Edward, in which he had a constant salary, and which includes the year 1989.
ANSWER: "1/1/88."
CATEGORY: (None, Event, Derived) / (Containment, Interval, Explicit) / (=, Constant)

QUERY Q 3.8.2: Find the dates, before or after the years 1984 and 1985, when *ED* acquired a skill.
ANSWER: "1/1/82" and "4/1/82."
CATEGORY: (None, Event, Derived) / (Ordering, Interval, Explicit) / (=, Constant)

QUERY Q 3.8.3: Find *ED*'s unemployment periods when he was not exactly 30 years old.
ANSWER: "2/1/87 – 3/31/87."
CATEGORY: (None, Element, Union) / (Ordering, Interval, User-defined) (Containment, Interval, User-defined) / (=, Constant)

QUERY Q 3.8.4: Find the time periods when *DI* worked in the department in which she has been working during all of 1987.
ANSWER: "1/1/82 – `now`."
CATEGORY: (None, Element, Union) / (Containment, Interval, Explicit) / (=, Constant)

QUERY Q 3.8.5: Find all the dates, between 1/1/83 and 12/31/85, when the Toy department budget changed.
ANSWER: "8/1/84."
CATEGORY: (None, Event, Derived) / (Containment, Interval, Explicit) (Ordering, Interval, Computed) / (=, Single) (<>, Single)

6.3.9 Class O2.S9 (Other, Element, Computed)

QUERY Q 3.9.1: At what times did an employee simultaneously posses at least the same skills that *DI* possessed?
ANSWER: "No times."
CATEGORY: (None, Element, Intersection) / (Containment, Element, Computed) / (=, Constant)

QUERY Q 3.9.2: When was the budget for Toy department more than 100K?
ANSWER: "01/01/82–12/31/86."
CATEGORY: (None, Element, Derived) / (Containment, Element, Computed) / (=, Constant) (<>, Constant)

QUERY Q 3.9.3: What was the last continuous period when *ED* was named Edward and had Driving, Filing and Typing as skills simultaneously?
ANSWER: "01/01/88–05/31/88."
CATEGORY: (None, Interval, Intersection) / (Containment, Event, Computed) (temporal aggregate) / (=, Constant) (=, Constant)

QUERY Q 3.9.4: When did *DI* earn less than *ED*?
ANSWER: "Never."
CATEGORY: (None, Element, Intersection) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 3.9.5: When did *ED* work in Toy department while the department was managed by *DI*?
ANSWER: "02/01/82–01/31/87"
CATEGORY: (None, Element, Union) / (Duration, Element, Computed) / (=, Constant) / (=, Constant) (=, Constant)

QUERY Q 3.9.6: When did an employee currently named Edward have driving skills?

ANSWER: "01/01/82–05/01/82" and "06/01/84–05/31/88."
CATEGORY: (None, Element, Derived) / (Containment, Element, Computed) / (=, Constant) (=, Constant)

6.3.10 Class O2.S10 (Other, Element, Other)

6.4 Explicit-attribute and Valid-time Output

The output from queries in this section contains explicit attribute values with associated valid times.

6.4.1 Class O3.S1 (Duration, Interval, Computed)

QUERY Q 4.1.1: Who, and when, were continuously employed in the Toy department shorter than *DI* was continuously employed in the Toy department?
ANSWER: "*ED* from 01-Feb-1982 to 31-Jan-1987."
CATEGORY: (Projected, Interval, Derived) / (Duration, Interval, Computed) / (=, Constant) (=, Constant) (=, Constant)

QUERY Q 4.1.2: Who, and when, were continuously employed in the Toy department shorter than *DI* was continuously employed in the Toy department, and what are their gender and date of birth?
ANSWER: "*ED* from 01-Feb-1982 to 31-Jan-1987, born 1-Jul-1955 and Male."
CATEGORY: (Complete, Interval, Derived) / (Duration, Interval, Computed) / (=, Constant) (=, Constant) (=, Constant)

QUERY Q 4.1.3: Who were continuously employed in the Toy department shorter than *DI* was continuously employed in the Toy department, and when did this employment start?
ANSWER: "*ED*, on 01-Feb-1982."
CATEGORY: (Projected, Event, Derived) / (Duration, Interval, Computed) / (=, Constant) (=, Constant) (=, Constant)

QUERY Q 4.1.4: Return the name on 01-Jan-1984 along with the date 01-Jan-1984 for each employee who was continuously employed in the Toy department shorter than *DI* was continuously employed in that department.
ANSWER: "Ed, 01-Jan-1984."
CATEGORY: (Projected, Event, Imposed) / (Duration, Interval, Computed) / (=, Constant) (=, Constant) (=, Constant)

6.4.2 Class O3.S2 (Duration, Interval, Other)

QUERY Q 4.2.1: When was the Toy department's budget constant and greater than $175K for more than one year, and what was the budget?
ANSWER: "$200K from 01-Aug-1984 to 31-Dec-1986."
CATEGORY: (Projected, Interval, Derived) / (Duration, Interval, Supplied) / ($\neq$, Constant) (=, Constant)

QUERY Q 4.2.2: When was the Toy department's budget constant and greater than $175K for more than one year, and who was the manager for that time?
ANSWER: "From 01-Aug-1984 to 31-Dec-1986, managed by *DI*."
CATEGORY: (Projected, Interval, Derived) / (Duration, Interval, Supplied) / ($\neq$, Constant) (=, Constant)

QUERY Q 4.2.3: Who managed a department with a budget that exceeded $175K and then held constant for one year, and when did that occur?
ANSWER: "*DI*, from 01-Aug-1984 to 31-Dec-1986."
CATEGORY: (Projected, Interval, Intersection) / (Duration, Interval, Supplied) / (=, Constant) (=, Foreign)

QUERY Q 4.2.4: What departments were in continuous operation (and when) longer than the duration between *ED*'s and *DI*'s birth dates?
ANSWER: "Toy, from 01-Jan-1982 to present."
CATEGORY: (Projected, Interval, Derived) / (Duration, Interval, Computed) / (=, Constant) (=, Constant)

QUERY Q 4.2.5: Who worked in one department for at least two years continuously and what were the periods of employment in that department?
ANSWER: "{ (*ED*, { [2/1/82 - 1/31/87] }), (*ED*, { [4/1/87 - 1/1/90] }), (*DI*, { [1/1/82 - 1/1/90] }) }"
CATEGORY: (Projected, Element, Derived) / (Duration, Interval, Explicit) /

6.4.3 Class O3.S3 (Duration, Element, Computed)

QUERY Q 4.3.1: Who, when, and for which department did anybody work for as long as the length of time that department's budget was below 200K?
ANSWER: "(*DI*, Toy, 01/01/82–01/01/90)" and "(*ED*, Book, 4/1/87–01/01/90
CATEGORY: (Projected, Element, Intersection) / (Duration, Element, Computed) / (<>, Constant)

QUERY Q 4.3.2: Who and when did anybody work in a department longer than their current manager worked in that department?
ANSWER: "None"
CATEGORY: (Projected, Element, Derived) / (Duration, Element, Computed) / (Containment, Event, Explicit) /

QUERY Q 4.3.3: For all employees who managed any departments at least as long as *DI* managed the Toy department, list their names, their gender, their departments and their salary histories during that time.
ANSWER: "(*DI*, (Di, 01/01/82–01/01/90), F, (Toy, 01/01/82–01/01/90), ((30K, 01/01/82–07/31/84), (40K, 08/01/84–08/31/86), (50K, 09/01/86–01/01/90)))"
CATEGORY: (Projected, Element, Derived) / (Duration, Element, Computed) / (=, Constant) (=, Constant)

QUERY Q 4.3.4: For departments that had a manager that served for the shortest total time, list the department name, the shortest-serving manager(s), an the times when those managers served the department.
ANSWER: "Book, *ED*, 04/01/87–01/01/90"
CATEGORY: (Projected, Element, Intersection) / (Duration, Element, Computed) /

QUERY Q 4.3.5: For all departments which had budgets of at least 200K for a longer total time than budgets of less than 200K, list their names, budgets, and times when the budgets were not below 200K.
ANSWER: "None."
CATEGORY: (Projected, Element, Derived) / (Duration, Element, Computed) / (<>, Constant) (<>, Constant)

QUERY Q 4.3.6: What skills did *ED* hold for at least as long as the total time during his employment that he did not have the Driving skill, and when did he have those skills?
ANSWER: "Typing, 04/01/82–01/01/90," "Filing, 01/01/85–01/01/90," and "Driving, (01/01/82–05/01/82, 06/01/84–05/31/88)."
CATEGORY: (Projected, Element, Derived) / (Duration, Element, Computed) / (=, Constant) (<>, Constant)

6.4.4 Class O3.S4 (Duration, Element, Other)

QUERY Q 4.4.1: Find the oldest employee who was a Typist on 12/31/85, and the times when that employee had been a Typist so far.
ANSWER: "(*ED*, 4/1/82 – 12/31/85)"
CATEGORY: (Projected, Element, Duration) / (Duration, Element, Explicit) / (=, Constant), (=, Foreign)

QUERY Q 4.4.2: For employees in the Toy department who had worked less than *DI* in that department as of 1/1/85, find their names on 1/1/85 and the time difference on 1/1/85.
ANSWER: "(Ed, 1 month less)"
CATEGORY: (Projected, —) / (Duration, Element, Explicit) / (=, Constant), (<>, Single)

QUERY Q 4.4.3: Find the current employees who worked the least during 1987, and the times in 1987 during which they worked.
ANSWER: "(*ED*, ((1/1/87 – 1/31/87), (4/1/87 – 12/31/87))"
CATEGORY: (Projected, Element, Intersection) / (Duration, Element, Explicit) / (Nothing)

6.4.5 Class O3.S5 (Other, Event, Computed)

QUERY Q 4.5.1: List the names and ages of all employees at the time they received their first salary increment.
ANSWER: "(Ed, 26 years & 11 months), (Di, 23 years & 10 months)"
CATEGORY: (Projected, —) / (Containment, Event, Computed) /

QUERY Q 4.5.2: List the name and salary histories up until their 25th birthday of all female employees.
ANSWER: "((Di, (1/1/82–10/1/85), ((30K, 01/01/82–07/31/84), (40K, 08/01/84–10/01/85)))"
CATEGORY: (Projected, Element, Derived) / (Containment, Event, Computed) (Duration, Interval, explicit) / (=, Constant)

QUERY Q 4.5.3: When and who ever changed their names?
ANSWER: "(*ED*, 01/01/88)"
CATEGORY: (Projected, Event, Derived) / (Containment, Event, Computed) /

QUERY Q 4.5.4: How old was *ED* and what skills did he have at the time he changed his name to Edward?
ANSWER: "(32 years & 6 months, {Driving, Filing, Typing})"
CATEGORY: (Projected, —) / (Containment, Event, Computed) / (=, Constant) (=, Constant)

QUERY Q 4.5.5: When did *ED* acquire the Driving skill, and what other skills did he have at the time?
ANSWER: "(01/01/82, None), (06/01/84, Typing)"
CATEGORY: (Projected, Event, Intersection) / (Containment, Event, Computed) / (=, Constant) (=, Constant)

QUERY Q 4.5.6: Who was the first female manager of the Toy department, and when did she become manager of that department for the first time?
ANSWER: "(*DI*, 01/01/82)"
CATEGORY: (Projected, Event, Derived) / (Containment, Event, Computed) (temporal aggregate) / (=, Constant) (=, Constant)

6.4.6 Class O3.S6 (Other, Event, Other)

QUERY Q 4.6.1: Find the name and salary histories of employees whose date-of-birth was after 1/1/56.
ANSWER: "((Di, (1/1/82–1/1/90), ((30K, 01/01/82–07/31/84), (40K, 08/01/84–08/31/86), (50K, 09/01/86–01/01/90)))"
CATEGORY: (Projected, Element, Derived) / (Ordering, Event, User-defined) /

QUERY Q 4.6.2: Find the name and salary histories of employees who were called "Ed" after 1/1/88.
ANSWER: "Empty Answer."
CATEGORY: (Projected, Element, Derived) / (Ordering, Event, User-Defined) /

QUERY Q 4.6.3: Find the name and salary histories since their latest pay raise of employees whose latest pay raise was in 1985.
ANSWER: "(((Ed, (2/1/85 – 12/31/87)), (Edward, (1/1/88 – 1/1/90))), (40K, ((2/1/85 – 1/31/87), (4/1/87 – 1/1/90))))"
CATEGORY: (Projected, Element, Derived) / (Containment, Event, Explicit) /

QUERY Q 4.6.4: Find the name and salary histories of employees whose latest pay raise occurred after the date-of-birth of every other employee.
ANSWER: The answer contains Both *DI*'s and *ED*'s enitre Name and Salary history, see Section 4.2.
CATEGORY: (Projected, Element, Derived) / (Ordering, Event, User-Defined) /

QUERY Q 4.6.5: Find the name and salary histories of employees whose latest pay raise occurred on the date-of-birth of some other employee.
ANSWER: "Empty answer."
CATEGORY: (Projected, Element, Derived) / (Containment, Event, User-Defined) /

QUERY Q 4.6.6: Who and when had at least the skills that *DI* currently has?
ANSWER: "(*DI*, (1/1/82–`now`"
CATEGORY: (Projected, Element, Derived) / (Containment, Event, Explicit) / (=, Constant)

6.4.7 Class O3.S7 (Other, Interval, Computed)

QUERY Q 4.7.1: For the time before *ED* first worked in the Toy department, find the salary paid, and the time it was paid, of any employee who was in the Toy department at a time before Ed worked there.
ANSWER: "($30K, 1/1/82 – 1/31/82)"
CATEGORY: (Projected, Not Empty) / (Ordering, Interval, Computed) / (<>, Constant)

QUERY Q 4.7.2: Find the greatest salary under $50K paid to *ED* and the times during which it was paid.
ANSWER: "($40K, 2/1/85 – 1/31/87)," "($40k, 4/1/87 – present)"
CATEGORY: (Projected, Not Empty) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 4.7.3: Find *ED*'s salary history.
ANSWER: "($20K, 2/1/82 – 5/31/82)," "($30K, 6/1/82 – 1/31/85)," and "($40K, (2/1/85 – 1/31/87), (4/1/87 – present))."
CATEGORY: (Projected, Not Empty) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 4.7.4: For employees that were drivers and simultaneously made less than $40K, find the names, salaries, and times during which this occurred.
ANSWER: "(Ed, $20K, 2/1/82 – 5/1/82)" and "(Ed, $30K, 6/1/84 – 1/31/85)."
CATEGORY: (Projected, Not Empty) / (Containment, Interval, Computed) / (<>, Constant)

QUERY Q 4.7.5: For the Toy department when *ED* worked there, find the budgets and associated times.
ANSWER: "(2/1/82 – 7/31/84, $150K)," "(8/1/84 – 12/31/86, $200K)," and "(1/1/87 – 1/31/87, $100K)"
CATEGORY: (Projected, Not Empty) / (Containment, Interval, Computed) / (=, Constant), (=, Foreign

6.4.8 Class O3.S8 (Other, Interval, Other)

QUERY Q 4.8.1: For all employees that have been in the Book or Toy departments sometime during the

last two years, find their current names and their skills together with the times when they were valid.
ANSWER: "(Edward, Typing, 1-Apr-1982 to date), (Edward, Filing, 1-Jan-1985 to date), (Edward, Driving, 1-Jan-1982 to 1-May-1982), (Edward, Driving, 1-Jun-1984 to 31-May-1988) and (Di, Directing, 1-Jan-1982 to date)."
CATEGORY: (Projected, Element) / (Containment, Interval, Explicit) / (=, Constant) (=, Constant)

QUERY Q 4.8.2: Find the current names of all people who reported to *DI* before last year along with the time when they reported to her.
ANSWER: "(Edward, 1-Feb-1982 to 31-Jan-1987)"
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (<, Constant)

QUERY Q 4.8.3: Find the manager and time when a skill was acquired between 1983 and 1987 (inclusive) by anyone who acquired a skill between these times.
ANSWER: "(*DI*, 1-Jun-1984) and (*DI*, 1-Jan-1985)."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (=, Constant)

QUERY Q 4.8.4: For anyone who had two raises in the period March 1982 to March 1985 (inclusive), find the current name and the dates when raises occurred during the aforementioned times.
ANSWER: "(Edward, 1-Jun-1982, 1-Feb-1985)."
CATEGORY: (Projected, None) / (Containment, Interval, Computed) / (=, Constant)

6.4.9 Class O3.S9 (Other, Element, Computed)

QUERY Q 4.9.1: Find the salary history associated with the name Ed when it was associated with a person that had the Driving skill.
ANSWER: "($20K, 2/1/82 – 5/1/82)," "($30K, 6/1/84 – 1/31/85)" and "($40K, 2/1/85 – 1/31/87 ∪ 4/1/87 – 12/31/87)"
CATEGORY: (Projected, Not Empty) / (Containment, Element, Computed) / (=, Constant)

QUERY Q 4.9.2: Find the salary history, during the periods in which *ED* had a driving skill, of the employees who earned less than $50K throughout 1989.
ANSWER: "(—, 1/1/82 – 1/31/82)," "($20K, 2/1/82 – 5/1/82)," "($30K, 6/1/84 – 1/31/85)," "($40K, 2/1/85 – 1/31/87)," "(—, 1/31/87 – 3/30/87)" and "($40K, 4/1/87 – 5/31/88)."
CATEGORY: (Projected, Not Empty) / (Containment, Element, Computed) (Containment, Interval, Explicit) / (=, Constant) (<>, Constant)

QUERY Q 4.9.3: Find the name and salary histories of male employees when they were directed by a woman.
ANSWER: "(Ed, 2/1/82 – 1/31/87)" and "(($20K, 2/1/82 – 5/31/82), ($30K, 6/1/82 – 1/31/85), ($40K, 2/1/85 – 1/31/87))."
CATEGORY: (Projected, Not Empty) / (Containment, Element, Computed) / (=, Constant) (=, Foreign)

QUERY Q 4.9.4: Find the department, the current manager name, and the periods when a department manager earned more than one third of the departmental budget.
ANSWER: "(Toy, Di, 1/1/87 – now), (Book, Edward, 4/1/87 – now)."
CATEGORY: (Projected, Not Empty) / (Containment, Element, Computed) / (<>, Arbitrary)

QUERY Q 4.9.5: Find the name and the salary history of the employees in the periods they earned as much as their managers (distinct from themselves).
ANSWER: "(Ed, $30K, 6/1/82 – 7/31/84)" and "(Ed, $40K, 2/1/85 – 8/31/86)."
CATEGORY: (Projected, Not Empty) / (Containment, Element, Computed) (Containment, Interval, Computed) / (=, Foreign) (=, Single) (<>, Single)

QUERY Q 4.9.6: When did one person earn a lower salary than another younger person, and who were those persons?
ANSWER: "*ED* and *DI*, 02/01/82–05/31/82, 08/01/84–01/31/85, 09/01/86–01/31/87, 04/01/87–01/01/90."
CATEGORY: (Projected, Element, Derived) / (Containment, Element, Computed) /

QUERY Q 4.9.7: When and who had the same salary for the longest continuous period of time?
ANSWER: "09/01/86–01/01/90, *DI*"
CATEGORY: (Projected, Element, Derived) / (Containment, Element, Computed) /

QUERY Q 4.9.8: List *DI*'s skill and salary histories during the time she was a manager.
ANSWER: "(Directing, 01/01/82–01/01/90)" and "(30K, 01/01/82–07/31/84), (40K, 08/01/84–08/31/86), (50K, 09/01/86–01/01/90)"
CATEGORY: (Projected, Element, Intersection) / (Containment, Element, Computed) / (=, Constant)

QUERY Q 4.9.9: List the names and salary histories of all employees when they were managers and earned at least 36K.

ANSWER: "((Di, 08/01/84–01/01/90), ((40K, 08/01/84– 08/31/86), (50K, 09/01/86–01/01/90))), (((Ed, 04/01/87–12/31/87), (Edward, 01/01/88–01/01/90)), (40K, 04/01/87–01/01/90))"

CATEGORY: (Projected, Element, Derived) / (Containment, Element, Computed) / (<>, Constant)

6.4.10 Class O3.S10 (Other, Element, Other)

QUERY Q 4.10.1: Find the budget history in the period from 1/1/82 to 12/31/84 and from 1/1/87 till now of all departments *ED* ever worked in.

ANSWER: "(Toy, $150K, 2/1/82 – 7/31/84)," "(Toy, $200K, 8/1/84 – 12/31/84)," "(Toy, $100K, 1/1/87 – now)," and "(Book, $50K, 4/1/87 – now)."

CATEGORY: (Projected, Not Empty) / (Containment, Element, Explicit) / (=, Foreign) (=, Constant)

QUERY Q 4.10.2: Find the name and the budget history in 1984 and 1987 of the department being directed by Di

ANSWER: "(Toy, $150K, 1/1/84 – 7/31/84)," "(Toy, $200K, 8/1/84 – 12/31/84)" and "(Toy, $100K, 1/1/87 – 12/31/87)."

CATEGORY: (Projected, Not Empty) / (Containment, Element, Explicit) / (=, Constant)

QUERY Q 4.10.3: Find the name of the department where *ED* working at the beginning of both of the years 1986 and 1987, and the periods *ED* worked there.

ANSWER: "(Toy, 2/1/82 – 1/31/87)."

CATEGORY: (Projected, Not Empty) / (Containment, Element, Explicit) / (=, Constant)

QUERY Q 4.10.4: Find the current name of the manager *ED* had on both 1984's Christmas and his 27^{th} birthday, and the dates the manager started as a manager.

ANSWER: "(Di, 1/1/82)."

CATEGORY: (Projected, Not Empty) / (Containment, Element, User-defined) / (=, Constant) (=, Single) (=, Foreign)

QUERY Q 4.10.5: Find the department name, the then manager, the modification dates and the new values of the budget for every budget change that occurred in 1984, 1986 and 1988.

ANSWER: "(Toy, Di, $200K, 8/1/84)."

CATEGORY: (Complete, Not Empty) / (Containment, Element, Explicit) (Ordering, Interval, Computed) / (=, Single) (<>, Single)

Contributors and Acknowledgements

We wish to thank the many participants at the "ARPA/NSF International Workshop on an Infrastructure for Temporal Databases" for their numerous useful comments and insights which improved the initial test suite.

An alphabetical listing of the affiliations and e-mail addresses of the contributors follows.

J. Clifford, Information Systems Dept., New York University, `jcliffor@is-4.stern.nyu.edu`; S. K. Gadia, Computer Science Dept., Iowa State University, `gadia@cs.iastate.edu`; F. Grandi, Dip. di Elettronica Informatica e Sistemistica, Università di Bologna, Italy, `fabio@deis64.cineca.it`; P. P. Kalua, Computer Science Department, Indiana University, `kalua@cs.indiana.edu`; N. Kline, Computer Science Dept., University of Arizona, `kline@cs.arizona.edu`; N. Lorentzos, Informatics Laboratory, Agicultural University of Athens, `eliop@isosun.ariadne-t.gr`; Y. Mitsopoulos, Informatics Laboratory, Agicultural University of Athens; A. Montanari, Dip. di Matematica e Informatica, Università di Udine, `montanari@uduniv.cineca.it`; S. S. Nair, Computer Science Department, Iowa State University, `snair@cs.iastate.edu`; E. Peressi, Dip. di Matematica e Informatica, Università di Udine, `peressi@udmi5400.cineca.it`; B. Pernici, Dip. di Matematica e Informatica, Università di Udine, `pernici@uduniv.cineca.it`; E. L. Robertson, Computer Science Department, Indiana University, `edrbtsn@cs.indiana.edu`; J. F. Roddick, School of Computer and Information Science, University of South Australia `roddick@unisa.edu.au`; N. L. Sarda, Computer Science and Eng. Dept., Indian Institute of Technology, Bombay, India, `nls@cse.iitb.ernet.in`; M. R. Scalas, Dip. di Elettronica Informatica e Sistemistica, Università di Bologna, Italy, `rita@deis64.cineca.it`; A. Segev, School of Business Adm. and Computer Science Research Dept., University of California, `segev@csr.lbl.gov`; R. T. Snodgrass, Computer Science Dept., University of Arizona, `rts@cs.arizona.edu`; A. Tansel, Bernard M. Baruch College, City University of New York, `uztbb@cunyvm.cuny.edu`; P. Tiberio, Dip. di Elettronica Informatica e Sistemistica, Università di Bologna, Italy, `tiberio@deis64.cineca.it`. A. Tuzhilin, Information Systems Dept., New York University, `tuzhilin@square1.stern.nyu.edu`; G. T. J Wuu, Bell Communications Research, `wuu@ctt.bellcore.com`.