Constraints for Semistructured Data

Ekaterina Pavlova , Igor Nekrestyanov, Boris Novikov
katya@meta.math.spbu.ru, igor@meta.math.spbu.ru, borisnov@acm.org
Saint-Petersburg State University

Abstract

A taxonomy of constraints for semistructured data models is presented. Several existing classes of constrains defined for different database models, such as relational and object-oriented, are analyzed in the context of semistructured data model.

To capture dynamic aspects of consistency, data validity constraints are introduced. The most important subclass of data validity constraints, temporal constraints are examined in more detail.

1 INTRODUCTION

Although the field of semistructured data is relatively new, it has attracted many researches [10, 3, 16]. The number of research topics has been discussed extensively including development of data models and query languages for semistructured data [1, 9, 12], information extraction and integration [17, 21], web site construction and restructuring [22, 14], etc.

Semistructured data models are now used as a proven tool for specification and representation of information stored in the frame of digital libraries. For example, XML language [25, 5, 11] is rapidly gaining acceptance as a replacement of HTML.

However, dynamic aspects have not been examined extensively. In particular, consistency issues for semistructured data did not attract much attention till now. The consistency is especially important in the context of digital libraries, which are meant as somehow controlled sources of information rather than just large collections of data.

One of approaches to address consistency for semistructured data model is to adapt traditional concepts of database consistency. For example, an adaptation of multilevel transaction model to semistructured environment was proposed [18].

However, techniques from traditional databases does not suit really well for the case semistructured data due to its specifics. For example, used in relational DBMS consistency constraints are associated with tables, that constitute the data structure. But semistructured data has no clear structure.

Data model for semistructured data that allows partial and inconsistent information was proposed in [23]. Authors mostly concerned the case when inconsistency arose from distribution of information about same real object among several data sources. They applied the model to develop manipulation language for such kind of data.

An interesting comparison of concept used in relational databases and semistructured data may be found in [19].

We are looking for consistency framework for semistructured data. The notion of consistency is tightly bound to semantics of constraints on the data (the satisfaction of all constraints in DB will imply DB consistency). Therefore, investigation of constrains that are applicable to semistructured data may be very useful in order to define consistency framework.

There were several research papers on integrity constraints on semistructured data [7, 6, 2]. However these papers investigated the question of constraint implication: given that certain constraints are known to hold, does it follow that some other constraints to be satisfied? A number of decidability and undecidability results were established.

Constraints for semistructured data were also studied in the [15]. The focus of that work was on verification of integrity constrains on web sites using declarative approach.

The major focus of this work is classification of types of constraints which can be found in semistructured data. In order to do this we are reusing constraints types from structured databases, such as relational and object-oriented [8]. Because semistructured data is generalization of structured data and any structured data may be easily described as semistructured it is interesting to see how constraints on structured data will be reformulated in semistructured environment.

We divide all constraints into two major groups by their relationship with transactions:

- integrity constraints

©Вторая Всероссийская научная конференция
ЭЛЕКТРОННЫЕ БИБЛИОТЕКИ:
ПЕРСПЕКТИВНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ,
ЭЛЕКТРОННЫЕ КОЛЛЕКЦИИ
26-28 сентября 2000г., Протвино

Constraints of this kind describes semantic integrity of the data and should be preserved by transaction.

- data validity constraints

  Such constraints describes conditions of validity of data. Therefore transaction execution is affected by them.

These groups are considered in more details in following sections.

2 CONSTRAINTS

Integrity constraints are constraints which should be preserved by (update) transactions. These constraints are important because they provide some form of semantic integrity of the data.

Certain kinds of constraints found in structured databases are also common in semistructured databases. However, in standard database systems, constraints are typically expressed as part of the schema, but in semistructured data there is no explicit schema.

We shall briefly review the forms in which constraints are expressed in relational and object-oriented databases. Later, we will show how these types of constraints are represented in semistructured databases.

2.1 Constraints in Structured DBs

In structured databases, the schema serves for two purposes. First, it describes the *structure* or type of the data; second, it describes certain *constraints*. The boundary between what is a type and what is a constraint is not always well-defined and depends on the formulation of the type system. As a practical distinction, the fact that a program does not violate the type of a database can usually be checked statically by analysis of the code of the program, while constraints are usually checked dynamically when the database is updated.

2.1.1 Constraints in Relational Databases

In relational databases the type of relation describes the structure of the tuples (the field names and their types), while the constraint include assertion of the keys and inclusion dependencies.

For example, a fragment of a typical relational schema definition in a figure 1 tells us more than tuple types. First, it tell us that there is a named component of the database, `Departments`, which has this type. This is often called an *extent*. Second, it imposes a key constraint so that no two tuples have the same `DeptId` field. In the `Employees` relation we see addition of a foreign key constraint: any `DeptId` field that occur in the `Employees` relation must also occur in the `Departments` relation.

2.1.2 Constraints in Object-Oriented Databases

In the object-oriented databases the story is essentially the same as in relational ones with some additional complications [8].

Figure 2 shows an ODMG schema. In addition to object type declarations it imposes some additional constraints:

```
create table Employees (
    EmpId:  integer,
    EmpName:  char(30),
    DeptId:  integer,
    ...
    primary key(EmpId),
    foreign key(DeptId)
        references Departments(DeptId)
)
create table Departments (
    DeptId:  integer,
    ...
    primary key(DeptId)
)
```

Figure 1: Sample relational schema

```
interface Publications
    extent publication {
        attribute String title;
        attribute Date date;
        relationship set<Author> auth
            inverse Author::pub;
    }
interface Author
    extend author {
        attribute String name;
        attribute String address;
        relationship set<Publication> pub
            inverse Publication::auth;
    }
```

Figure 2: Sample ODMG schema

1. The `extent` declaration defines two persistent variables `publication` and `author` whose types are, respectively, `set<Publication>` and `set<Author>`.
   Whenever an instance of `Publication` or `Author` is created, it is automatically inserted into the `publication` or `author` set.

2. For any publication `p`, the set `p.auth` is a subset of the set `author`. Similarly, for any author `a`, the set `a.pub` is a subset of `publication`.

   Such conditions are called *inclusion constraints*.

3. For any publication `p`, and for any author `a` in `p.auth`, `p` is a member of `a.pub`. Similarly, for any author `a`, and for any publication `p` in `a.pub`, `a` is a member of `p.auth`.

   Such conditions are called *inverse relationships*.

Note that these constraints are independent of each other. None of them is a result of any subset of the others.

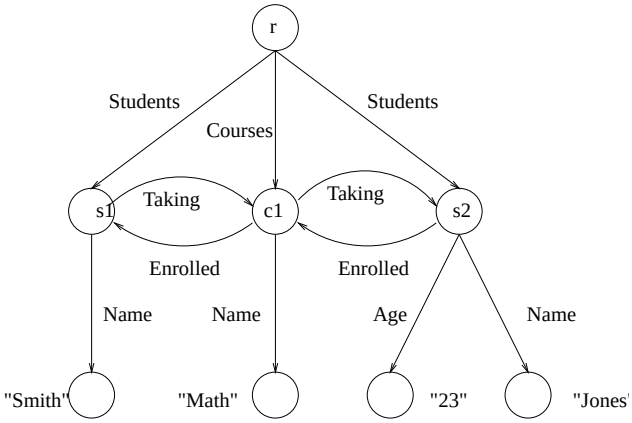

Figure 3: Sample OEM data graph

2.2 Constraints in Semi-structured DBs

Semi-structured data is a generalization of structured data in a sense that any structured data may be easily considered as semi-structured ones. In order to fully represent schema of structured data in semistructured data model we should map existing integrity constraints too. In particular, this means that we should be able to formulate constraints from structured databases in terms of semistructured data model in some generic fashion.

2.2.1 Type Constraints

In the light weight data models for semistructured data, such as OEM [1], the data are represented as labeled directed graph. In this graph only leaf nodes have values.

In our taxonomy *type constraints* are the constraints on nodes, but not on the edges of data graph. The sample type constraint is the "salary is a non-negative".

Note, that semistructured data is unconstrained by any type system or schema. As a consequence, the statement like "age is an integer", which is represented as a type in structured databases, may only be described as type constraint in semistructured database.

From other side, the object boundaries are not well-defined in semistructured data. Therefore, the statement "any student must have age", which is also represented as type definition in structured databases, is not represented as a type constraint.

Note, that some type constraints may be extracted from the data (for example, DataGuides). These constraints may be useful for query optimization purposes. However, they are not obligatory and may lost validity with data changes. This makes them useless from the point of view of consistency framework. Due to this reason here and in following we will assume that all constraints are explicitly specified.

2.2.2 Path Constraints

Unlike *type constraints*, *path constraints* are the restrictions on the data graph structure. Figure 3 shows a sample data graph. Lets see how constraints we have found in structured databases are expressed in semistructured:

- **Inclusion**

  Sample extent constraint is "person who is enrolled to the course must be a student". By taking edge label as predicates this can be stated as follows:

  $$\forall s\,(\exists c\,(Courses(r,c)\ \texttt{and}\ Enrolled(c,s))\ \rightarrow\ Students(r,s))$$

  Here r denotes the root node of data tree, and variables c, s range over vertices. This constraints states that any vertex that is reached from root by following $Courses$ edge followed by an $Enrolled$ edge can also be reached from the root by following a $Students$ edge.

- **Inverse Relationship**

  With respect to our example data graph, the inverse relationship between $Taking$ and $Enrolled$ is expressed as:

  $$\forall s\,(Students(r,s)\ \rightarrow\ \forall c\,(Taking(s,c)\ \rightarrow\ Enrolled(c,s)))$$
  $$\forall c\,(Courses(r,c)\ \rightarrow\ \forall s\,(Enrolled(c,s)\ \rightarrow\ (Taking(s,c)))$$

  The first constraints above states that for any student s and any c, if c is reachable from the s by following the $Taking$ edge, then s is also reachable from c by following $Enrolled$ edge. Second constraints states similar condition for any course s.

In general path constraint either have the form

$$\forall x\,(\alpha(r,x)\ \rightarrow\ \forall y\,(\beta(x,y)\ \rightarrow\ \gamma(x,y)))$$

or the form

$$\forall x\,(\alpha(r,x)\ \rightarrow\ \forall y\,(\beta(x,y)\ \rightarrow\ \gamma(y,x)))$$

where $\alpha(x,y)$, $\beta(x,y)$ and $\gamma(x,y)$ represent a path, i. e. a sequence of edge labels, from node x to node y.

All constraints that we have so far encountered may be expressed by constraints of the this class. Further generalization has sense only if we will find useful constraints that do not fail into specified class.

2.2.3 Complex Constraints

In practice, most of real-world logical constraints on the data do not fail into category of pure type or path constraints. Indeed such constraints are represented as complex mix of constraints of both kinds.

An example of such real-world constraint is the statement "sum of ingredients should not exceed 100%".

2.2.4 Constraints in XML

Recently, XML (eXtensible Markup Language [25]) has emerged as a standard for data exchange on the World Wide Web. While a XML-Schemas [11] may be imposed on an XML document, it is not required, and XML data is usefully treated as semistructured data.

There are a number of proposals for a type systems for constraining the structure of XML documents. At the

present they are just type systems; they do not impose constraints in the sense we have described above.

For example, XML-Data [20] may be used to describe the type of data. It is easy to imagine inverse and inclusion constraints to be added to XML-Data in much the same way that they have been expressed in object-oriented data definition languages. Apart from XML-Link [26], which can express simple co-reference constraints, there is nothing in the current proposals for inclusion or path constraints.

3 DATA VALIDITY CONSTRAINTS

In this section we discuss other type of constraints – *data validity constraints*. Unlike *integrity constraints* discussed in previous section these constraints should not be preserved by transactions. Indeed they impose some additional conditions which affect execution of transactions.

Data validity constraints describe conditions of validity of data. One of examples of data validity constraints is confidence constraints. For example, staff and customers may have different views of the same data.

However, it is impossible to classify all subtypes of data validity constraints because the notion of validity of data is application specific.

Here we mainly discuss temporal validity constraints. We also show how generic data validity constraints can be described in existing data models for semistructured data.

3.1 Temporal Validity Constraints

The example of loosing of temporal validity is data integration view. Data integration view — single view to info from external semistructured sources. Because of no notifications on updates in external sources, data in view may be obsolete or contradictory. Similar problems can be found in:

- Client-pull caches

  Web-caching provides an intermediate storage space between client and web server in order to reduce both the latency and network traffic. As downside, client may get obsolete information. To solve this in client-pull kind of caches, client periodically checks whenever external sources are updated.

- Temporal data in real-time databases

  In real-time databases both transactions and data may have some validity intervals.

3.1.1 Temporal Consistency

The term of *temporal consistency* is come from Real-Time DB systems [13, 24, 4] In such DB systems the value of objects must correctly reflect the state of environment. This means that data in DB must be fresh and also must be temporally related with each other. These requirements lead to the notion of temporal consistency [27, 28] which consists of two components:

- **Absolute consistency**
  Each value of data item has some absolute validity interval in which it is considered to be correct.

- **Relative Consistency**
  Relationships among data items are essential for calculation of dataset validity interval.

  More formally the relative validity interval is associated with set of data items. The difference of timestamps of each pair of data items should not exceed the value of relative validity interval.

Data on highly dynamic web-sites (like stock rates) have natural dependency on time. Therefore time constraints for semistructured data becomes essential. Such constraints can be expressed in pretty much similar way to Real-Time databases. This means that absolute and relative constraints can be associated with semistructured data.

3.2 How to Describe Validity Constraints in Existing Models

In structured databases constraints are mostly part of the schema. But this is not the case with semistructured databases because they have no schema. However, there exist several known data models for semistructured data in which some constraints can be easily described.

One of the models which can be used for this is DOEM [9]. DOEM extends OEM [1] with special annotations on edges to record information about updates; in particular, the time and kind of update. This permits a history of changes to a semistructured data to be maintained.

The DOEM model allows to capture some semantics of relative temporal consistency by comparing timestamps which are saved in annotations on edges. It also allows to speak about temporal consistency under assumption that all data is up to date (i.e. satisfy absolute consistency).

DOEM model is illustrated by example on Figure 4.

Yet one suitable model to describe validity constraints is MOEM model [12], This model is extensible, semistructured data model that generalizes existing lightweight semistructured models. In this model, each label is a set of descriptive properties. A property is a kind of meta-data. Typical properties are name of the edge and the level of security that protects the edge, but any property can be used in a label to describe the nodes that are reachable through that edge.

This model provides much better possibilities to capture data validity constraints. We can easily describe absolute validity interval just by adding the corresponding property to the labels. The example of how temporal constraints can be described in this model is shown on Figure 5.

4 CONCLUSION

In this paper we studied constraints that are meaningful for semistructured data. Constraints are important because they are essential part of consistency framework.

We have divided all constraints on data into two groups — integrity constraints and data validity constraints.

Integrity constraints describes semantic integrity of the data. Because transactions are intended to preserve integrity of the data they should preserve constraints of this kind.

Among integrity constraints we distinguish type and path constraints. Any real-world constraint is usually represented by some mix of constraints of these types.

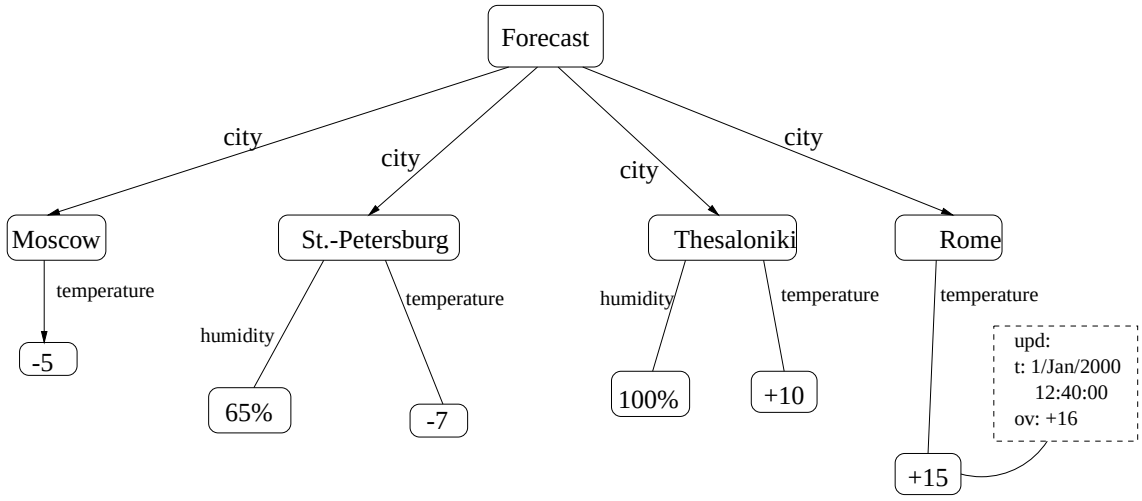

Figure 4: Capturing data validity constraints in DOEM

Data validity constraints describes conditions of validity of data. In order to produce consistent result transactions must also take them into account. In other words, transaction execution is affected by constraints of this kind.

It is impossible to make a complete taxonomy of data validity constraints because the notion of validity of data is application specific. We specially distinguish one important subclass of data validity constraints — temporal validity constraints.

In structured databases constraints are mostly part of the schema. But this is not the case with semistructured databases because they have no schema. However, there were proposed several data models for semistructured data in which some constraints can be easily described.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1:68–88, April 1997.

[2] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proc. of the PODS'1997*, 1997.

[3] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web*. Morgan Kaufmann Publishers, 1999.

[4] Azer Bestavros, Kwei-Jay Lin, and Sang Hyuk Son. *Real–Time Database Systems: Issues and Applications*. Kluwer Academic Publishers, 1997.

[5] Angela Bonifati and Stefano Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, 29(1), March 2000.

[6] Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints on semistructured and structured data. In *Proc. of the PODS'1998*, June 1998.

[7] Peter Buneman, Wenfei Fan, and Scott Weinstein. Interaction betwen path and type constraints. In *Proc. of the PODS'1999*, 1999.

[8] R. G. G. Cattell, editor. *The Object Database Standard: ODMG-93 (Release 1.2)*. Morgan Kaufmann Publishers, 1996.

[9] S. Chawathe, S. Abiteboul, and J. Widom. Representing and querying changes in semistructured data. In *Proceedings of the Fourteenth International Conference on Data Engineering*, February 1998.

[10] Sophie Cluet and Tova Milo, editors. *ACM SIGMOD Workshop on The Web and Databases (WebDB)*, Philadelphia, Pennsylvania, USA, June 1999.

[11] Lee Dongwon and W. Chu Wesley. Comparative Analysis of Six XML Schema Languages. *SIGMOD Record*, 29(3), September 2000.

[12] Curtis E. Dyreson, Michael H. Bohlen, and Christian S. Jensen. Capturing and querying multiple aspects of semistructured data. In *Proc. of the 25th VLDB Conference*, Edinburgh, Scotland, 1999.

[13] Joakim Eriksson. Real-Time and Active Databases: A Survey. In *Active, Real-Time, and Temporal Database Systems, Second International, ARTDB-97*, pages 1–23, Como, Italy, September 1997.

[14] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Strudel: a web-site management system. In *Proc. of the ACM SIGMOD conference on Management of Data*, 1997.

[15] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. Verifying integrity constraints on web sites. In *Proc. of the IJCAI'99*, August 1999.

[16] Michael J. Franklin (ed.). Management of semistructured data. *SIGMOD Record*, 26(4), 1997.

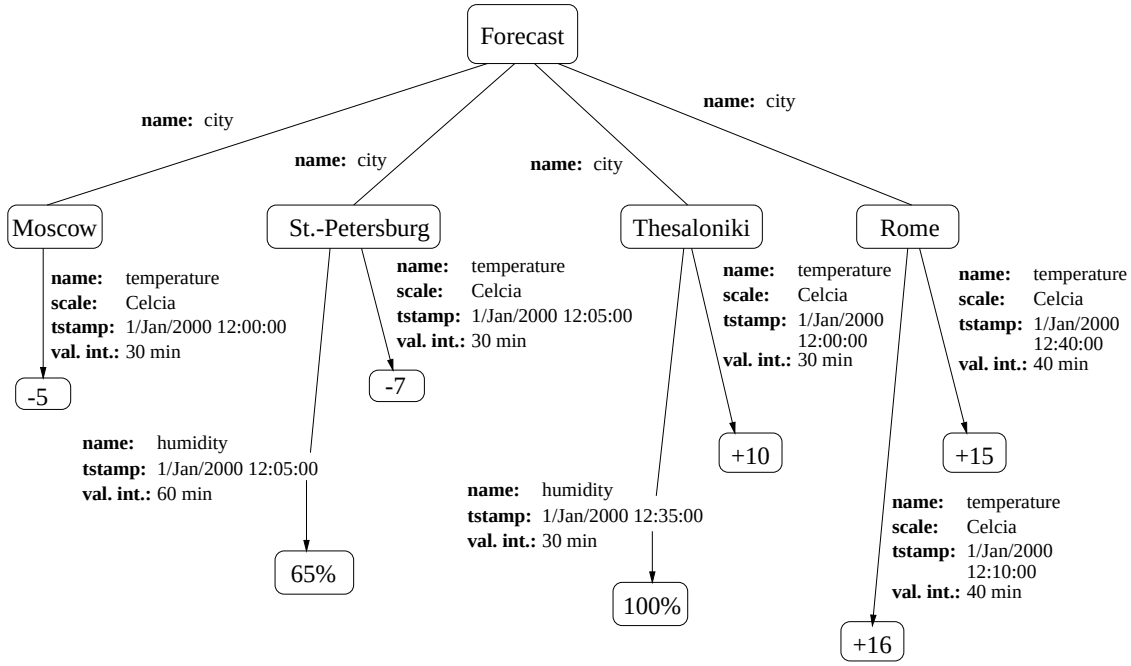

Figure 5: Capturing data validity constraints in MOEM

[17] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. In *Proceedings of Second International Workshop on Next Generation Information Technologies and Systems*, June 1995.

[18] Ekaterina Gorshkova, Igor Nekrestyanov, Boris Novikov, and Pavlova Ekaterina. Looking for consistency of semi-structured data. In *Proc. of the Russian Digital Libraries'1999*, October 1999.

[19] Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. X-Ray – Towards Integrating XML and Relational Database Systems. Technical report, Johannes Kepler University Linz, July 2000.

[20] A. Layman, E. Jung, E. Maler, H. J. Thompson, J. Paoli, J. Tigue, N. J. Mikula, and S. De Rose. *XML Data*. World Wide Web Consortium, 1998.

[21] Giansalvatore Mecca and Paolo Atzeni. Cut and paste. *Journal of Computer and System Sciences (JCSS)*, 58(3):453–482, 1999.

[22] Giansalvatore Mecca, Paolo Merialdo, Paolo Atzeni, and Valter Crescenzi. The (short) Araneus guide to web-site development. In *Proc. ot the WebDB*, pages 13–18, 1999.

[23] Liu Mengchi and Wang Ling Tok. A Data Model for Semistructured Data with Partial and Inconsistent Information. In *Proc. of the 7th International Conference on Extending Database Technology (EDBT 2000)*, Konstanz, Germany, April 2000. LNCS. Springer-Verlag.

[24] J. Stankovic, S. H. Son, and J. Hansson. Misconceptions About Real-Time Databases. *IEEE Computer*, 32(6):29–36, June 1999.

[25] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0*, February 1998.

[26] World Wide Web Consortium. *XML Linking Language (XLink)*, July 1999.

[27] M. Xiong, K. Ramamritham, J.A. Stankovic, D. Towsley, and R. Sivasankaran. Maintaining Temporal Consistency: Issues and Algorithms. In *First International Workshop on Real-Time Databases*, Newport Beach, California, March 1996.

[28] Ming Xiong, Raju Sivasankaran, Jack Stankovic, Krithi Ramamritham, and Don Towsley. Scheduling Transactions with Temporal Constraints: Exploiting Data Semantics. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, Washington, DC, December 1996.