Accessing Imprecise Data: An Approach Based on Intervals

Roger S. Barga and Calton Pu

Technical Report No. CS/E 93-005

March 1993

Accessing Imprecise Data: An Approach Based on Intervals

Roger S. Barga, Calton Pu
Dept. of Computer Science and Engineering
Oregon Graduate Institute
Internet: {barga,calton}@cse.ogi.edu

Abstract

In many real world applications (even in banking), imprecise data is a matter of fact. However, classic database management systems provide little if any help in the management of imprecise data. We are applying methods from interval arithmetic, epsilon serializability, and other related areas to help the application designers in the management of naturally imprecise data. Our approach includes operators on imprecise data that give bounds on the result imprecision and algorithms that constrain the imprecision propagation in the database.

1 Introduction

Traditional database management systems provide support only for precise data, though in the physical world data is often imprecise. An important class of examples is the scientific data such as incomplete recording of data, instrument noise, measurement error, computational model imprecision, and data aggregation of one kind or another. Another example is the "fuzzy" data managed by Epsilon Serializability algorithms [PL91, DP93]. In these cases, the application designer is left with the unpleasant decision whether to ignore the imprecise information and store some approximation to the precise value, or to manage the imprecision by hand. Our objective is to provide database management systems that support both precise and imprecise data directly.

In many situations where precise data is not available, much more useful information on the imprecision other than 'value unknown' or 'predicate is possibly true' is available. Different approaches to the problem of managing imprecise information have been proposed. These range from a consistent way of handling incomplete information [Lip79] to recent work on fuzzy data models [SMF90] and probabilistic data models [BGMP92]. One of the most common and useful forms of information available to an application designer is the ability to bound the amount of imprecision or error in the value of a data item. We think that it would prove valuable if the database management system was able to represent and store this type of information on imprecision, as well as provide operators to manipulate the imprecision bounds in a consistent and useful manner. In our examples, the most natural way to represent imprecise data is through an *interval*. For simplicity of presentation we are concerned with numerical data only in this paper.

The immediate problems we wish to address here are how to represent imprecise information (particularly bounded imprecision), the development of a data model for imprecise information, and how to manipulate imprecise data in a consistent manner, particularly when a database state is being transformed. Our basic strategy for dealing with bounded imprecision is to permit each component of a tuple to be an interval of values drawn from the corresponding database domain. We are working on an algebra for interval relations as an extension of relational algebra, using methods from interval arithmetic to manipulate the imprecision bounds.

2 Interval Relational Data Model

The interval relational data model is defined as an enhanced relational database that allows imprecise attribute values and imprecise queries; both are expressed in terms of intervals. As in classical relational database theory, an interval relational database is defined as a set of relations where each relation is a set of tuples. The key departure of the interval relational data model is that tuple components are not confined to single elements drawn from the underlying domain; instead, as an attribute value, an interval is used to represent imprecise data. An example is presented in Figure 1. An interval database

Time	Temperature	Wind Speed	Humidity
1230	[23...25]	[7...10]	[0.10...0.16]
1400	[28...42]	[13...32]	[0.22...0.34]
1530	[44...48]	[9...14]	[0.48...0.62]

$\text{SELECT}_{Temperature \subset [20...45]}$ WITH PRECISION 0.90

Figure 1: Interval relation and query.

consists of relations, where a relation $R(t_1, \ldots, t_n)$ is a subset of the Cartesian product $I_1 \times I_2 \times \cdots \times I_n$ of interval domains I_i; each I_i is the Cartesian product $D_i \times D_i$ constructed from an attribute domain D_i $(1 \leq i \leq n)$. A pair $(d, d') \in I_i$ represents the interval $[d, d']$, with an interval width of zero $(d = d')$ representing a precise value. In the interval relational data model, key attributes require precise values.

It is the ability to use an interval of values from a domain as a component in a tuple that provides us with the basic structure for storing imprecise information. An interval representation also provides an intuitively appealing scheme for understanding imprecise information. The width of the interval of a tuple component is a measure of the imprecision of the stored information: the larger the interval the more imprecise the information. The intuition is that for the Ith component of a tuple, an interval containing the entire domain D_i corresponds to information of the worst precision, while an interval of width zero in D_i represents information of the highest precision[1].

With imprecise values specified using intervals, a measure of relative imprecision can be derived. We give just one example here, although many other useful alternative definitions exist.

Definition 1 (Relative Imprecision): The relative imprecision of an interval data element is defined from a maximum imprecision bound Δ_{max},where $\Delta_{max} = max(\mid val(Attr) - low(Attr) \mid, \mid val(Attr) - High(Attr) \mid)$, by $\varepsilon = \frac{\Delta_{max}}{|val(Attr)|}$.

where $val(Attr)$ is the value of the data element stored in the database, without the imprecision bounds.

As a consequence of being able to quantify the relative imprecision of the values assigned to data elements, a tuple may be selected from an interval relation based on the precision of the data it contains. This affords the application designer the ability to control the quality of data used in further experiments and decision making. In addition, bounds on the imprecision of data values is available to bound imprecision in results derived from simulation models or aggregate functions which use this data as input.

3 Interval Relational Algebra

An interval relational algebra is proposed as an enhancement of the relational algebra. The fundamental interval relational algebra operators are UNION, DIFFERENCE, CARTESIAN PRODUCT, PROJECT, and SELECT. These operators are the interval counterparts of the primitive operations in the classical relational algebra, and are intended to directly extend the corresponding precise classical operations

[1]The problem of missing information is outside the scope of this paper. However, it is conceivable that one could represent the fact that an attribute is *"missing but applicable"* using an interval that contains the entire domain D_i.

to the 'non-precise' attributes of the interval relation. We limit our discussion here to the SELECT operation. (Other interval relational algebra operators are discussed in [BP93].)

The syntax of the SELECT operation at level of precision P applied to an interval relation I with selection condition C is: $\textbf{SELECT}_C(I)$ WITH PRECISION P. The selection condition C is a Boolean expression specified on the attributes of interval relation I. The Boolean expression is made up of a number of clauses of the form *Arg op Arg*, where *Arg* is either a constant interval value or the name of an attribute in an interval relation, and *op* is one of the comparison operators $\{ \in, =, <, \leq, \geq, >, \neq, \supset, \subset \}$. Clauses can be arbitrarily connected by the logical connectives **AND**, **OR**, and **NOT** to form a general selection condition.

Many applications often require query capabilities involving aggregate and statistical functions. The interval representation allows for a natural definition of a full range of aggregate functions over imprecise data: *max, min, mean, count, avg, sum*, and *product*, based on interval arithmetic. We discuss this in detail in [BP93].

4 Bounding the Imprecision on Data

When a data element acquires imprecision due either to external directive, or from the result of a transaction (e.g., UPDATE operations), importing and exporting imprecise values, we would like the DBMS to update and maintain bounds on the amount of imprecision introduced for each data item. Using our interval representation, the amount of imprecision imported by a transaction can be recorded in the database as the interval bounds of an attribute, or the bounds of an existing interval value can be updated following the rules of interval arithmetic.

Classic transaction models do not include inconsistencies such as data imprecision, since a transaction is defined as a program that transforms a consistent database state into another consistent state. We have introduced the concept of *epsilon serializability* [PL91] (ESR) as a family of correctness criteria that bound the amount of inconsistency introduced into transactions. Efficient divergence control algorithms [PHK+93] guarantee epsilon serializability at a cost close to that of classic concurrency control algorithms, from which the divergence control algorithms were derived. In addition to transaction control, consistency restoration algorithms [DP93] can reduce the inconsistency introduced into the database, given some information on the correction of data values.

Divergence control and consistency restoration both support the interval notation of data imprecision. Although originally developed to extend classic (commercial) transaction processing, these algorithms are being adapted for bounding the imprecision on data in general, including scientific data and in particular, the interval relational data model sketched in the previous sections. To achieve this, we specify the amount of imprecision allowed on each data object, denoted as data-ϵ-spec, in analogy to the specification on the amount of allowed transaction inconsistency, denoted as trans-ϵ-spec, in epsilon transactions (ETs). Each data item may contain some imprecision (stored with the interval value and managed by the Interval Relational DBMS), limited by its data-ϵ-spec.

When ETs access imprecise data in an interval relational DBMS, the data imprecision is accumulated by the ET. If the imprecision is tolerable (compared to trans-ϵ-spec) then the ET commits; The query algebra of the interval relational DBMS can be used to later retrieve the imprecise data values committed by transactions relaxed with ESR. Otherwise, the ET may wait, abort, or trigger a consistency restoration method. ESR algorithms are then used to constrain the imprecision propagation.

As an example, consider the collection and processing of environmental data from a number of remote sensing devices. Each device introduces instrument noise and measurement imprecision, which can be quantified and used to bound the precise values for the collected data. Differences in instrument type, quality, and placement also effect the precision of the data and result in varying width interval bounds. Using an ESR-based transaction processing system, the bounded amount of imprecision may be recorded into an update transaction and stored in the interval relational DBMS. Additional processing of the imprecise data values, such as aggregate operations to compute the average temperature or total precipitation over the monitored area, using only the interval relational algebra operators ensures that interval bounds will be properly maintained. The precision of the data retrieved from the collection can be specified by providing a precision requirement along with a query. ESR divergence control algorithms will maintain the amount of imprecision in the query result within this limit.

5 Closing Remarks

In this paper, we have proposed the concept of the interval relation to capture the natural imprecision in much of real world data. The interval relational data model and algebra have both an intuitive appeal and a simple representation. We are developing interval analogs to the fundamental relational algebra operations and new operators that have no counterpart in classical relational systems. We regard *imprecision* and *uncertainty* as two complementary aspects of imperfect information we have of the physical world. Imprecision refers to the value of a data object, while uncertainty refers to the degree of confidence in the value assigned to an attribute. The ability to represent and manage both imprecision and uncertainty in a database has considerable promise for many practical applications such as scientific data management.

Ongoing research on Epsilon Serializability has produced divergence control algorithms [WYP92, PHK+93] and consistency restoration algorithms [DP93] that preserve bounded inconsistency in data. Queries on data with bounded inconsistency can be formulated easily using our interval algebra. We are actively investigating other aspects of managing imprecision as well as methods to manage uncertainty information, and expect to incorporate both into our system in the future.

References

[BGMP92] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):487–502, October 1992.

[BP93] R. Barga and C. Pu. Handling inconsistency in scientific data management: An approach based on intervals. Technical Report OGI-CSE-93-005, Department of Computer Science and Engineering, Oregon Graduate Institute, 1993.

[DP93] P. Drew and C. Pu. Asynchronous consistency restoration under epsilon serializability. Technical Report OGI-CSE-93-004, Department of Computer Science and Engineering, Oregon Graduate Institute, 1993. Also available as tech. report HKUST-CS93-002, Department of Computer Science, Hong Kong University of Science and Technology.

[Lip79] W. Lipski. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4(3):262–296, September 1979.

[PHK$^+$93] C. Pu, W.W. Hseush, G.E. Kaiser, P. S. Yu, and K.L. Wu. Distributed divergence control algorithms for epsilon serializability. In *Proceedings of the Thirteenth International Conference on Distributed Computing Systems*, Pittsburgh, May 1993.

[PL91] C. Pu and A. Leff. Replica control in distributed systems: An asynchronous approach. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 377–386, Denver, May 1991.

[SMF90] S. Shenoi, A. Melton, and L.T. Fan. An equivalence classes model of fuzzy relational databases. *Fuzzy Sets and Systems*, 38:153–170, 1990.

[WYP92] K.L. Wu, P. S. Yu, and C. Pu. Divergence control for epsilon-serializability. In *Proceedings of Eighth International Conference on Data Engineering*, pages 506–515, Phoenix, February 1992. IEEE/Computer Society.