December 1998

A Brief Overview and Illustrative Application of Interval Databases

John Starner
University of Texas at El Paso

Kurt Pflughoeft
University of Texas at El Paso

Peeter Kirs
University of Texas at El Paso

A Brief Overview and Illustrative Application of Interval Databases

John Starner
Peeter Kirs
Kurt Pflughoeft
University of Texas at El Paso

Abstract

The value of interval data and interval databases has been recognized by other disciplines for quite some time. Very little attention has been given to the application of interval databases, however. This article provides an overview of how interval databases can be manipulated and provides an illustrative example of the usefulness of such databases.

Introduction

The underlying concepts of interval databases have been extensively investigated from a theoretical perspective. Mathematicians have long recognized the usefulness of intervals and interval arithmetic, and their properties are well defined. More recently, computer scientists have considered the theoretical formulations necessary to operationalize interval databases in relational format (Ariav, 1986; Sarda, 1990; Lorentzos, 1993). It is only in the recent past, however, that efforts have been directed toward investigation of potential applications of interval databases, although the majority of these studies have focused their attention on such issues as feasibility and efficiency (Starner, 1996; Lorentzos, 1997). Very little examination of the useful of interval database in real-world applications has been conducted. Further, little or no empirical investigation of the effect of database manipulations on interval data series has been performed.

In this paper, we provide a brief overview of interval databases and their usefulness in a variety of settings. We then present some results obtained in the manipulation of an interval database when standard queries are applied. Due to space limitations, only one query, and outcomes obtained, will be discussed.

Intervals

An interval is a generic term which defines the maximum and minimum limits between which a data series can range (in the case of numeric data), or specifies a defined set of values which can exist with a sequence (in ordinal or alphabetic series). The use of intervals in everyday life are quite common. We refer to *teenagers* as those individuals whose ages range between 13 and 19; an *average IQ* might imply that the scores obtained from standardized testing range between 85 and 115; the *Viet Nam Era* is a term applied to a specific time period; for the sake of convenience, we frequently segregate people waiting in a queue by their last names (e.g., those with last names starting with 'F' - 'L'). Notice that the intervals may be either continuous (teenagers and the Viet Nam Era), discrete (IQs), or alphabetical (Last name).

Storing Interval Data

Manipulation of interval data can not be adequately achieved by merely storing the lower and upper bounds of an interval as two separate fields in a table (one for the lower and one for the upper). Such an approach places severe limitations on the results obtained, especially when attempting to join two tables.

Consider the following two tables:

TAXES

amt	*tax*
[0.01, 0.11]	.01
[0.12, 0.23]	.02
[0.24, 0.36]	.03
[0.37, 0.49]	.04
[0.50, 0.62]	.05
[0.63, 0.75]	.06
[0.76, 0.88]	.07
[0.89, 1.01]	.08
• • •	•

INVENTORY

p#	*qty*	*price*
01	10	[.09, .15]
02	13	[.24, .32]
03	15	[.47, .56]

Table TAXES show the amount tax payable at an eight percent (8%) tax rate. Table INVENTORY represents the prices paid for each corresponding part (i.e., since they were purchased at different points in time, the unit price varies).

One concern might be the total tax liability of the inventory on hand. Typically, such a question would be answered by *joining* the tables on a common field. Given that the join field for these tables is an interval data type, however, unique problems will be encountered.

Before proceeding, consider the use of joins on scalar valued tables:

TABLEA

ColAA	ColAB
A	1
B	1
C	2
D	4

TABLEB

ColBA	ColBB
1	X
2	Y
3	Z

ColAB (in TABLEA) and COLBA (in TABLEB) are of the same data type. Performing natural *inner*, *left*, *right* and *outer joins* would result in the following:

TABLEA *inner join* TABLEB

A	1	X
B	1	X

Only those rows that match the join column are retained.

TABLEA *left join* TABLEB

A	1	X
B	1	X
C	2	Y

All rows of TABLEA are used; Corresponding values in TABLEB are added where possible (else NULL)

TABLEA *right join* TABLEB

A	1	X
B	1	X
C	2	Y

All rows of TABLEB are used; Corresponding values in TABLEA are added where possible (else NULL)

TABLEA *right join* TABLEB

A	1	X
B	1	X
C	2	Y
D	4	--

All rows of TABLEB *AND* TABLEB are used; Corresponding values are added where possible (else NULL)

All four of these joins are *equi-joins*. That is, they are based on retaining rows when then values in the join columns of the two tables are equivalent.

Given that tables TAXES and INVENTORY have an interval data series as their common field, joining can become considerably more complex. While the *price* interval in INVENTORY for *p#02* ([0.24, 0.32]) falls within the *tax* range associated with the value 0.03 in table TAX ([0.24, 0.36]), the same can not be said for the other two parts. For both of these parts, the intervals overlap two tax rates, and thus the rows of both must be retained.

To determine the tax liability of the entire inventory, the following SQL command might be issued:

```
SELECT SUM(INVENTORY.qty *
           TAXES.tax)
FROM price INNER JOIN TAXES
           ON price, amt;
```

This command will multiply the quantity on hand (*qty*) by the tax rate (*tax*) only when a direct correspondence is found. However, in our example, there are no matches, and hence the query would result in a value of zero (0).

Since a part can be associated with two (or more) tax rates associated, we must examine the overlap between common fields. Consider the following command:

```
SELECT SUM(INVENTORY.qty *
           TAXES.tax)
FROM price (OVERLAP INNER) JOIN
           TAXES ON price, amt;
```

If we attempt to evaluate the expression as a simple scalar overlap, we find:

Part	Liability
01	10 * .01 = .10
01	10 * .02 = .20
02	13 * .03 = .39
03	15 * .04 = .60
03	15 * .05 = .75
Sum	2.04

As noted, parts 01 and 03 overlap two tax rates, and thus are included twice in the total. Obviously, the solution reached is unacceptable.

Because we are dealing with intervals, the result should also be an interval, even though it is computed as the product of two scalar variables.

Part	Minimum	Maximum
01	10 * .01 = .10	10 * .02 = .20
02	13 * .03 = .39	13 * .03 = .39
03	15 * .04 = .60	15 * .05 = .75
Interval	1.09	1.34

While obtaining an interval ([1.09, 1.34]) as a result might initially seem to be a less than optimal result, in fact, it is the only reasonable solution.

Conclusions

This paper has discussed some of the applications for interval data series in a database, the usefulness of such data, and has given one example of the results which can be obtained through querying the database. Considerably more research is necessary to determine the efficacy of applying other queries to other interval data series.

References

Available upon request.