Reliable Computing 2 (4) (1996), pp. 373–381

Some examples using the interval data type in the relational database model

John W. Starner

This paper is devoted to the idea of using interval data in relational databases. We show, on several simple examples, that interval data can be added to the query language SQL and used with a great effect in business applications of relational databases.

Since the examples are simple, they do not require any new mathematical results or non-trivial algorithms, only standard formulas of interval arithmetic.

The intended audience includes both interval researchers and specialists in relational databases. For researchers who are not well familiar with interval computations, we have included the detailed description of interval arithmetic (that can be skipped by other readers).

Примеры использования интервального типа данных в модели реляционной базы данных

Дж. Старнер

В работе рассматривается возможность использования интервальных данных с реляционными базами данных. На нескольких простых примерах показано, как интервальный тип вводится в язык запросов SQL и какую пользу может принести использование интервалов в деловых применениях реляционных баз данных.

Простые примеры, приводимые в статье, не потребовали никаких новых математических средств или нетривиальных алгоритмов. Использовались только стандартные формулы интервальной арифметики.

Работа может быть интересна как исследователям, занимающимся интервальной математикой, так и специалистам по реляционным базам данных. Для тех, кто слабо знаком с интервальными вычислениями, приводится подробное введение в интервальную арифметику, которое более подготовленные читатели могут пропустить.

1. Introduction: formulation of the problem and the main idea

1.1. Relational databases: brief introduction

Many decision support systems use *relational databases* to represent data (see, e.g., [3]):

- A relational database is a collection of *tables*. (Tables are also called *relations*; hence the name of these databases.)
- Each table has several columns corresponding to different *attributes*; each table consists of several *records*.

- Each record describes an object by describing the values of the corresponding attributes.

Example. We can have a table of *tanks* with three attributes: tank number, tank location, and tank contents. Then, a tank number T1 located in location A and containing 60 gallons will be described by a three-component record $(T1, A, 60)$.

To make this information useful, we must be able to *maintain* the tables (e.g., add, delete, and update information if necessary), and to *answer queries*. A typical query asks for all the objects that have certain values of their attributes. For example, for the above tanks table, if we want to fill the truck located in city A with 55 gallons of the substance, and we would like to use one tank only, we will thus be interested in all table in location A that have at least 55 gallons. We may also be interested in more complicated queries, such as the total amount of the substance in location A.

In this example, we considered a *simplified* case in which all the data is stored in a *single* table. In reality, different parts of information about the same data can be stored in *different* tables. For example, we may have one table in which we store the the tank number and its location, and another table in which we store the tank number and the tank contents. In this case, the information about the tank T1 is stored in two records from these two tables: $(T1, A)$ and $(T1, 60)$. If the information is stored in two or more tables, then, to pick the objects with the desired values of attributes, we need to look over several tables. The corresponding operations are called *θ-restrict* and *θ-join*.

The complete set of possible operations, together with their interpretation, is called a *relational database management system model* (RDBMS).

To handle relational databases in a user-friendly manner, several languages have been proposed. One of the most widely used of these languages is the Structured Query Language, SQL. In this language, there are three basis types of attribute values: strings, numbers (integer and real), and Boolean (true or false).

1.2. It is desirable to add intervals to relational databases

In many real-life situations, we do not know the exact values of some numerical attributes; instead, we know the *interval* $\mathbf{x} = [\underline{x}, \overline{x}]$ of possible values.

Example. If we know the *exact* contents of tank T1, i.e., if we know that this tank contains exactly 60 gallons, then we represent this knowledge as a record $(T1, A, 60)$.

In reality, however, we rarely know the exact value. Most frequently, we only know the *lower* and *upper estimates* of this contents. For example, we may not know the exact contents of the tank T1; we may know only that this tank contains between 50 and 70 gallons, i.e., that the actual contents belongs to the interval $[50, 70]$. In this case, the information about the tank T1 is naturally represented by a record $(T1, A, [50, 70])$.

In other words, we would like to use *intervals* as attribute values. Since SQL and other relational query languages do not allow that, it is therefore desirable to add intervals to relational databases.

1.3. It is possible to add intervals to relational databases

The question of adding new data types to the relational model was analyzed by Date [2]; according to Date, it is, in principle, possible to add an arbitrary data type that has well-defined *arithmetic* operations and *comparison* operations (similar to $<$, $=$, etc.).

Arithmetic operations. How to define *arithmetic* operations for interval data type in relational databases? For example, what is the interval analogue of sum? If we know the exact values of two quantities x and y, and we are interested in their sum, then the value of this sum is $x+y$. If, on the other hand, the only information that we have about the value x is that x belongs to the interval $\mathbf{x} = [\underline{x}, \overline{x}]$, and the only information that we have about the value y is that y belongs to the interval $[\underline{y}, \overline{y}]$, what can we say about the sum $x+y$ of these two quantities? It is easy to show that this sum $x+y$ can take any values from the interval $[\underline{x}+\underline{y}, \overline{x}+\overline{y}]$. Thus, the sum operation for intervals (that corresponds to adding the unknown actual values) can be described as follows:

$$[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x}+\underline{y}, \overline{x}+\overline{y}].$$

Similarly, all other arithmetic operations can be defined (see, e.g., [4]):

$$\begin{aligned}
[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] &= [\underline{x}+\overline{y}, \overline{x}-\underline{y}];\\
[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] &= [\min(\underline{x}\cdot\underline{y}, \underline{x}\cdot\overline{y}, \overline{x}\cdot\underline{y}, \overline{x}\cdot\overline{y}), \max(\underline{x}\cdot\underline{y}, \underline{x}\cdot\overline{y}, \overline{x}\cdot\underline{y}, \overline{x}\cdot\overline{y})];\\
1/[\underline{x}, \overline{x}] &= [1/\overline{x}, 1/\underline{x}] \quad (\text{if } 0 \notin [\underline{x}, \overline{x}]);\\
[\underline{x}, \overline{x}]/[\underline{y}, \overline{y}] &= [\underline{x}, \overline{x}] \cdot (1/[\underline{y}, \overline{y}]).
\end{aligned}$$

The formulas for multiplication and division can be further simplified in the (frequent) case when both intervals only contain non-negative numbers; in this case,

$$\begin{aligned}
[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] &= [\underline{x}\cdot\underline{y}, \overline{x}\cdot\overline{y}];\\
[\underline{x}, \overline{x}]/[\underline{y}, \overline{y}] &= [\underline{x}/\overline{y}, \overline{x}/\underline{y}].
\end{aligned}$$

For example, if $\mathbf{x} = [2.0, 3.5]$ and $\mathbf{y} = [0.2, 0.4]$, then $\mathbf{x}+\mathbf{y} = [2.2, 3.9]$, $\mathbf{x}-\mathbf{y} = [1.6, 3.3]$, $\mathbf{x}\cdot\mathbf{y} = [.4, 1.4]$, and $\mathbf{x}/\mathbf{y} = [5.0, 17.5]$.

Comparison operations. For intervals, *comparison* operators are also well known. Actually, for intervals, the set of possible comparison operators is much richer: If we know the *exact* values of the quantities x and y, then we have only three possible relations between them: $x < y$, $x = y$, and $x > y$. In a more realistic situation, when we only know *intervals* $\mathbf{x} = [\underline{x}, \overline{x}]$ and $\mathbf{y} = [\underline{y}, \overline{y}]$ of possible values, we have more options; for example:

- if $\underline{x} > \overline{y}$, then x is *necessarily* greater than y; this is denoted by $\Box(\mathbf{x} > \mathbf{y})$;
- if $\overline{x} > \underline{y}$, then x is *possibly* greater than y; this is denoted by $\Diamond(\mathbf{x} > \mathbf{y})$;
- if the intervals $\mathbf{x}$ and $\mathbf{y}$ have a non-empty intersection (overlap), then x is *possible* equal to y; this is denoted by $\Diamond(\mathbf{x} = \mathbf{y})$.

In addition to these (more traditional) comparison operators that compare intervals as possible *values*, we can compare these interval as *sets*: e.g., we can check whether $\mathbf{x}$ is a subset of $\mathbf{y}$, i.e., whether $\mathbf{x} \subseteq \mathbf{y}$.

1.4. What we are planning to do

In the following text, we will show, on several realistic examples, that adding the interval data types to relational databases can be really helpful in solving real-life problems.

These examples do not require any new mathematical results or any sophisticated interval algorithms. In all these example, even the use of the "naive" interval arithmetic turns out to be beneficial.

2. Application examples

Let us first consider the case in which we have only one table TANK, with three attributes described above:

Tank#	Location	Contents (gals.)
T1	A	[50, 70]
T2	B	[65, 85]
T3	A	[20, 25]
T4	C	[150, 170]
T5	B	[10, 15]
T6	A	[30, 40]
T7	C	[25, 30]
T8	D	[80, 95]

Table 1. TANK

Example 1. The real-life problem is: *What size tanks would be needed if the inventory were consolidated at each location?* In *interval* terms, this means that we must compute the sum of the TANK.Contents attribute of this table for each possible location.

The corresponding SQL query is:

SELECT TANK.Location, **SUM**(TANK.Contents)
FROM TANK
GROUP BY TANK.Location;

The result is shown in Table 2:

Location	SUM(Contents)
A	[100, 135]
B	[75, 100]
C	[175, 200]
D	[80, 95]

Table 2.

Comment. Notice that the SQL query closely matches the original statement of the problem.

Example 2. *Given that the cost of 1 gal. of the solvent is between $1.50 and $1.65, what is the total value of the inventory of the solvent?* In interval terms, the answer is an interval that

is computed as the interval [1.50, 1.65] times the interval sum of the Contents attributes for all of the rows in the Table 1.

The corresponding SQL query is:

SELECT [1.50, 1.65] * **SUM**(TANK.Contents)
FROM TANK;

Here, SUM(TANK.Contents), through a series of interval additions, produces the interval [430, 530]. The value of the inventory is then equal to the interval [1.50, 1.65] · [430, 530], i.e., to [$645.00, $874.50].

Example 3. *Find all the tanks which may have more than 55 gallons.* In interval terms, the answer is the list of all tanks for which 55 is possibly smaller than TANK.Contents.

The corresponding SQL query is:

SELECT TANK.T#
FROM TANK
WHERE ◊(55 < TANK.Contents);

The result is the Table 3:

Tank #
T1
T2
T3
T4

Table 3.

Let us now consider more complicated examples in which the knowledge is contained in *two* tables: PROJECT (Table 4) is a table with attributes P#, Status and Score, and RATE (Table 5) is a table with attributes Rate (an interval), and Bonus.

P#	Status	Score
P1	3	3.56
P2	4	4.02
P3	2	1.87
P4	3	2.65
P5	1	3.97
P6	5	4.78
P7	3	2.53
P8	4	3.16
P9	2	2.17
P10	1	3.15
P11	5	3.46
P12	3	4.05

Table 4. PROJECT

Rate	Bonus
[0.0, 0.49]	0
[0.5, 0.99]	5
[1.0, 1.49]	7
[1.5, 1.99]	10
[2.0, 2.49]	15
[2.5, 2.99]	25
[3.0, 3.49]	35
[3.5, 3.99]	40
[4.0, 4.49]	45
[4.5, 5.00]	50

Table 5. RATE

Example 4. *Enumerate all projects for which bonus is 45 (i.e., for which the score lies in the interval for which bonus is 45).* In SQL, this query takes the following form:

```
SELECT PROJECT.P#
    FROM PROJECT
    WHERE PROJECT.Score ⊆
        (SELECT RATE.Rate
        FROM RATE
        WHERE RATE.Bonus = 45);
```

and leads to Table 6:

P#
P2
P3

Table 6.

Example 5. *Find all the projects of status 3 for which the Bonus is 40 or higher.* To be more precise: Find the P#s for those projects whose Status is 3 and whose score is contained in the range of rates for Bonus ≥ 40.

The corresponding SQL query is:

```
SELECT PROJECT.P#
    FROM PROJECT, RATE
    WHERE PROJECT.Score ⊆ RATE.Rate
        AND PROJECT.Status = 3
        AND RATE.Bonus ≥ 40;
```

Since this query uses both tables, SQL will use a θ-join between the tables, joining the records for which PROJECT.Score $\subseteq$ RATE.Rate, and then selecting records (rows) for which Status = 3 and Bonus ≥ 40. The result is given in Table 7.

P#
P1
P12

Table 7.

In the last two examples, we will use another pair of tables: PRICE_LIST (Table 8) with attributes Part#, Part_name and Price (where Price is an interval of possible prices) and TARIFF_LIST (Table 9), where the Tariff is determined by a range of prices.

P#	P_name	Price
1001	nail	[0.04, 0.07]
1002	bolt	[0.23, 0.24]
1003	screw	[0.21, 0.30]
1004	nut	[0.03, 0.10]
1005	washer	[0.27, 0.31]
1006	cam	[0.85, 1.10]
1007	gear	[2.13, 2.45]
1008	axle	[1.75, 1.95]
1009	wheel	[2.56, 2.85]

Table 8. PRICE_LIST

Price	Tariff
[0.00, 0.15]	A
[0.16, 0.25]	B
[0.26, 0.50]	C
[0.51, 1.00]	D
[1.01, 1.10]	E
[1.11, 1.50]	F
[1.51, 2.00]	G
[2.01, 3.00]	H
[3.00, 9.99]	I

Table 9. TARIFF_LIST

Example 6. *Which parts may be subject to Tariff A?* In other words, for which parts, the parts may belong to the price range for Tariff A?

In interval terms, the corresponding relation "may belong" between the two intervals $\mathbf{x} = [\underline{x}, \overline{x}]$ and $\mathbf{y} = [\underline{y}, \overline{y}]$ can be described as $\Diamond(\mathbf{x} \in \mathbf{y})$, which is equivalent to $\mathbf{x} \cap \mathbf{y} \neq \emptyset$, or, to $\max(\underline{x}, \underline{y}) \leq \min(\overline{x}, \overline{y})$.

The corresponding SQL query is:

```
SELECT Price_list.P#, Price_list.P_name
    FROM Price_list, Tariff_List
    WHERE ◊(Price_List.Price ∈ Tariff_List.Price)
        AND Tariff_List.Tariff = 'A';
```

The resulting table is Table 10:

P#	P_name
1001	nail
1004	nut

Table 10.

Example 7. *If the price changes for next year are between −2% and 3%, which parts are guaranteed to have tariff groups less than 'G' next year?*

In interval terms: which P#, P-name pairs will have the range for their prices next year that is less than or equal to the range of prices for Tariff G?

The corresponding SQL query is:

```
SELECT Price_list.P#, price_list.P_name
    FROM Price_List, Tariff_List
    WHERE
        □(Price_List.Price * (1. + [−.02, .03]) ≤ Tariff_list.Price)
        AND Tariff_List.Tariff = 'G';
```

The resulting table is Table 11:

P#	P_name
1001	nail
1002	bolt
1003	screw
1004	nut
1005	washer
1006	cam
1008	axle

Table 11.

3. Conclusion

For cases where it is natural to think of a data item as having a *range* of values, the data type interval can be naturally used. Using interval arithmetic, it is possible to do calculations and comparisons of interval data items and produce interval results without having to be concerned with the details of how to handle the end-points of the intervals.

References

[1] Chang, T. H. and Sciore, E. *A universal relation data model with semantic abstractions.* IEEE Transactions on Knowledge and Data Engineering **4** (1) (1992), pp. 23–33.

[2] Date, C. J. *A proposal for adding date and time support to SQL.* ACM SIGMOD Record **17** (2) (1988).

[3] Korth, H. F. and Silberschatz, A. *Database system concepts.* McGraw-Hill Co., N.Y., 1986.

[4] Moore, R. E. *Interval analysis.* Prentice Hall, Englewood Cliffs, 1966.

[5] Parasaye, K., Chignell, M., Khoshafian, S., and Wong, H. *Intelligent Databases.* J. Wiley and Sons, N.Y., 1989.

Received: May 14, 1995
Revised version: November 22, 1996

Information and Decision Sciences Department
The University of Texas at El Paso
El Paso, Texas 79968
USA