USING INTERVAL DATA LIBRARY FOR SOLVING INTERVAL OBJECTS CLUSTERING PROBLEMS

Artem I. Miroshnikov, Pavel V. Saraev

Dept. of Automation and Computer Science, Lipetsk State Technical University
e-mails: a.i.miroshnikov@yandex.ru, psaraev@yandex.ru
Russian Federation

Abstract: The article describes the use of the developed interval data library for solving the problem of cast iron grades clustering which characteristics are recorded in the database in an interval form. This dynamic link library allows application programmers to write simple SQL-query using user-defined data type and it will be processed on the database server.

Key words: interval-valued data, user-defined data types, interval queries, reliable calculations, interval database.

1. INTRODUCTION

Modern information systems store and process huge amounts of data on technical and economic processes, statistical information. In this connection, the question arises of maximally efficient use of this data, of extracting hidden knowledge that is potentially useful to its owner (which is often used by large organization) in commercial terms.

The approach to the organization of a custom type allows to use existing database technologies to integrate methods of working with interval-valued data types in existing systems which allows to achieve high performance when executing queries based on the use of simple SQL syntax.

One of the most common databases are reference systems which are in particular an element of the information system on cast iron brands. This system which stores information about all brands of cast iron can be used to solve the problem of clustering or classification of cast iron brands. Clustering and classification analysis can be applied according to various characteristics stored in the database.

When clustering over interval-valued data it is necessary to use an interval comparison in the k-means algorithm and when calculating the metric, the square of

the Euclidean distance it is convenient to have a tool to calculate the square of the difference of interval values.

2. RELATED WORK DISCUSSION

Unfortunately, it is not always possible to determine the exact value of certain parameters due to the nature of the measuring instruments or the very nature of the data [1]. In cases where it is only possible to determine the boundaries of the range within which the exact value lies, use the theory of interval arithmetic and interval analysis [2 – 4].

In [5] the description of the interval-valued data, arithmetic operations, aggregating functions over interval types is given. Problems (in particular the absence of an unambiguous formalized solution in the case of intersection of compared intervals) are described and a method for comparing interval-valued data types using a probability indicator is presented. Issues of efficient storage of interval-valued data and the use of indexes of database management systems [6, 7] to speed up access operations are considered.

3. MATERIALS AND METHODS

Consider the process of implementing application software that uses the interval-valued data library using the Microsoft Visual Studio Express IDE and Microsoft SQL Server DBMS in C# (Figure 1) [8, 9]. First of all in Visual Studio it is necessary to specify a link to the interval-valued data library by adding it in the section Reference → Add reference. This library contains the description of the base class iInterval which constructor contains data integrity checks and implements rounding in accordance with the properties of the interval-valued type. The private fields of the class and their purpose are listed in Table 1. The assignment of values to the closed fields is realized through the mechanism of properties containing the corresponding getters and setters calling the methods for checking the integrity of interval-valued data.

Table 1. Base class field description

Base class field	***Base class field description***
private double down;	Lower interval limit
private double up;	Upper interval limit
private bool isNull;	Value used to implement the INullable interface
public static double p;	Parameter used when implementing the comparison of interval values

It contains overriding of interval arithmetic operations, comparison of interval data, methods that return the upper and lower bounds of interval values, width, average, radius, difference square, product of interval and scalar values. Methods of

special structures that allow aggregate operators to be implemented over a user-defined data type when writing transparent SQL queries are listed in Table 2.

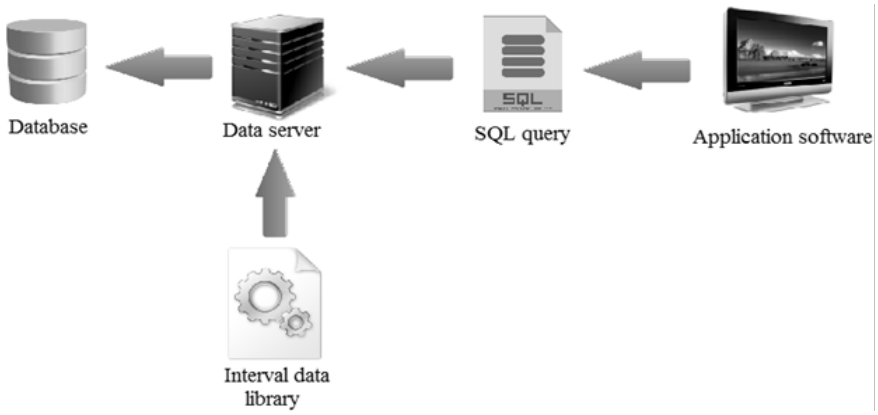

Fig. 1. The class that implements the calculation of the square of the difference of interval-valued data

Separately it is worth noting the description of the class (Figure 2), which allows to calculate the square of the difference between interval-valued data used in the calculation of the Euclidean distance between two interval values. This implementation becomes possible after creating an overload of standard operators (Table 2).

Table 2. Aggregating operators over interval data

Structure header	***Aggregate description***
public struct ICOUNT	Number of values in the column of the table containing interval data
public struct ISUM	The sum of the values in the column of the table containing the interval data
public struct ISUB	Difference of values in the column of the table containing interval data
public struct IMULT	Product of the values in the column of the table containing the interval data

After connecting the link to the interval data library it is necessary to allow SQL Server to execute code from custom assemblies [10] by passing the "[clr enabled], 1" parameter to the stored procedure "sp_configure" and then calling the "reconfigure" instruction. Next register the assembly (CREATE ASSEMBLY) specifying the path to the library of interval-valued data in the "Programming" section of SQL Server. Next it is necessary to run a Transact-SQL script that registers user-stored functions and aggregates implemented in the interval-value data

library. An example of creating a function that calculates the Euclidean distance is shown in Figure 3.

```
public class I_SQ_DIFF
{
    [SqlFunctionAttribute()]
    public static iInterval SQ_DIFF(iInterval obj1, iInterval obj2)
    {
        iInterval s = new iInterval();
        s = (obj1 - obj2) * (obj1 - obj2);
        return s;
    }
}
```

Fig. 2. The class that implements the calculation of the square of the difference of interval-valued data

```
GO
IF EXISTS (SELECT * FROM sys.objects WHERE [name] = 'I_SQ_DIFF')
DROP FUNCTION I_SQ_DIFF;
GO
CREATE FUNCTION dbo.I_SQ_DIFF(@value1 iInterval, @value2 iInterval)
RETURNS iInterval
AS EXTERNAL NAME
INTERVALS.[iInterval.I_SQ_DIFF].[SQ_DIFF]
```

Fig. 3. Custom stored function registration

Further implementation of application software that produces the cast iron grades clustering will use all the possibilities provided by the interval-valued data library. Consider the sequence of actions of the k-average interval algorithm using the Euclidean distance as a metric for clustering.

Step 1. Creating centroids containing random interval-value data. The init_centroids method is called to initialize with random values (Figure 4). The input parameters are a list of objects of type iIron (Figure 5) and the number of centroids that are defined by the user.

```
private void init_centroids(List<iIron> centriods, int number_of_centroids)
```

Fig. 4. The header of the method initializing the initial values of the centroids

Due to the fact that the input parameters of the methods are passed by reference and not by value the type of the return value is implemented as void.

Fig. 5. Cast iron storage format

```
SELECT id, iron_name,
I_SQ_DIFF(@C,  Iron.C)  +
I_SQ_DIFF(@Si, Iron.Si) +
I_SQ_DIFF(@Mn, Iron.Mn) +
I_SQ_DIFF(@Cr, Iron.Cr) +
I_SQ_DIFF(@Cu, Iron.Cu) +
I_SQ_DIFF(@Ni, Iron.Ni) +
I_SQ_DIFF(@S,  Iron.S)  +
I_SQ_DIFF(@P,  Iron.P)  +
I_SQ_DIFF(@Al, Iron.Al) +
I_SQ_DIFF(@Hardness,  Iron.Hardness) +
I_SQ_DIFF(@Strength,  Iron.Strength) +
I_SQ_DIFF(@Extensio1, Iron.Extension) AS distance
FROM Iron
```

Fig. 6. SQL query that returns the square of the Euclidean distance from centroid to each of the objects by 12 parameters

Step 2. Calculation of the minimum distances from each object containing interval-valued data to the centroids. This distance determines the belonging of each object to a specific cluster. Comparison of interval-valued distances is performed in accordance with the method described in [4].

Step 3. Recalculation of interval characteristics of centroids.

Step 4. If a certain number of iterations of the previous two points is performed or the values of the centroids change less than a given constant the current division into clusters is displayed otherwise the process is repeated.

4. EXPERIMENTAL RESULTS

For easy viewing of the initial data on the chemical composition and properties of various grades of cast iron and setting clustering parameters the software "Clustering of cast iron using the K-average interval algorithm" was implemented. The structure of the developed software that solves this problem is presented in Fig.7.

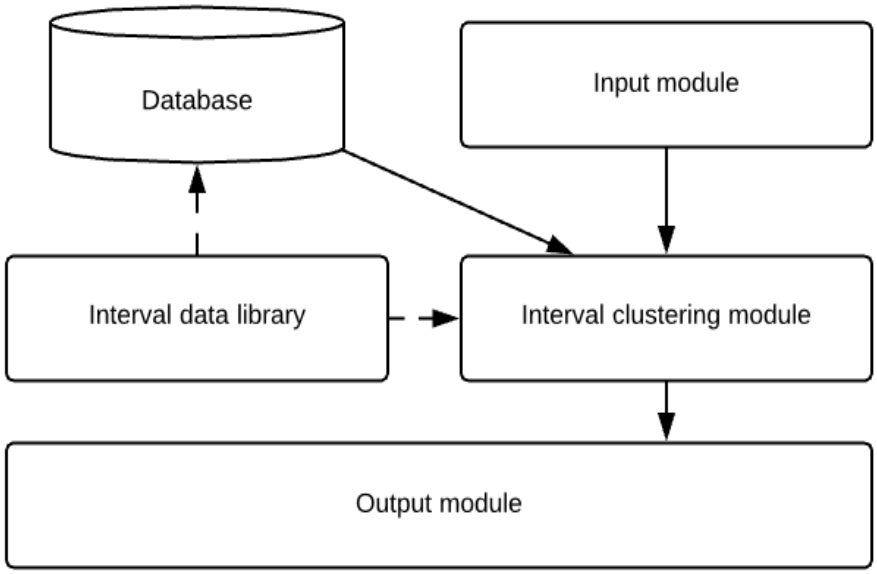

Fig. 7. Interval clustering software structure

The part of application software interface and one result of interval clustering are presented in Figure 8.

The results obtained in the field "Output data" are consistent with the results obtained by other methods. Thus the grades of chromium high alloyed cast irons were assigned to one cluster and austenitic high alloyed cast irons with high heat resistance to another cluster.

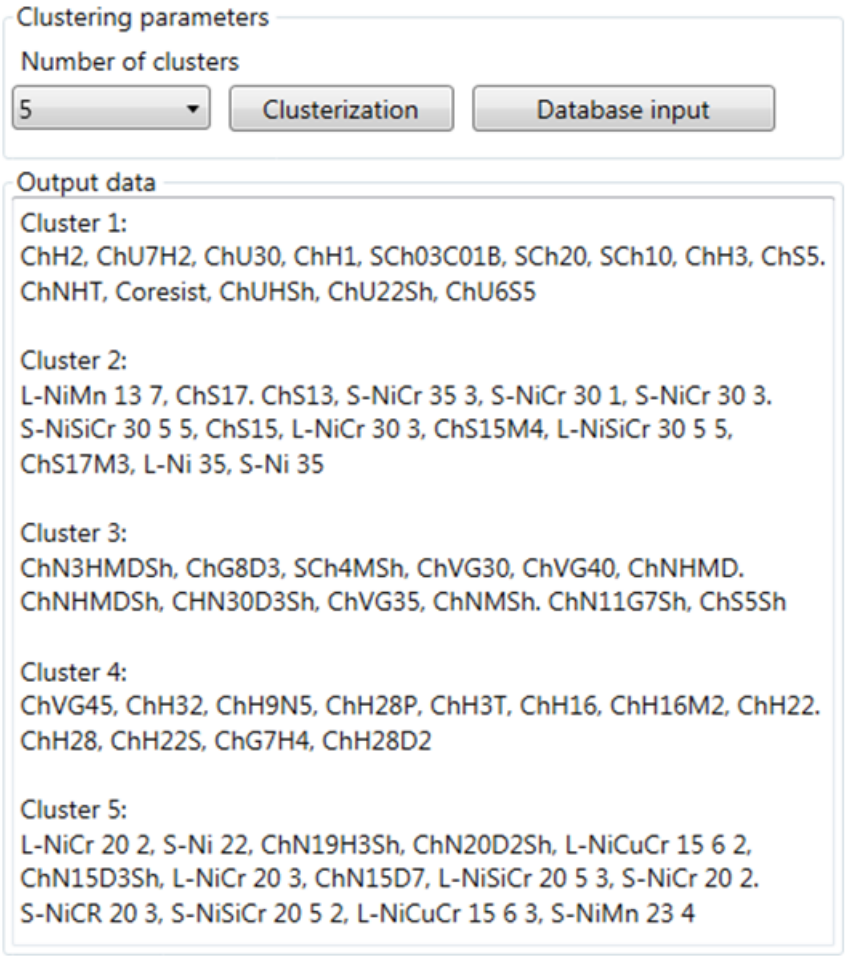

Fig. 8. Interval clustering results

4. CONCLUSION

The development of mathematical and software for analytical processing of information using the library of interval-valued data accelerates the creation of applications, ensuring programmers reliable and efficient processing of user data types allowing them to concentrate on writing the basic logic of the program.

Comparison of the obtained interval values is one of the main tasks when performing interval cluster analysis. Having calculated the indicators for comparison of intervals using the implemented queries and a given probability p=0.7, the grades of cast iron were distributed among the clusters. The probability parameter increasing leads to insignificant changes in the composition of clusters. However, the meaningful nature of clusters remains unchanged.

REFERENCES

[1] Kaufmann, M. Storing and Processing Temporal Data in a Main Memory Column Store. *Proc. of the VLDB Endowment*, 2013, pp. 1444-1449.

[2] Moore, R.E., Kearfott, R.B., Cloud, M.J., Introduction to interval analysis. Philadelphia: SIAM, 2009.

[3] Jaulin, L., Kieffer, M., Didrit, O., Walter, E. Applied interval analysis. London: Springer, 2001.

[4] Alefeld, G., Mayer, G. Interval analysis: theory and applications. *Journal of Computational Applied Mathematics*, vol. 121, 2000, pp. 421-464.

[5] Hansen E., Walster G.W., Global optimization using interval analysis, New York: Marcel Dekker, 2004.

[6] Strate, J., Krueger, T. Expert Performance Indexing for SQL Server 2012, Apress, USA, 2012.

[7] Corlatan, C., Lazar, M., Luca, V., Petricica, O. Query Optimization Techniques in Microsoft SQL Server, *Database Systems*, 2 (vol.5), 2014, pp.33-48.

[8] Troelsen A., Japikse P. Pro C# 7: With .NET and .NET Core, Apress, US, 2017. 1372 pp.

[9] Connolly, T., Begg, C. Database Systems: A Practical Approach to Design, Implementation, and Management, Pearson, 2015.

[10] Petkovic, D. Microsoft SQL Server 2016: A Beginner's Guide, McGraw-Hill Education, 2016.

Information about the authors:

Artem Miroshnikov – Lecturer at the Department of Automation and Computer Science, Lipetsk State Technical University, Russia. Area of Research is interval analysis, data management.

Pavel Saraev – Rector of Lipetsk State Technical University, Russia. Areas of Research are neural modelling, interval analysis.

Manuscript received on 10 July 2019