IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 4, 2018, accepted October 18, 2018, date of publication November 6, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2879829

Fuzzy Join for Flexible Combining Big Data Lakes in Cyber-Physical Systems

BOŻENA MAŁYSIAK-MROZEK, (Member, IEEE), ANNA LIPIŃSKA, AND DARIUSZ MROZEK, (Senior Member, IEEE)
Institute of Informatics, Silesian University of Technology, 44-100 Gliwice, Poland

Corresponding author: Dariusz Mrozek (dariusz.mrozek@polsl.pl)

This work was supported in part by the Microsoft Research within Microsoft Azure for Research Award Grant, in part by the Habilitation Grant of the Rector of the Silesian University of Technology, Gliwice, Poland, under Grant 02/020/RGH18/0148, and in part by the Statutory Research Funds of Institute of Informatics, Silesian University of Technology, under Grant BK/213/RAU2/2018.

ABSTRACT Cyber-physical systems produce large amounts of data that are stored in domain-related data lakes in a variety of formats. By using the big data technologies that enable efficient data processing, the value of the data increases, as these technologies can turn the data into actionable information that influences important decision-making processes. However, a broader view of the operational environment, an investigated phenomena, and challenges related to them can frequently be obtained after combining data from many data sets located in various big data lakes. This requires contact points in both data lakes that must be flexibly joined because in many cases, data sets do not correspond to one another directly. In this paper, we show fuzzy join operation for flexible combining big data lakes. The fuzzy join transforms numerical values of common attributes of joined data sets into fuzzy sets and uses such a representation in the join operation. We propose two variants of the join operation that transforms crisp numerical values of joining attributes into: 1) fuzzy numbers and 2) linguistic terms. The fuzzy join operation is implemented and tested in the declarative U-SQL language that is used for scalable and parallel querying in big data lakes. The ideas presented here are exemplified by a distributed analysis of cardiac disease data on Microsoft Azure cloud. The results of the conducted experiments confirm that the fuzzy join can enrich data sets that are used in making critical decisions and, as a highly scalable cloud-based solution, can be successfully used in processing large volumes of data delivered by cyber-physical systems.

INDEX TERMS Cyber-physical systems, big data, fuzzy logic, querying, cloud computing, biomedical data analysis, declarative languages.

I. INTRODUCTION

Sensor-based and communication-enabled cyber-physical systems (CPSs) continuously generate large volumes of data which necessitates the use of Big Data techniques to process the data and to improve scalability, security, and efficiency of various processes [1]. The term Big Data frequently describes challenges that arise when data sets are so large that the conventional database management systems and data analysis tools are insufficient to process them [2]. The challenges include capture, curation, storage, search, sharing, transfer, analysis and visualization of data [3]. However, even in sensor-rich CPSs covering various areas of our lives we are still able to analyze only a small percentage of the data that is captured and stored. For example, health care providers are usually only interested in 10-20% of the data produced by IoMT (Internet of Medical Things) devices within telemedical CPSs, discarding the rest due to the lack of the need to analyze it or the lack of the idea that such analyzes may be useful. The Big Data technologies promise to turn at least a part of the remaining 80-90% of the unused data into actionable information. These technologies change our thinking about what we can analyze and how we can improve current decision-making processes in many areas of our lives.

Huge storage spaces, which are one of the distinctive features of the era of Big Data that we entered several years ago, fueled by progress in the storage space delivery provided by cloud computing, allow us to collect various types of structured, semi-structured, and unstructured data in data lakes created within various types of CPSs [4]–[8]. When having various data sets in a CPS-related data lake it may

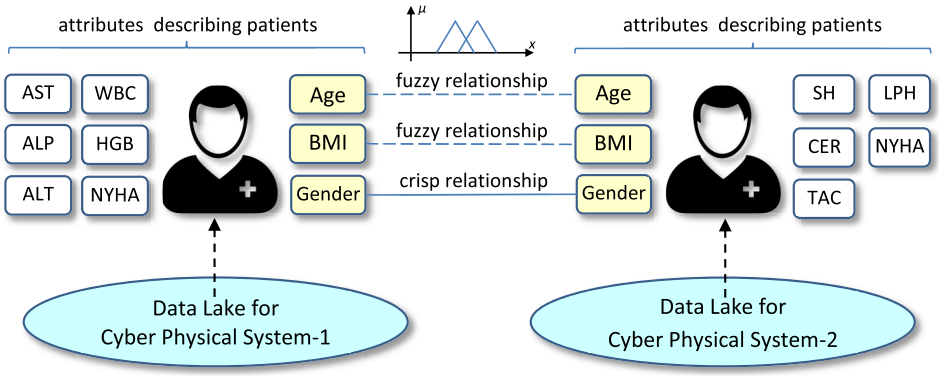

FIGURE 1. Joining two data lakes from various cyber-physical systems on the basis of common attributes (*Age*, *BMI*, and *Gender*).

turn out that some of those data sets share common attributes and, intentionally or not, complement one another [9]. Data sets in various data lakes may describe the same phenomena (e.g., engine vibrations or astronomical observations), the same cases (e.g., exceeding a certain level of temperature in the blast furnace), or the same types of cases (e.g., patients with particular types of diabetes), or profiles (e.g., molecular profiles of patients affected by a tumor), but do not directly correspond with one another (e.g., they do not describe the same patient in a biomedical data lake). They cannot be naturally joined to present a broader view of the investigated phenomena. To clarify this problem better, let us consider the following real-life scenarios.

SCENARIO 1

Two groups of doctors investigate the risk of particular heart-related incidents among patients. They have different intuition on what to investigate, which translates into various features they monitor within medical CPS. Therefore, patients are tested for various blood features. These two groups of doctors produce two different data sets in their data lakes with the information on patients having some common characteristics, like gender, age, BMI (body mass index), NYHA functional classification (New York Heart Association functional classification), and red blood cell morphology characteristics (Fig. 1). However, they also collect other blood characteristics that are driven by their investigations and intuition. For example, the first group collects blood biochemical parameters, like enzyme activities: AST, ALT, GGT; ALP and CK, substrate concentrations: total proteins, albumin, globulins, cholesterol, triglyceride, Ca, P, Mg, glucose, urea and creatinine. Meanwhile, the second group collects antioxidative parameters in the serum, like SH, CER, TAC, LPH. They may want to combine these two data sets to enrich the picture of the disease or the phenomena by joining them on common attributes (Fig. 1). But the precise join is not appropriate for this case, since characteristics collected in both data sets do not apply to the same patients.

SCENARIO 2

A group of doctors investigate cases of heart-related incidents in patients from a particular region of a country. They collect some common characteristics, like gender, age, BMI, NYHA classification, and various serum characteristics in their CPS-related data lakes. They want to supplement their data set with the information on frequent eating habits among people of the same profile living in this region. They need to flexibly combine their data set with the external data sets provided by the National Department of Agriculture or World Health Organization (WHO). This scenario will be covered in Sect. III.

SCENARIO 3

Patients after heart-related incidents or older people are monitored at homes with the use of sensors that are connected to various IoT devices. A part of data is sent through a mobile phone, and another part through a base unit or IoT gateway of a home health monitoring system (Fig. 2). Sensor readings are sent to a common storage place, e.g., a health cloud and its data lake. Sensor readings are sent at a different pace. Then, if an alert is triggered in a particular moment on the basis of exceeded value of one of the monitored parameters (e.g., blood pressure or a fall of a person), which may lead to possibly dangerous situation, we want to take and analyze all of the monitored parameters of the patient (e.g., temperature, respiration, pulse, blood pressure, vital signs) for the specified, *near*-incident moment. However, since these parameters are not saved in the same data sets of the data lake and not in the same moments, we have to join them on the time of the incident, taking into account that the moment in time must be flexible. The flexibility can be implemented by using the idea of *opening umbrella* as it is presented in Fig. 3.

All presented scenarios, in which attributes cannot be joined in a simple manner, motivate the use of fuzzy equi-join operations. The goal of this paper is to show how we can flexibly combine Big Data Lakes with the use of the fuzzy join, extend the picture of the studied phenomena, and have a broader view on its challenges. We present two novel

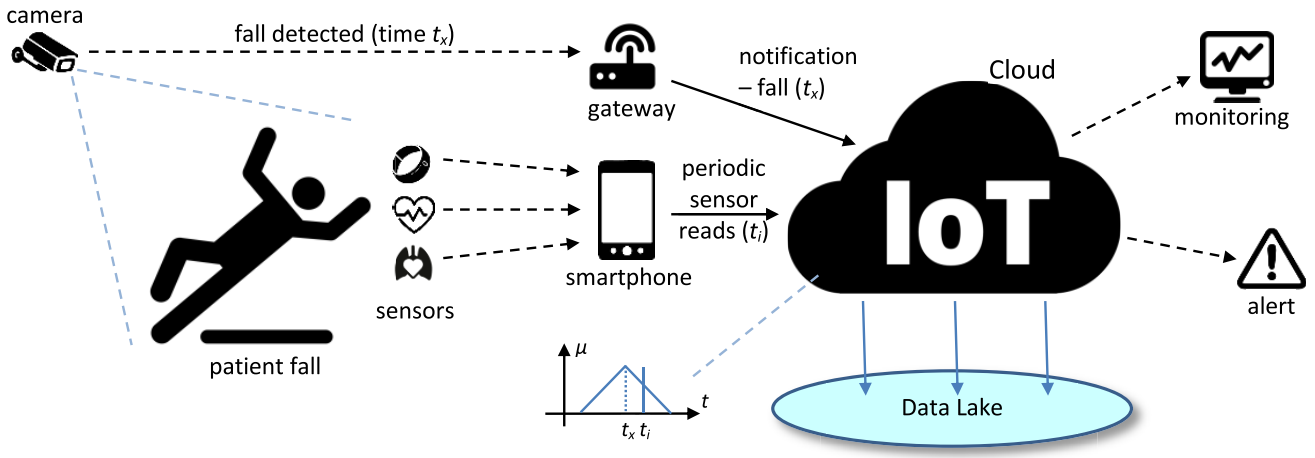

FIGURE 2. Sensor-based telemedical cyber-physical system for monitoring patients or older people at home, sending sensor readings to the Cloud and its data lake. Parameters related to detected dangerous situations, like falls, are sent to the Cloud, where they are joined with periodic readings from body sensors with the use of fuzzy join.

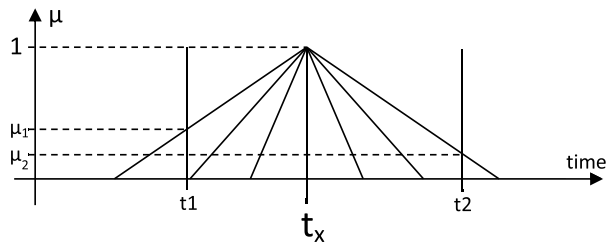

FIGURE 3. The idea of *opening umbrella* for flexible joining sensor readings on the time of an event.

variants for the join operation that rely on the transformation of numerical values of joining attributes to: (1) fuzzy numbers, and (2) linguistic terms. Both approaches are formally defined in Sect. III of the paper. We also show original implementation of the proposed methods in the scalable Azure Data Lake environment (Sect. IV) and define several metrics for evaluation of presented ideas (Sect.V).

II. RELATED WORKS

The term *fuzzy join* is not new in the world scientific literature, but so far it has mainly be applied to joining or matching text, not numerical data. For example, Ananthakrishna *et al.* [10] and Chaudhuri *et al.* [11] proposed an algorithm for eliminating duplicates in dimensional tables in a data warehouse on the basis of textual similarity and fuzzy match operations for online data cleaning. Both works were devoted to fuzzy matching of tuples from relational tables on the basis of textual attributes, and became the foundation for the fuzzy lookup and the fuzzy grouping transformations in the Microsoft SQL Server Integration Services. Recent works in this area, e.g., presented by Afrati *et al.* [12], Deng *et al.* [13], Kimmett *et al.* [14], Yan *et al.* [15], are devoted to performing fuzzy (textual) joins against large collections of data with the use of Big Data technologies, including Hadoop and Spark. Yu *et al.* [16] presented a detailed survey on the used methods. Several works are devoted to join complex data on the basis of vector similarity. Das Sarma *et al.* [17] proposed to use a technique called ClusterJoin for finding pairs of records with similarity score exceeding a certain threshold. Threshold-based similarity is also used by De Francisci Morales and Gionis [18] who proposed the similarity self-join, but authors dedicated their solution for data streams. Similarity join is also used by Kalashnikov [19] for combining multi-dimensional data sets and by Silva *et al.* [20] for joining data in relational databases. MapReduce-based similarity join was proposed by Fries *et al.* [21] and Ma *et al.* [22] for joining high-dimensional vector data, and by Nie *et al.* [23] for entity resolution. Zhao *et al.* [24] proposed a distributed similarity join operator for multi-dimensional arrays dedicated to processing an increasing volume of multi-dimensional data that are largely processed inside distributed array databases. An experimental survey of MapReduce-based similarity joins for Big Data sets is presented in [25]. These works confirm that joining similar data with proximity-based and inequality-based predicates is necessary in many domains to solve existing problems. However, although some of them address the challenges of the Big Data, they do not address problems raised in scenarios presented in Sect. I. In these works the terms *fuzzy join* or *similarity join* are used to reflect soft character of the join operation, which usually bases on the proximity of strings, not numerical values. In fact, none of these works apply fuzzy techniques in the join operation performed.

Meanwhile, the last two decades have brought several applications of fuzzy sets theory for the representation and retrieval of imprecise data in relational databases. Buckles and Petry [26] introduced similarity-based models, where the ordinary equivalence relation between domain values is replaced by similarity relations. Similar works were carried out by Shenoi and Melton [27], [28] who proposed proximity

relations. In possibility-based models [29]–[33] attribute values can be modeled as fuzzy sets on attribute domain. Various extensions to the declarative SQL query language, including SQLf [34], FQUERY [35], Soft-SQL [36], fuzzy Generalised Logical Condition [37], FuzzyQ [38]–[40], and others [41], allowed for flexible data retrieval from relational databases or from fuzzy data warehouses [42], [43]. They are noteworthy, as they deliver various fuzzy techniques for data exploration, like fuzzy filtering, fuzzy inference, generalization with fuzzy linguistic variables, and fuzzy grouping. However, they do not address the *volume* characteristic of Big Data, which are produced by CPSs.

The latest works in this area, including Khorasani *et al.* [44] and the Fuzzy Search Library for Data Lake Analytics (FSL4DLA) proposed by Małysiak-Mrozek *et al.* [5], enable performing scalable fuzzy relational operations in Big Data environments. The approach proposed by Khorasani *et al.* [44] implements these operations as MapReduce procedures (jobs) in Hadoop. It is also the first solution that allows fuzzy theory-based fuzzy join for big numerical data. The FSL4DLA proposed by Małysiak-Mrozek *et al.* [5] implements various information retrieval operations in the Azure Data Lake cloud environment and, in contrast to the MapReduce implementation of Khorasani *et al.*, allows for declarative, scalable querying of big data lakes with the U-SQL language. However, so far it has not allowed for fuzzy joins.

The solution presented in this paper extends capabilities of the Fuzzy Search Library for Data Lake Analytics (FSL4DLA) proposed by Małysiak-Mrozek *et al.* [5] toward performing fuzzy joins on Big Data sets. The range of novelties proposed and presented in the paper covers: (1) enabling the fuzzy join operation for large data sets, (2) providing two variants of the join operation – (i) by representing attribute values as fuzzy numbers, and (ii) through assignment to linguistic value, (3) allowing the fuzzy join operation in declarative queries, which simplifies information retrieval.

III. FUZZY EQUI-JOIN

In relational algebra [45], [46] an equi-join is an inner join that uses an equivalence operation (i.e., attributeA = attributeB) to match rows from different tables. Fuzzy equi-join adds some flexibility while matching rows and does not assume strict equivalence of attribute values, but their similarity.

Let us consider two data lakes *DL1* and *DL2* providing two data sets schematized to rowsets R of the schema $\chi_R = A_1, A_2, \ldots, A_k, R_1, R_2, \ldots, R_m$ and S of the schema $\chi_S = A_1, A_2, \ldots, A_k, S_1, S_2, \ldots, S_n$, where attributes $A_1, A_2, \ldots, A_k$ are common (or related) for both rowsets:

$$\{A_1, A_2, \ldots, A_k\} = \chi_R \cap \chi_S, \tag{1}$$

k is the number of common attributes in rowsets R and S, $R_1, R_2, \ldots, R_m$ are remaining attributes of the rowset R, and $S_1, S_2, \ldots, S_n$ are remaining attributes of the rowset S.

```
Heart disease = Ge Age BMI WBC HGB
                -- --- --- --- ---
                 M  45  24 7.9 6.6
                 F  67  36 5.6 5.2
                 M  25  29 6.0 8.3
                 F  38  26 9.9 9.4
                 M  72  23 5.8  11

Eating habits = Ge Age BMI Sugar Veget
                -- --- --- ----- -----
                 F  42  28   700  1100
                 M  55  26  1000   420
                 M  23  27  1200   670
                 F  61  35  1700   250
```

FIGURE 4. Two sample rowsets from two data lakes sharing common attributes: gender (*Ge*), *Age*, *BMI*.

For example, Fig. 4 shows two rowsets *Heart disease* (abbrev. *HD*) and *Eating habits* (abbrev. *EH*) with common attributes, i.e., `Gender` (`Ge`), `Age`, `BMI`, and additional informative attributes `WBC` and `HGB` for the rowset `HR`, and `Sugar` and `Veget` for the rowset `EH`. The `WBC` (White Blood Cell Count, $10^3/mm^3$) and the `HGB` (Hemoglobin, $mmol/L$) are blood morphology markers taken from patients with a heart-related incident, and `Sugar` and `Veget` show consumption of sugar and vegetables (cal/day) for people in a particular age and gender.

For the rowsets R and S we define the fuzzy equi-join as follows:

$$R \tilde{\bowtie} S = \Pi_{\chi_R \cup \chi_S \cup \chi_\sim}(\tilde{\sigma}^{\lambda}_{R.A_1 \approx S.A_1 \wedge R.A_2 \approx S.A_2 \wedge \ldots \wedge R.A_{k'} \approx S.A_{k'}} (R \times S)), \tag{2}$$

where $\tilde{\bowtie}$ is the symbol used for the fuzzy join operation, $\Pi_{\chi_R \cup \chi_S \cup \chi_\sim}$ is the regular projection operation as defined in the relational algebra, $\chi_R \cup \chi_S \cup \chi_\sim$ is the new schema for the produced rowset containing attributes of both rowsets R and S, and additional attributes produced by the fuzzy operation (represented by schema $\chi_\sim$), $\tilde{\sigma}^{\lambda}_{R.A_1 \approx S.A_1 \wedge R.A_2 \approx S.A_2 \wedge \ldots \wedge R.A_{k'} \approx S.A_{k'}}$ is the fuzzy selection operation as defined in [5], and $R \times S$ is the Cartesian product of the rowset R with the rowset S.

The

$$R.A_1 \approx S.A_1 \ \wedge \ R.A_2 \approx S.A_2 \ \wedge \ \ldots \ \wedge \ R.A_{k'} \approx S.A_{k'} \tag{3}$$

is called the *fuzzy join condition*, and is actually the filtering condition in the fuzzy selection operation $\tilde{\sigma}$. The $\approx$ is a fuzzy match operator for comparing and matching values of joining attributes A_i (for $i = 1 \ldots k'$, where $k' \leq k$), λ is a minimum membership (or similarity) degree (cutoff threshold) for which the whole fuzzy search condition in the fuzzy selection must be satisfied, k' is the number of joining attributes. The prefix *equi-* is added to the term *fuzzy join* to emphasize the fact that values of joining attributes must approximately match to one another (we use the approximate match operator $\approx$), but the fuzzy join can rely on other operators (e.g., fuzzy greater $\gtrsim$ or fuzzy lower $\lesssim$, like in fuzzy selection conditions used in our extensions to Doctrine Query

Language for object-relational mapping [47], [48]). The set of joining attributes is a subset of the set of common attributes:

$$\{A_1, A_2, \ldots, A_{k'}\} \subseteq \chi_R \cap \chi_S, \tag{4}$$

where $k' \leq k$, which means that not all common attributes must participate in the fuzzy equi-join as joining attributes. For example, joining the two rowsets presented in Fig. 4 is possible with the use of one or more of the common attributes (*Ge*, *Age*, *BMI*). The join operation can be performed traditionally, i.e., as a natural join, with the use of all of the three common attributes. However, the returned resulting rowset would be too narrow or empty. The join operation can also be performed as the fuzzy equi-join with the use of the *Age* or *BMI* attributes, or both. This should provide a much broader result set, which can be advantageous in many situations, e.g., it could allow to study various possible dependencies between certain facts, like the dependency between values of blood morphology markers and eating habits.

The fuzzy equi-join can be implemented in various ways. In the following sections, we show two variants of the fuzzy join operation:

1) by representing values of joining attributes as fuzzy numbers,
2) by assigning crisp values of joining attributes to linguistic values defined by fuzzy sets.

A. FUZZY JOIN ON FUZZY NUMBERS

The first implementation of the fuzzy join operation assumes fuzzification of values of the joining attributes and representing them as fuzzy numbers. This process is schematically shown in Fig. 5. Having the function $\phi(A)$ that fuzzifies attributes of the join condition given as follows:

$$\phi(A) : t(A) \longrightarrow \{(t(A), \mu(t(A)))\}, \tag{5}$$

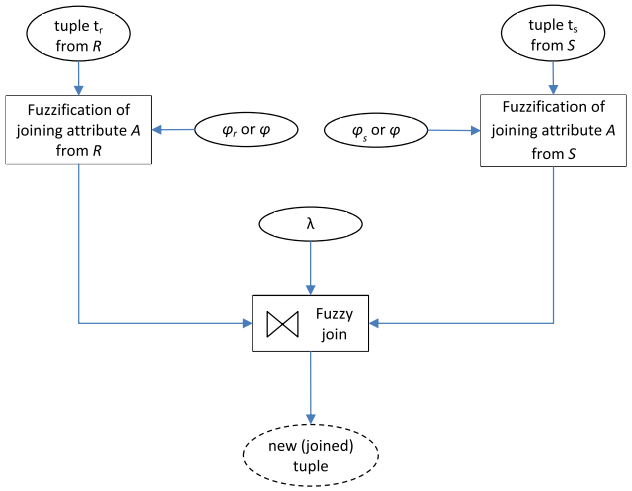

FIGURE 5. Schematic overview of the fuzzy join performed by representing joined values of a single joining attribute (A) as fuzzy numbers.

where $\mu(t(A))$ is the membership function defining a fuzzy set for each value of the attribute A in tuple t, we can perform a fuzzy-equi join with fuzzified values of the joining attributes by comparing them and examining their similarity:

$$R \tilde{\bowtie}_{\wedge} S = \Pi_{\chi_R \cup \chi_S \cup \chi_\sim} (\tilde{\sigma}^{\lambda}_{\phi_{R1}(R.A_1) \approx \phi_{S1}(S.A_1) \wedge \ldots \wedge \phi_{Rk'}(R.A_{k'}) \approx \phi_{Sk'}(S.A_{k'})} (R \times S)), \tag{6}$$

where $\tilde{\bowtie}_{\wedge}$ is the symbol used for the fuzzy join operation performed by representing attribute values as fuzzy numbers.

The fuzzification functions ϕ_{Ri} and ϕ_{Si} ($i = 1, \ldots, k'$, k' is the number of joining attributes) are different for various (pairs of joining) attributes. For example, we can use different functions for the fuzzification of the attribute *BMI* and the attribute *Age*. In formula 6 we also assume that the fuzzification functions ϕ_{Ri} and ϕ_{Si} can be different for each of corresponding attributes from both rowsets R and S. For example, we can use different spreads of the triangular membership function or even different types of membership functions (triangular, trapezoidal, Gaussian) when fuzzifying the attribute *BMI* from the rowset *Heart disease* and the attribute *BMI* from the rowset *Eating habits*. However, in practice, we use the same fuzzification function (in terms of the type and the parameters) for each pair of joining attributes $(R.A_1, S.A_1), (R.A_2, S.A_2), \ldots, (R.A_{k'}, S.A_{k'})$. This leads to the following (simplified) definition of the fuzzy-equi join:

$$R \tilde{\bowtie}_{\wedge} S = \Pi_{\chi_R \cup \chi_S \cup \chi_\sim} (\tilde{\sigma}^{\lambda}_{\phi_1(R.A_1) \approx \phi_1(S.A_1) \wedge \ldots \wedge \phi_{k'}(R.A_{k'}) \approx \phi_{k'}(S.A_{k'})} (R \times S)), \tag{7}$$

where $\{\phi_1, \phi_2, \ldots, \phi_{k'}\}$ is the set of common functions for fuzzifying joining attributes $A_1, A_2, \ldots, A_{k'}$ from both rowsets R and S. The set $\chi_\sim$ of additional attributes produced by the fuzzy equi-join contains the similarity degree for each pair of joining attributes:

$$\chi_\sim = \{\mu_1(R.A_1, S.A_1), \ldots, \mu_{k'}(R.A_{k'}, S.A_{k'})\}, \tag{8}$$

For example, the fuzzy join with the use of the BMI attribute between two rowsets *Heart disease* (abbrev. *HD*) and *Eating habits* (abbrev. *EH*) for the $\lambda = 0.5$ can be formally written as follows:

$$HD \tilde{\bowtie}_{\wedge} EH = \Pi_{HD \cup EH \cup \chi_\sim} (\tilde{\sigma}^{0.5}_{\phi(HD.BMI) \approx \phi(EH.BMI)} (HD \times EH)). \tag{9}$$

The fuzzification of the joining attributes and examination of their similarity is presented in Fig. 6. The results of performing the fuzzy join operation on the basis of the *BMI* attribute and, additionally, the *Gender* attribute from two sample rowsets from Fig. 4 for the given $\lambda = 0.5$ cutoff threshold are shown in Fig. 7. The example displays selected eating habits and blood morphology parameters for people of the same gender that have similar BMI.

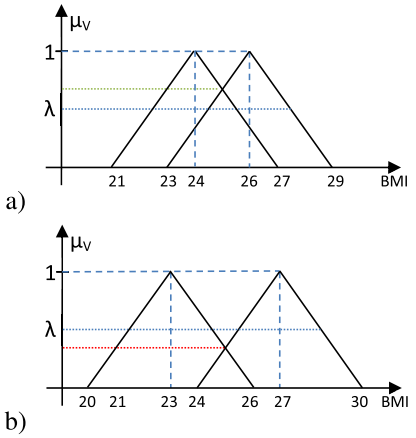

FIGURE 6. Matching fuzzified values of the *BMI* attributes in the fuzzy join condition: a) two BMI values satisfying the join condition, b) two BMI values that do not satisfy the join condition for the given λ cutoff threshold.

Produced result:

BmiHD	BmiEH	MDegBmi	GeHD	GeEH	AgeHD	AgeEH	Sugar	Veget	WBC	HGB
24	27	0.500	M	M	45	23	1200	670	7.9	6.6
24	26	0.667	M	M	45	55	1000	420	7.9	6.6
36	35	0.833	F	F	67	61	1700	250	5.6	5.2
29	27	0.667	M	M	25	23	1200	670	6.0	8.3
29	26	0.500	M	M	25	55	1000	420	6.0	8.3
26	28	0.667	F	F	38	42	700	1100	9.9	9.4
23	26	0.500	M	M	72	55	1000	420	5.8	11.0

FIGURE 7. Results of the fuzzy join operation performed by representing attribute values as fuzzy numbers on the basis of the *BMI* and the *Gender* attributes from two sample rowsets for the given λ cutoff threshold.

Formally, the operation presented in Fig. 7 can be written as follows:

$$HD \,\tilde{\bowtie}_{\wedge}\, EH = \Pi_{HD \cup EH \cup \chi_{\sim}} (\tilde{\sigma}^{0.5}_{\phi(HD.BMI) \approx \phi(EH.BMI) \wedge HD.Ge = EH.Ge} (HD \times EH)). \quad (10)$$

The selection σ consists of two filtering conditions: (1) the fuzzy one on the *BMI* attributes from both rowsets (values of the BMI from both rowsets should be similar), and (2) the crisp one on the *Gender* attributes from both rowsets (values of the *Gender* attributes should be equal). Such a report allows to check if there is any correspondence between eating habits and morphology parameters of people with heart-related incidents.

The produced report shows both values of the BMI (from *HD* and *EH* rowsets), the similarity degree between them (`MDegBmi`), gender and age from both rowsets (`GeHD`, `GeEH`, `AgeHD`, `AgeEH`, respectively), eating habits (`Sugar` and `Veget`), and blood morphology markers (`WBC`, `HGB`).

B. FUZZY EQUI-JOIN WITH LINGUISTIC VALUES

Fuzzy join can be also performed by assigning crisp values of the joining attributes from both rowsets to most appropriate linguistic values of predefined linguistic variables. This process is schematically illustrated in Fig. 8. Then, a regular (natural) join can be performed on the linguistic counterparts

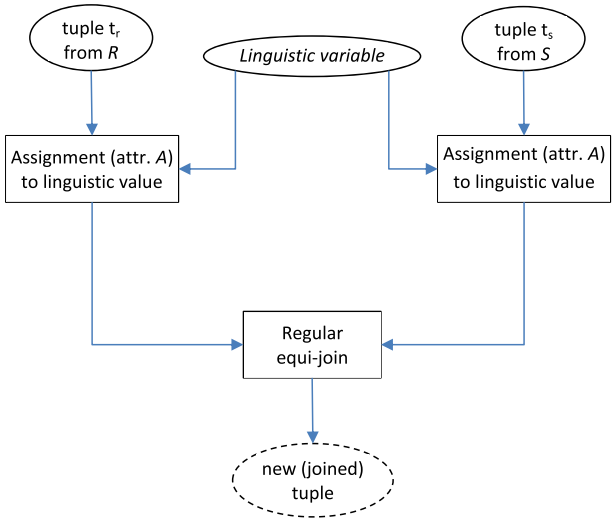

FIGURE 8. Schematic overview of the fuzzy join performed through assignment of joined values of the common joining attribute (A) to appropriate linguistic value.

of the joining attributes. For the same two rowsets R and S from a data lake (or two independent data lakes), we can define the fuzzy equi-join with linguistic values as follows:

$$R \,\tilde{\bowtie}_{L}\, S = \tilde{\Pi}_{\chi_R \cup \chi_S \cup \chi_{\sim}} (\sigma_{\tilde{T}_{L1}(R.A_1) = \tilde{T}_{L1}(S.A_1) \wedge \ldots \wedge \tilde{T}_{Lk'}(R.A_{k'}) = \tilde{T}_{Lk'}(S.A_{k'})} (R \times S)), \quad (11)$$

where $\tilde{\bowtie}_L$ is the symbol used for the fuzzy join operation performed through assignment to linguistic values, $\tilde{\Pi}_{\chi_R \cup \chi_S \cup \chi_{\sim}}$ is the extended projection with a fuzzy transformation [5], $\sigma_{\tilde{T}_{L1}(R.A_1) = \tilde{T}_{L1}(S.A_1) \wedge \ldots \wedge \tilde{T}_{Lk'}(R.A_{k'}) = \tilde{T}_{Lk'}(S.A_{k'})}$ is the standard selection operation [45], [46] with the filtering (selection) condition $\tilde{T}_{L1}(R.A_1) = \tilde{T}_{L1}(S.A_1) \wedge \ldots \wedge \tilde{T}_{Lk'}(R.A_{k'}) = \tilde{T}_{Lk'}(S.A_{k'})$, which is actually the join condition for the fuzzy join operation.

The extended projection with a fuzzy transformation performs assignment of (crisp) values of the rowset attribute A_i to appropriate linguistic values. The fuzzy transformation $\tilde{T}(A_i)$ used in the extended projection consists of two component transformations:

$$\tilde{T}(A_i) = \left\langle \tilde{T}_L(A_i), \tilde{T}_{\mu}(A_i) \right\rangle, \quad (12)$$

The $\tilde{T}_L(A_i)$ component transformation (used also in the filtering/join condition in the selection operation, see definition of the fuzzy join in Eq.11) assigns and returns a linguistic value l of the defined linguistic variable $L = \{l_1, l_2, \ldots, l_{|L|}\}$, $|L| \in \mathbb{N}_+$, for the value of an attribute A_i from each tuple t of the rowset R and S, in such a way that:

$$\forall_{t \in R,S} \;\; \exists_{l \in L} \quad \tilde{T}_L(t(A_i)) = l \;\wedge \mu_l(t(A_i)) = \max\{\mu_{l_1}(t(A_i)), \mu_{l_2}(t(A_i)), \ldots, \mu_{l_{|L|}}(t(A_i))\}. \quad (13)$$

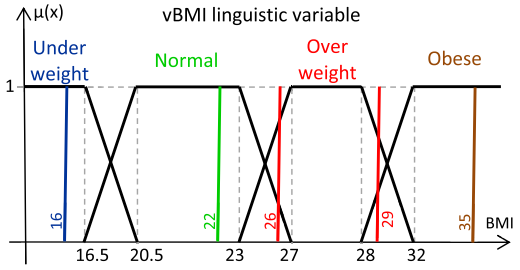

FIGURE 9. Linguistic variable *vBMI* and its linguistic values (*Underweight*, *Normal*, *Overweight*, *Obese*), and the assignment of sample values of the *BMI* attribute to appropriate linguistic value. Defined on the basis of simplified BMI classification of World Health Organization (WHO) with boundaries corresponding to $\mu(BMI) = 0.5$.

The linguistic variable L must be defined by an expert who knows the domain of the data or on the basis of existing standards. Additionally, the $\tilde{T}_\mu(A_i)$ component transformation calculates the membership degree of the value $t(A_i)$ of an attribute A_i for each tuple of the rowsets R and S to the linguistic value l the tuple was assigned to:

$$\forall_{t \in R,S} \;\; \tilde{T}_\mu(t(A_i)) = \mu_l(t(A_i)). \tag{14}$$

The schema of the rowset produced as a result of the fuzzy join $\chi_R \cup \chi_S \cup \chi_\sim$ consists of all attributes of the rowsets R and S, and the set $\chi_\sim$ of additional attributes produced by the fuzzy transformation $\tilde{T}(A_i)$. For each pair of joining attributes $(R.A_i, S.A_i)$, the set $\chi_\sim$ of additional attributes produced by the fuzzy join contains linguistic values for both attributes ($R.A_i$ and $S.A_i$), and similarity degrees μ_i showing the compatibility between the attribute values and the linguistic values:

$$\begin{aligned}\chi_\sim = \{&l(R.A_1), \mu_1(R.A_1, l), l(S.A_1), \mu_1(S.A_1, l),\\ &l(R.A_2), \mu_2(R.A_2, l), l(S.A_2), \mu_2(S.A_2, l), \ldots,\\ &l(R.A_{k'}), \mu_{k'}(R.A_{k'}, l), l(S.A_{k'}), \mu_{k'}(S.A_{k'}, l)\},\end{aligned} \tag{15}$$

The compatibility degree is calculated as it is presented in Fig. 9. Since values of each of the joining attributes $(R.A_i, S.A_i)$ are assigned to linguistic terms $l(R.A_i)$ and $l(S.A_i)$, this produces two compatibility degrees $\mu_i(R.A_i, l)$ and $\mu_i(S.A_i, l)$ per each pair of joining attributes.

Fig. 10 shows results of the fuzzy equi-join through assignment to linguistic values for the sample data from Fig. 4. The linguistic variable *vBMI* and its linguistic values used in the assignment step (fuzzy transformation) are presented in Fig. 9. Likewise in the example presented in the previous section, the join is performed on the `BMI` attribute, and additionally, on the `Gender` attribute between two rowsets *Heart disease* (abbrev. *HD*) and *Eating habits* (abbrev. *EH*). This operation can be formally written as follows:

$$\begin{aligned}HD \tilde{\bowtie}_L EH = \tilde{\Pi}_{HD \cup EH \cup \chi_\sim} \\ (\sigma_{\tilde{T}_L(HD.BMI) = \tilde{T}_L(EH.BMI) \wedge HD.Ge = EH.Ge} \\ (HD \times EH)).\end{aligned} \tag{16}$$

Likewise in the example presented in the previous section, the selection σ consists of two filtering/join conditions. However, in contrast to the previous example, in this case both of the filtering/join conditions are crisp: (1) the natural join is performed on the `BMI` attributes from both rowsets assigned to linguistic values (named fuzzy sets), and (2) the natural join is performed on the `Gender` attributes from both rowsets.

The assignment of the values of the joining attributes (`BMI`, both from the rowset *HD* and the rowset *EH*) and calculation of the similarity degree is presented in Fig. 9. Results of the fuzzy join on the basis of the `BMI` attribute and, additionally, the `Gender` attribute from two sample rowsets from Fig. 4 are shown in Fig. 10. Again, the example shows selected eating habits and blood morphology parameters for people of the same gender that have similar BMI. However, the similarity of BMI is now calculated through assignment to a linguistic value.

The produced report shows both values of the BMI (from *HD* and *EH* rowsets, `BmiHD` and `BmiEH`), the linguistic values they were assigned to (`bmiLingHD` and `bmiLingEH`), the similarity degrees between the value of the BMI and the linguistic value the BMI was assigned to (`MDegBmiHD` and `MDegBmiEH`), gender and age from both rowsets (`GeHD`, `GeEH`, `AgeHD`, `AgeEH`, respectively), eating habits (`Sugar` and `Veget`), and blood morphology markers (`WBC`, `HGB`). The linguistic values (`bmiLingHD` and `bmiLingEH`) in each row (tuple) are equal, which allows to verify that the join was performed correctly. However, real values of the BMI in both rowsets (`BmiHD` and `BmiEH`) are different, which confirms that both rowsets were joined with the fuzzy join.

IV. IMPLEMENTATION

Both techniques for performing fuzzy join presented in Sect. III were implemented in the Azure Data Lake environment. We used two main components of the Azure Data Lake environment: (1) Data Lake Store (DLS) for petabyte scale, unlimited storage for data in various formats,

Produced result:

BmiHD	bmiLingHD	MDgBmiHD	BmiEH	bmiLingEH	MDgBmiEH	GeHD	GeEH	AgeHD	AgeEH	Sugar	Veget	WBC	HGB
36	Obese	1	35	Obese	1	F	F	67	61	1700	250	5.6	5.2
29	Overweight	0.75	27	Overweight	1	M	M	25	23	1200	670	6.0	8.3
29	Overweight	0.75	26	Overweight	0.75	M	M	25	55	1000	420	6.0	8.3
26	Overweight	0.75	28	Overweight	1	F	F	38	42	700	1100	9.9	9.4

FIGURE 10. Results of the fuzzy join operation through assignment of *BMI* attribute values to linguistic terms and by joining the *Gender* attributes from two sample rowsets.

and (2) Data Lake Analytics (DLA) for the efficient and scalable analysis of data. We implemented both fuzzy equi-join techniques by extending the spectrum of methods available in the Fuzzy Search Library for Data Lake Analytics [5]. The Fuzzy Search Library for Data Lake Analytics (FSL4DLA) allows fuzzy searching, filtering, transformation, and grouping data in the Data Lake Analytics environment. It uses the U-SQL declarative queries to perform various operations on large volumes of data in a distributed execution environment of the Microsoft Azure cloud. We extended the FSL4DLA with a set of methods that help in calculations related to performed fuzzy join operations. We also exposed two new specific predicates and one function that can be used in the U-SQL scripts when performing fuzzy joins:

- `bool Udfs.AreSimilar(field1, spread1, field2, spread2, lambda)`, from the `Udfs` module of the FSL4DLA, which is used to examine whether two numeric values from `field1` and `field2` fuzzified with the use of `spread1` and `spread2` (two fuzzy numbers) match each other with the minimum similarity (membership or compatibility) degree `lambda`;
- `double Udfs.SDegree(field1, spread1, field2, spread2)`, from the `Udfs` module of the FSL4DLA, which returns the similarity degree between two numeric values from `field1` and `field2`, fuzzified with the use of `spread1` and `spread2` (two fuzzy numbers);
- `bool Udl.AreSimilar(field1, field2, lingvar)`, from the `Udl` module of the FSL4DLA, which is used to examine whether two numeric values from `field1` and `field2` match each other after assigning them to a linguistic value from the linguistic variable `lingvar`.

The `field1` and the `field2` are joining fields (joining attributes in the fuzzy equi-join) from two joined rowsets. Predicates `AreSimilar` were developed in two different modules of the FSL4DLA library, i.e., the `Udfs` module with methods implementing operations on fuzzy numbers and the `Udl` module with methods for operating on linguistic variables and linguistic values. Both predicates have similar function (hence the same name) – examine the compatibility of two attribute values – return the value of *true* or *false*, and are used in the `WHERE` clause of the U-SQL query statements.

Sample U-SQL code for the fuzzy equi-join operation on the basis of the `BMI` attribute (where values are represented as fuzzy numbers) and the `Gender` attribute from two sample rowsets, for the given λ cutoff threshold, is presented in Listing 1. The script (part QP1) extracts data from the source files declared in lines 8-9, and produces two rowsets `@hd` and `@eh` (lines 10-20). The select statement located in QP2 (lines 22-33) performs fuzzy equi-join between these rowsets with the use of the `BMI` attribute. Values of the `BMI` attribute from both rowsets are fuzzified and represented by triangular membership functions (fuzzy sets) with the spread equal to 3 (line 32). The `bool Udfs.AreSimilar` predicate is used to verify if values of the `BMI` attributes match to one another. Fuzzified values of the BMI must match to one another with the similarity degree $\lambda \geq 0.5$. The SELECT clause in lines 24-29 displays selected attributes from both rowsets, and the similarity degree between fuzzified `BMI` attributes by invoking the `Udfs.SDegree` function (line 25). The rowset produced as a result of the fuzzy join is saved to the Data Lake Store in the text format in the OUTPUT statement in QP3 (lines 34-37).

```
1  REFERENCE ASSEMBLY HeartDisease.HeartDiseaseData;
2  REFERENCE ASSEMBLY HeartDisease.FuzzySearchLib;
3  USING Udfs = FuzzySearchLib.Udfs;
4  USING Variables = HeartDiseaseData.Variables;
5  USING Udl = FuzzySearchLib.Udl;
6
7  // QP1: Extract data
8  DECLARE @dataPathHD string = "/Dataset/hd.csv";
9  DECLARE @dataPathEH string = "/Dataset/eh.csv";
10 @hd =
11     EXTRACT Gender string, Age int, Bmi float,
12       Wbc float, Hgb float
13     FROM @dataPathHD
14     USING Extractors.Csv();
15
16 @eh =
17     EXTRACT Gender string, Age int, Bmi float,
18       Sugar float, Veget float
19     FROM @dataPathEH
20     USING Extractors.Csv();
21
22 // QP2: Perform fuzzy join
23 @fjoin =
24   SELECT hd.Bmi AS BmiHD, eh.Bmi AS BmiEH,
25       Udfs.SDegree(hd.Bmi, 3, eh.Bmi, 3)
26           AS MDegBmi,
27       hd.Gender AS GeHD, eh.Gender AS GeEH,
28       hd.Age AS AgeHD, eh.Age AS AgeEH,
29       Sugar, Veget, Wbc, Hgb
30   FROM @hd AS hd JOIN @eh AS eh
31       ON hd.Gender==eh.Gender
32   WHERE Udfs.AreSimilar(hd.Bmi,3, eh.Bmi,3, 0.5);
33
34 // QP3: Save data to Data Lake Store
35 OUTPUT @fjoin
36 TO "/output/fuzzyJoinTest.txt"
37 USING Outputters.Text();
```

Listing 1. Sample U-SQL code for the fuzzy equi-join operation on the *BMI* and the *Gender* attributes from two sample rowsets, for the given λ cutoff threshold. Values of the *BMI* attributes are represented as fuzzy numbers.

The invocation of the `bool Udfs.AreSimilar` predicate in line 32 can be also replaced by the invocation of the `Udfs.SDegree` function in the same place, as it is presented in Listing 2. In both cases, the U-SQL script produces results that were presented in Fig. 7.

Listing 3 shows sample U-SQL code for the fuzzy equi-join operation performed through assignment of the `BMI` attribute values to linguistic terms and by joining the `Gender` attributes from two sample rowsets. Similarly to the previous case, the script (part QP1) extracts data from the source files declared in lines 8-9, and produces two rowsets `@hd` and `@eh` (lines 10-20). Then, in lines 22-23 (QP2), it declares and defines the `vBMI` linguistic variable and its linguistic values (in the U-SQL code represented by the `@bmiLV` string variable). Trapezoidal membership functions (encoded by `T`)

```
1  ...
2  // QP2: Perform fuzzy join
3  @fjoin =
4    SELECT hd.Bmi AS BmiHD, eh.Bmi AS BmiEH, ...
5      ...
6    WHERE Udfs.SDegree(hd.Bmi,3,eh.Bmi,3) >= 0.5;
7  ...
```

Listing 2. Sample U-SQL code for the fuzzy join operation through attribute fuzzification on the basis of the *BMI* and the *Gender* attributes from two sample rowsets, for the given λ cutoff threshold with the invocation of the *SDegree* function in the joining condition. Values of the *BMI* attributes are represented as fuzzy numbers.

```
1  REFERENCE ASSEMBLY HeartDisease.HeartDiseaseData;
2  REFERENCE ASSEMBLY HeartDisease.FuzzySearchLib;
3  USING Udfs = FuzzySearchLib.Udfs;
4  USING Variables = HeartDiseaseData.Variables;
5  USING Udl = FuzzySearchLib.Udl;
6
7  // QP1: Extract data
8  DECLARE @dataPathHD string = "/Dataset/hd.csv";
9  DECLARE @dataPathEH string = "/Dataset/eh.csv";
10 @hd =
11     EXTRACT Gender string, Age int, Bmi float,
12       Wbc float, Hgb float
13     FROM @dataPathHD
14     USING Extractors.Csv();
15
16 @eh =
17     EXTRACT Gender string, Age int, Bmi float,
18       Sugar float, Veget float
19     FROM @dataPathEH
20     USING Extractors.Csv();
21
22 // QP2: Define linguistic variable
23 DECLARE @bmiLV string = "vBMI:Underweight,T
       ,0,0,16.5,20.5/Normal,T,16.5,20.5,23,27/
       Overweight,T,23,27,28,32/Obese,T,28,32,100,100
       ";
24
25 // QP3: Perform fuzzy join
26 @fjoin =
27   SELECT hd.Bmi AS BmiHD,
28       Udl.LingVal(@bmiLV, hd.Bmi) AS bmiLingHD,
29       Udl.MDegree(@bmiLV, hd.Bmi) AS MDgBmiHD,
30       eh.Bmi AS BmiEH,
31       Udl.LingVal(@bmiLV, eh.Bmi) AS bmiLingEH,
32       Udl.MDegree(@bmiLV, eh.Bmi) AS MDgBmiEH,
33       hd.Gender AS GeHD, eh.Gender AS GeEH,
34       hd.Age AS AgeHD, eh.Age AS AgeEH,
35       Sugar, Veget, Wbc, Hgb
36     FROM @hd AS hd JOIN @eh AS eh
37         ON hd.Gender==eh.Gender
38     WHERE Udl.AreSimilar(hd.Bmi, eh.Bmi, @bmiLV);
39
40 // QP4: Save data to Data Lake Store
41 OUTPUT @fjoin
42 TO "/output/fuzzyJoinLing.txt"
43 USING Outputters.Text();
```

Listing 3. Sample U-SQL code for the fuzzy equi-join operation through assignment of the *BMI* attribute values to linguistic terms and by joining the *Gender* attributes from two sample rowsets.

are used to define each linguistic value (`Underweight`, `Normal`, `Overweight`, `Obese`). The definition is consistent with the one shown in Fig. 9. The SELECT statement located in QP3 (lines 25-38) performs fuzzy equi-join between these rowsets with the use of the `BMI` attribute. However, for each row of both rowsets, values of the `BMI` attribute are first assigned to corresponding linguistic value. This is done in two places – in the SELECT clause and in the WHERE clause. Invocation of the `Udl.AreSimilar` predicate in the WHERE clause (line 38) transforms the crisp, numerical value of the `BMI` attributes to linguistic values defined in the `@bmiLV` (`vBMI`) variable, and checks if the values match to one another. Two invocations of the `Udl.LingVal` function in lines 28 and 31 transform values of the BMI from both rowsets and display them (return to the produced rowset `@fjoin`, line 26). Additionally, the `Udl.MDegree` is invoked in lines 29 and 32 in order to return the membership (compatibility) degree between the crisp, numerical value of the BMI and the linguistic value it was assigned to. Again, the rowset produced as a result of the fuzzy join is saved to the Data Lake Store in the text format in the OUTPUT statement in part QP4 (lines 40-43).

The invocation of the `bool Udl.AreSimilar` predicate in line 38 can be also replaced by the invocation of the `Udl.LingVal` function in the same place, as it is presented in Listing 4. In both cases, the U-SQL script produces results that were presented in Fig. 10.

```
1  ...
2  // QP3: Perform fuzzy join
3  @fjoin =
4    SELECT hd.Bmi AS BmiHD, ...
5      ...
6    WHERE Udl.LingVal(@bmiLV,hd.Bmi)
7            == Udl.LingVal(@bmiLV,eh.Bmi);
8  ...
```

Listing 4. Sample U-SQL code for the fuzzy join operation through assignment of the *BMI* attribute values to linguistic terms and by joining the *Gender* attributes from two sample rowsets, with the invocation of the *Udl.LingVal* function in the joining condition.

V. EXPERIMENTAL RESULTS

Both of the implemented techniques of fuzzy join were tested in the Azure Data Lake environment. During the tests we focused on the functionality, the effectiveness, and the performance of the presented methods. Tests were performed on two data lakes with medical data related to cardiomyopathy obtained from Medical University of Silesia in Zabrze and with eating habits, for which the cross product gives almost 3 billion rows. We narrowed the first data set to the one that contained cases that were interesting from the medical point of view (dilated cardiomyopathy) and were further investigated, producing the final rowset with 0.6 billion rows. In this section, we show that by using the fuzzy join we are able to enrich the outcome of the standard join operation and add similar cases on the basis of flexible joining conditions. During our experiments we used several measures to assess the effectiveness of fuzzy querying. They allow to express how much the returned result is enriched by using the fuzzy join and assess the quality of the enrichment.

Selectivity of query is a measure that represents the percentage of rows for which the query (with its filtering conditions) is expected to return the *true* value. The selectivity is a decimal value between 0 and 1 expressed as a ratio between

the cardinality of the output rowset R_O and the cardinality of the processed input rowset R_I:

$$Selectivity = \frac{card(R_O)}{card(R_I)}, \tag{17}$$

where *card* returns the number of rows for the given rowset. Since, we perform the join operation on two rowsets, we will investigate the selectivity of the join condition over the Cartesian product of the two rowsets R and S, which constitutes a possible input rowset for the query:

$$R_I = R \times S, \tag{18}$$

therefore:

$$Selectivity = \frac{card(R_O)}{card(R \times S)}. \tag{19}$$

Relative cardinality, as defined in [49], is expressed as follows:

$$RelCard(R_O^F) = \frac{\sum\limits_{t \in R_O^F} \mu_{R_O^F}(t)}{\sum\limits_{t \in R \times S} \mu_{R_I}(t)}. \tag{20}$$

where R_O^F is the output rowset of the query with the fuzzy join operation, R_I is the input rowset as defined in Eq. 18, and t is a tuple.

For the fuzzy equi-join performed by representing attribute values as fuzzy numbers the $\mu_{R_O^F}$ is calculated as follows:

$$\mu_{R_O^F}(t) = \min\{\mu_1(R.A_1, S.A_1), \ldots, \mu_{k'}(R.A_{k'}, S.A_{k'})\}, \tag{21}$$

where $\mu_1, \mu_2, \ldots, \mu_{k'}$ are similarity degrees calculated for each pair of joining attributes as defined in Eq. 8.

For the fuzzy equi-join performed through assignment to linguistic values the $\mu_{R_O^F}$ is calculated as follows:

$$\mu_{R_O^F}(t) = \min\{\mu_{L1}(R.A_1, S.A_1), \ldots, \mu_{Lk'}(R.A_{k'}, S.A_{k'})\}, \tag{22}$$

where $\mu_{L1}, \mu_{L2}, \ldots, \mu_{Lk'}$ are, calculated for each pair of joining attributes $A_1, A_2, \ldots, A_{k'}$, mean compatibility degrees between the attribute values from both rowsets and the linguistic value they were assigned to. Since each pair of joining attributes produces two compatibility degrees (see Fig. 10) that are returned together with linguistic terms, in order to reflect how well they contribute to the enrichment of the output rowset, we have to include both of them. We decided that their contribution will be calculated as the mean of both of compatibility (membership) degrees:

$$\mu_{Li}(R.A_i, S.A_i) = \frac{\mu_i(R.A_i, l) + \mu_i(S.A_i, l)}{2}, \tag{23}$$

where $\mu_i(R.A_i, l)$ is the membership (compatibility) degree of the value of the joining attribute A_i from the left rowset (R) of the join and the linguistic value l it was assigned to, and $\mu_i(S.A_i, l)$ is the membership (compatibility) degree of the value of the joining attribute A_i from the right rowset (S) of the join and the linguistic value l it was assigned to (defined in formula 14).

Fuzzy join changes flexibility of queries executed against data lakes. Flexibility of the query Q_F containing the fuzzy join operation expresses the contribution of additional rows R_O^Δ returned by the fuzzy query (with respect to the rowset R_O^N returned by its counterpart with the natural-join operation) in the output rowset R_O^F of the query with the fuzzy join operation:

$$Flex(Q_F) = \frac{card(R_O^\Delta)}{card(R_O^F)} = \frac{card(R_O^F \setminus R_O^N)}{card(R_O^F)}. \tag{24}$$

For example, if the query with the fuzzy equi-join condition $hd.BMI \approx eh.BMI$ returns 100 rows, $card(R_O^F) = 100$, and its counterpart with natural join condition $hd.BMI = eh.BMI$ returns 20 rows, $card(R_O^N) = 20$, the number of additional rows would be 80, $card(R_O^\Delta) = 80$, and the flexibility of the fuzzy query would be $Flex = 0.8$.

Confidence of the output rowset R_O^F allows to assess how much the inclusion of fuzzy join conditions influences the compliance of the output rowset with the join criteria:

$$Conf(R_O^F) = \frac{\sum\limits_{t \in R \times S} \mu_{R_O^F \cap R_O^N}(t)}{\sum\limits_{t \in R \times S} \mu_{R_O^F}(t)} = \frac{\sum\limits_{t \in R \times S} \mu_{R_O^N}(t)}{\sum\limits_{t \in R \times S} \mu_{R_O^F}(t)}, \tag{25}$$

assuming that $R_O^N \subseteq R_O^F$, which always holds.

Uncertainty of the output rowset R_O^F is defined as follows:

$$Uncr(R_O^F) = \frac{\sum\limits_{t \in R \times S} \mu_{R_O^\Delta}(t)}{\sum\limits_{t \in R \times S} \mu_{R_O^F}(t)} = \frac{\sum\limits_{t \in R \times S} \mu_{R_O^F \setminus R_O^N}(t)}{\sum\limits_{t \in R \times S} \mu_{R_O^F}(t)}, \tag{26}$$

$$Uncr(R_O^F) = 1 - Conf(R_O^F). \tag{27}$$

Precision allows to assess the relevance of the output rowset R_O^F. Fuzzy precision is calculated according to the following formula [50]:

$$FPrec = \frac{\sum\limits_{t \in R \times S} \mu_{R_O^F}(t)}{card(R_O^R)}, \tag{28}$$

where R_O^R denotes the set of relevant (correct or expected) rows.

The effectiveness of both techniques for fuzzy join in querying big data with the use of queries shown in Listings 1 and 3 is presented in Tables 1 and 2. Additionally, the last row in Tables 1 and 2 contains values of the same measures for crisp counterparts of the fuzzy join conditions (e.g., natural $hd.BMI = eh.BMI$ vs. fuzzy $hd.BMI \approx eh.BMI$). As can be observed in Table 1 for the fuzzy equi-join performed by representing values of the joining attributes as fuzzy numbers, the decreasing value of the λ-cutoff threshold makes the join operation more flexible. Flexibility of the join operation depends on the assumed λ threshold, the fuzzification of the attributes used in the join condition (used spreads), and the input rowsets (source data). For example, flexibility of the fuzzy query presented in Listing 1 reaches a high level of

Flex = 0.628 (Table 1) already for $\lambda = 0.9$, since there are many rows from both rowsets that match to each other and can be joined. Further increasing of the flexibility of the join operation causes that more and more rows are joined to each other from both rowsets, which is also visible in the growing selectivity (*Selc*) and relative cardinality (*RCard*). For $\lambda = 1.0$ selectivity and relative cardinality are identical, since only rows with $\mu_{R_O}(t) = 1.0$ qualify for the output rowset R_O. Then, their values become increasingly divergent. Larger differences suggest that rows with lower membership degree are joined and returned in the final rowset. For $\lambda = 1.0$ (crisp join condition) the query joins and returns only 1% of rows (*Selc* = 0.01), while for $\lambda = 0.5$ (fuzzy join) it joins and returns 10% of possible rows (*Selc* = 0.104). This shows the enrichment of the output rowset. Uncertainty of the output rowset increases and the confidence of the rowset decreases proportionally with the growing flexibility and decreasing λ. Relatively high values of the uncertainty and low values of the confidence for $\lambda = 0.5$ reflect that within the 9% of additional rows the output rowset was enriched with (in R_O^{Δ}) dominate those that were produced by the fuzzy join (those that have similar, but not equal values of the `BMI` attribute). The quality of the enrichment is reflected in values of the fuzzy precision, which is high, e.g., *FPrec* = 0.729 for $\lambda = 0.5$. This shows that rows were joined with high values of the similarity degree and are highly relevant in the output rowset.

TABLE 1. The effectiveness of the fuzzy join operation performed by representing values of the joining attributes as fuzzy numbers for the sample query from Listing 1 for various λ.

λ	Selc	RCard	Flex	Conf	Uncr	FPrec
1.0	0.010	0.010	0.000	1.000	0.000	1.000
0.9	0.026	0.024	0.628	0.397	0.603	0.937
0.8	0.040	0.036	0.764	0.264	0.736	0.895
0.7	0.056	0.048	0.830	0.199	0.801	0.853
0.6	0.071	0.058	0.865	0.165	0.835	0.815
0.5	0.104	0.076	0.908	0.126	0.874	0.729
0.4	0.119	0.082	0.920	0.117	0.883	0.687
0.3	0.133	0.086	0.928	0.111	0.889	0.649
0.2	0.148	0.090	0.936	0.106	0.894	0.608
0.1	0.162	0.092	0.941	0.103	0.897	0.571
>0.0	0.175	0.094	0.946	0.102	0.898	0.533
Natural	0.010	0.010	0.000	1.000	0.000	1.000

The effectiveness of the fuzzy join performed through assignment to linguistic values can be observed in Table 2. The result of the experiment presented in Table 2 shows that this join technique flexibly joins similar rows that were assigned to the same linguistic term. Flexibility and selectivity remain at the same level as for the first join technique for the $\lambda = 0.5$, and the relative cardinality and the fuzzy precision are even higher. This proves that this join technique provides good enrichment capabilities with high quality of the produced outcome.

Scalability of fuzzy queries performed with the use of FDL4DLA library in the Azure Data Lake environment was presented in our previous work [5]. In terms of the execution time, the fuzzy equi-join performed by assigning attribute values to linguistic terms is faster (5.8 minutes on the tested volume of data) than the fuzzy equi-join performed by representing the attribute values as fuzzy numbers (13.6 minutes on the tested volume of data).

TABLE 2. The effectiveness of the fuzzy join operation through assignment to linguistic values for the sample query from Listing 3.

Join	Selc	RCard	Flex	Conf	Uncr	FPrec
Fuzzy	0.104	0.097	0.908	0.098	0.902	0.936
Natural	0.010	0.010	0.000	1.000	0.000	1.000

VI. DISCUSSION AND CONCLUDING REMARKS

Large volumes of data produced and processed within the cyber-physical systems possess a huge informative potential, which needs to be discovered in order to enrich the spectrum of performed analyses in smart solutions for humanity. Adding some flexibility when combining a variety of data sets during these analyses is one of the important steps of the Data-to-Information Conversion level in the 5C architecture of cyber-physical systems [51]. It is particularly advantageous for those CPSs, which require integration of stream or offline historical data coming from various sensors used in distributed medical monitoring, manufacturing, industrial control systems, intelligent transportation, and other domains.

The fuzzy equi-join presented in the paper provides this flexibility and can be successfully used to combine various, usually Big Data sets produced by sensors and IoT devices, or collected by IoT hubs or monitoring servers. It allows performing the join operation on the basis of elastic joining conditions. The elasticity is achieved by fuzzification of attributes either by representing their values as fuzzy numbers, or by assigning to a linguistic value of the predefined linguistic variable. This leads to two variants of performing the fuzzy join operation. Both of them have some advantages and disadvantages. The first variant of the fuzzy join, which represents attribute values as fuzzy numbers, can be applied quickly to any data without a deep knowledge of the data, the nature and even the domain of the data. The λ-cutoff threshold (minimum membership degree) and spreads used in the fuzzification of attribute values can be determined dynamically, ad hoc, by trials and errors, or can be dynamically regulated, e.g., extended (like the *opening umbrella* in Fig. 3). On the other hand, testing various values of the spread and λ on large data sets may lead to time-consuming calculations. Therefore, it is advised to first make some tests on smaller data sets in order to adjust these values. The second variant of the fuzzy join, which assigns attribute values to linguistic terms, lessens the disadvantage of the dynamic adjustment of particular parameters, but needs the linguistic variable with all the linguistic values to be determined earlier. The linguistic variable, its values (i.e., fuzzy sets), and underlying

membership functions must be defined priorly by the domain specialist (e.g., a production engineer or an information engineer in a factory). This can be difficult in some domains due to the nature of data, nature of a process being monitored, or a lack of the specialist, but once the linguistic variable is defined, the join process occurs according to transparent rules as the domain of data is appropriately divided. Therefore, the first variant of the fuzzy join will be particularly useful when combining those readings from two or more sensors, which are shifted in time, while the second variant will be beneficial if we work on historical, offline data sets and the domain is easily divisible (e.g., according to existing medical standards).

Our solution adds two new variants of the flexible join operation to the collection of join operations mentioned in Sect. II. It extends the group of fuzzy join operations for string data, which calculate the similarity of texts and are not appropriate in processing numerical measurements from sensor-based IoT devices. It allows to perform the fuzzy sets-based join for Big Data lakes, like some of the SQL-based solutions for relational databases, but on a much larger scale, for Big Data sets, and in *schemaless* Big Data Lake environments, which is not possible in highly-structuralized relational databases. Finally, it perfectly complements the fuzzy equivalent join operation presented in [44], which combines tuples from two data sets by calculating the absolute difference of crisp values of the joining attributes and by matching the difference to a predefined fuzzy set of implemented fuzzy comparator. This solution and the two variants of the fuzzy join operation presented in our paper address the characteristics of the 5V model of Big Data, especially the *volume*, the *variety*, and the *velocity*. Both of them allow to perform the join on various data sets and to distribute the join-related data processing in higly-scalable Big Data environments. However, Khorasani *et al.* [44] utilized the Hadoop clusters for the operation and this approach requires implementation of a particular MapReduce job for every join operation performed. Our solution is more flexible in this regard, as users operate in a declarative environment by writing U-SQL scripts with commands similar to the SQL queries. The join operation is a part of the U-SQL query, which is executed in a parallel execution environment of the Azure Data Lake. Both solutions allow to scale computations on the Cloud, which facilitates the accommodation of the dynamic growth of data in cyber-physical systems.

The fuzzy join operations presented in the paper were developed as an extension to the FSL4DLA library [5] for fuzzy data processing and querying in the Cloud. The proposed fuzzy join techniques have a general purpose – they are not limited only to processing and querying biomedical data, but also other types of data. The FSL4DLA library is fully a cloud-dedicated solution, which enables peta-scale storage and wide processing capabilities. This also does involve some consequences and poses some limitations. The necessity to operate on the specific cloud platform and to adapt to the specificity of the Azure Data Lake environment is one of them. Moreover, the use of cloud resources requires a paid subscription for the Microsoft Azure cloud, where the Azure Data Lake is hosted. However, on the other hand, as a cloud-dedicated solution, it exempts possible users from the maintenance of the whole hardware infrastructure kept on the premises.

For the cyber-physical systems that store their data in various data lakes the fuzzy join operation presented in the paper becomes an intuitive alternative to combine related entities. The declarative character of the U-SQL simplifies the implementation of the join operation for users. The Big Data environment of the Azure Data Lake allows to perform fuzzy join operations for large amounts of data from various domains. Finally, clouds enable scaling all computations related to data processing and accommodate the growth of data. These characteristics are highly required as the growth of data in modern cyber-physical systems seems to be unavoidable and the growth rate is constantly increasing.

ACKNOWLEDGMENT

Fuzzy Search Library for Data Lake Analytics (FSL4DLA) is available at (http://www.zti.aei.polsl.pl/w3/dmrozek/science/fsl4dla.htm). Users must use their own Microsoft Azure subscription in order to run the FSL4DLA on the cloud.

REFERENCES

[1] L. Da Xu and L. Duan, "Big data for cyber physical systems in industry 4.0: A survey," *Enterprise Inf. Syst.*, pp. 1–22, 2018, doi: 10.1080/17517575.2018.1442934.
[2] National Research Council, *Frontiers in Massive Data Analysis*. Washington, DC, USA: National Academy Press, 2013.
[3] D. Mrozek, *Scalable Big Data Analytics for Protein Bioinformatics* (Computational Biology), vol. 28. Cham, Switzerland: Springer, 2018.
[4] T. S. J. Darwish and K. A. Bakar, "Fog based intelligent transportation big data analytics in the Internet of vehicles environment: Motivations, architecture, challenges, and critical issues," *IEEE Access*, vol. 6, pp. 15679–15701, 2018.
[5] B. Małysiak-Mrozek, M. Stabla, and D. Mrozek, "Soft and declarative fishing of information in big data lake," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2732–2747, Oct. 2018.
[6] M. Marjani *et al.*, "Big IoT data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[7] A. A. Munshi and Y. A.-R. I. Mohamed, "Data lake lambda architecture for smart grids big data analytics," *IEEE Access*, vol. 6, pp. 40463–40471, 2018.
[8] Q. Zou, X.-B. Li, W.-R. Jiang, Z.-Y. Lin, G.-L. Li, and K. Chen, "Survey of MapReduce frame operation in bioinformatics," *Briefings Bioinf.*, vol. 15, no. 4, pp. 637–647, 2014, doi: 10.1093/bib/bbs088.
[9] G. Wu, Y. He, and X. Hu, "Entity linking: An issue to extract corresponding entity with knowledge base," *IEEE Access*, vol. 6, pp. 6220–6231, 2018.
[10] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating fuzzy duplicates in data warehouses," in *Proc. 28th Int. Conf. Very Large Databases (VLDB)*, P. A. Bernstein, Y. E. Ioannidis, R. Ramakrishnan, and D. Papadias, Eds. San Francisco, CA, USA: Morgan Kaufmann, 2002, pp. 586–597. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781558608696500585
[11] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2003, pp. 313–324, doi: 10.1145/872757.872796.
[12] F. N. Afrati, A. D. Sarma, D. Menestrina, A. Parameswaran, and J. D. Ullman, " Fuzzy joins using MapReduce," in *Proc. IEEE 28th Int. Conf. Data Eng. (ICDE)*, Washington, DC, USA, Apr. 2012, pp. 498–509, doi: 10.1109/ICDE.2012.66.

[13] D. Deng, G. Li, S. Hao, J. Wang, and J. Feng, "MassJoin: A mapreduce-based method for scalable string similarity joins," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 340–351.

[14] B. Kimmett, V. Srinivasan, and A. Thomo, "Fuzzy joins in MapReduce: An experimental study," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1514–1517, Aug. 2015, doi: 10.14778/2824032.2824049.

[15] C. Yan, X. Zhao, Q. Zhang, and Y. Huang, "Efficient string similarity join in multi-core and distributed systems," *PLoS ONE*, vol. 12, no. 3, pp. 1–16, 2017, doi: 10.1371/journal.pone.0172526.

[16] M. Yu, G. Li, D. Deng, and J. Feng, "String similarity search and join: A survey," *Frontiers Comput. Sci.*, vol. 10, no. 3, pp. 399–417, Jun. 2016, doi: 10.1007/s11704-015-5900-5.

[17] A. Das Sarma, Y. He, and S. Chaudhuri, "Clusterjoin: A similarity joins framework using map-reduce," *Proc. VLDB Endowment*, vol. 7, no. 12, pp. 1059–1070, Aug. 2014, doi: 10.14778/2732977.2732981.

[18] G. De Francisci Morales and A. Gionis, "Streaming similarity self-join," *Proc. VLDB Endowment*, vol. 9, no. 10, pp. 792–803, Jun. 2016, doi: 10.14778/2977797.2977805.

[19] D. V. Kalashnikov, "Super-ego: Fast multi-dimensional similarity join," *VLDB J.*, vol. 22, no. 4, pp. 561–585, Aug. 2013, doi: 10.1007/s00778-012-0305-7.

[20] Y. N. Silva, S. S. Pearson, J. Chon, and R. Roberts, "Similarity joins: Their implementation and interactions with other database operators," *Inf. Syst.*, vol. 52, pp. 149–162, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437915000186

[21] S. Fries, B. Boden, G. Stepien, and T. Seidl, "PHiDJ: Parallel similarity self-join for high-dimensional vector data with MapReduce," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 796–807.

[22] Y. Ma, X. Meng, and S. Wang, "Parallel similarity joins on massive high-dimensional data using MapReduce," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 1, pp. 166–183, 2015, doi: 10.1002/cpe.3663.

[23] T. Nie, W.-C. Lee, D. Shen, G. Yu, and Y. Kou, "Distributed entity resolution based on similarity join for large-scale data clustering," in *Web-Age Information Management*, F. Li, G. Li, S.-w. Hwang, B. Yao, and Z. Zhang, Eds. Cham, Switzerland: Springer, 2014, pp. 138–149.

[24] W. Zhao, F. Rusu, B. Dong, and K. Wu, "Similarity join over array data," in *Proc. Int. Conf. Manage. Data*, New York, NY, USA, 2016, pp. 2007–2022, doi: 10.1145/2882903.2915247.

[25] Y. N. Silva, J. Reed, K. Brown, A. Wadsworth, and C. Rong, "An experimental survey of MapReduce-based similarity joins," in *Similarity Search and Applications*, L. Amsaleg, M. E. Houle, and E. Schubert, Eds. Cham, Switzerland: Springer, 2016, pp. 181–195.

[26] B. P. Buckles and F. E. Petry, "A fuzzy representation of data for relational databases," *Fuzzy Sets Syst.*, vol. 7, no. 3, pp. 213–226, 1982. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0165011482900525

[27] S. Shenoi and A. Melton, "Proximity relations in the fuzzy relational database model," *Fuzzy Sets Syst.*, vol. 31, no. 3, pp. 285–296, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0165011489902017

[28] S. Shenoi and A. Melton, "An extended version of the fuzzy relational database model," *Inf. Sci.*, vol. 52, no. 1, pp. 35–52, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020025590900348

[29] H. Prade and C. Testemale, "Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries," *Inf. Sci.*, vol. 34, no. 2, pp. 115–143, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020025584900203

[30] J. M. Medina, M. A. Vila, J. C. Cubero, and O. Pons, "Towards the implementation of a generalized fuzzy relational database model," *Fuzzy Sets Syst.*, vol. 75, no. 3, pp. 273–289, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016501149400380P

[31] P. Bosc and H. Prade, *An Introduction to the Fuzzy Set and Possibility Theory-Based Treatment of Flexible Queries and Uncertain or Imprecise Databases*. Boston, MA, USA: Springer, 1997, pp. 285–324, doi: 10.1007/978-1-4615-6245-0_10.

[32] J. Galindo, A. Urrutia, and M. Piattini, *Fuzzy Databases: Modeling, Design And Implementation*. Hershey, PA, USA: IGI Global, 2006.

[33] K. Myszkorowski, *Inference Rules for Fuzzy Functional Dependencies in Possibilistic Databases*. Cham, Switzerland: Springer, 2016, pp. 181–191.

[34] P. Bosc and O. Pivert, *SQLf Query Functionality on Top of a Regular Relational Database Management System*. Heidelberg, Germany: Physica-Verlag HD, 2000, pp. 171–190.

[35] J. Kacprzyk and S. Zadrożny, *Data Mining Via Fuzzy Querying Over the Internet*. Heidelberg: Physica-Verlag HD, 2000, pp. 211–233.

[36] G. Bordogna and G. Psaila, "Customizable flexible querying in classical relational databases," in *Handbook of Research on Fuzzy Information Processing in Databases*, J. Galindo, Ed. Hershey, PA, USA: IGI Global, 2008, pp. 191–217.

[37] M. Hudec, "An approach to fuzzy database querying, analysis and realisation," *Comput. Sci. Inf. Syst.*, vol. 6, no. 12, pp. 127–140, 2009. [Online]. Available: http://eudml.org/doc/253504

[38] B. Małysiak-Mrozek, S. Kozielski, and D. Mrozek, "Modern software tools for researching and teaching fuzzy logic incorporated into database systems," in *Proc. iNEER Int. Conf. Eng. Educ.*, Gliwice, Poland, Jul. 2010, pp. 1–8.

[39] B. Małysiak, D. Mrozek, and S. Kozielski, "Processing fuzzy SQL queries with flat, context-dependent and multidimensional membership functions," in *Proc. Int. Conf. Comput. Intell. (IASTED)*, M. H. Hamza, Ed. Calgary, AB, Canada: ACTA Press, Jul. 2005, pp. 36–41.

[40] B. Małysiak-Mrozek, D. Mrozek, and S. Kozielski, *Data Grouping Process in Extended SQL Language Containing Fuzzy Elements*. Berlin, Germany: Springer, 2009, pp. 247–256.

[41] L. Portinale and S. Montani, "A fuzzy logic approach to case matching and retrieval suitable to SQL implementation," in *Proc. 20th IEEE Int. Conf. Tools Artif. Intell.* Washington, DC, USA, Nov. 2008, pp. 241–245.

[42] G. A. Lara, M. Delgado, and N. Marín, *Fuzzy Multidimensional Modelling for Flexible Querying of Learning Object Repositories*. Berlin, Germany: Springer, 2013, pp. 112–123.

[43] B. Malysiak-Mrozek, D. Mrozek, and S. Kozielski, "Processing of crisp and fuzzy measures in the Fuzzy Data Warehouse for global natural resources," in *Trends in Applied Intelligent Systems* (Lecture Notes in Computer Science), vol. 6098, N. Garca-Pedrajas and, Eds. Berlin, Germany: Springer, 2010, pp. 616–625.

[44] E. S. Khorasani, M. Cremeens, and Z. Zhao, "Implementation of scalable fuzzy relational operations in MapReduce," *Soft Comput.*, vol. 22, no. 9, pp. 3061–3075, May 2018, doi: 10.1007/s00500-017-2561-3 .

[45] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970, doi: 10.1145/362384.362685.

[46] C. Date, *An Introduction to Database Systems*, 8th ed. Reading, MA, USA: Addison-Wesley, 2003.

[47] B. Małysiak-Mrozek, H. Mazurkiewicz, and D. Mrozek, "Extending the doctrine ORM framework towards fuzzy processing of data," in *Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation*, S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek, and D. Kostrzewa, Eds. Cham, Switzerland: Springer, 2017, pp. 386–402.

[48] B. Małysiak-Mrozek, H. Mazurkiewicz, and D. Mrozek, *Incorporating Fuzzy Logic in Object-Relational Mapping Layer for Flexible Medical Screenings*. Cham, Switzerland: Springer, 2019, pp. 213–233, doi: 10.1007/978-3-319-77604-0_16.

[49] L. A. Zadeh, "A computational approach to fuzzy quantifiers in natural languages," *Comput. Math. Appl.*, vol. 9, no. 1, pp. 149–184, 1983.

[50] B. Ziółko, "Fuzzy precision and recall measures for audio signals segmentation," *Fuzzy Sets Syst.*, vol. 279, pp. 101–111, Nov. 2015.

[51] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.[Online]. Available: http://www.sciencedirect.com/science/article/pii/S221384631400025X

BOŻENA MAŁYSIAK-MROZEK (M'13) received the M.Sc. and Ph.D. degrees in computer science from the Silesian University of Technology, Gliwice, Poland. She is an Assistant Professor with the Institute of Informatics, Silesian University of Technology, and also a member of the IBM Competence Center. She is a coeditor of 14 books devoted to databases and data processing. Her scientific interests cover information systems, computational intelligence, bioinformatics, databases, big data, cloud computing, and soft computing methods. She is a member of the IEEE CI and SMC Societies. She is also a member of the Organizing Committee of the International Conference Beyond Databases, Architectures, and Structures.

ANNA LIPIŃSKA received the B.S. degree in biotechnology and the M.Sc. degree in computer science from the Silesian University of Technology, Gliwice, Poland, in 2015 and 2017, respectively. She is currently a software engineer. Her interests cover Java/Groovy programming, serverless architectures, front-end frameworks, but also bioinformatics and new technologies for humanity.

DARIUSZ MROZEK (M'13–SM'18) received the Ph.D. and D.Sc. degrees from the Silesian University of Technology (SUT), Gliwice, Poland, in 2006 and 2017, respectively. He is currently an Associate Professor and the Head of the Division of Theory of Informatics, Institute of Informatics, SUT. His research interests cover bioinformatics, information systems, parallel and Cloud computing, databases, IoT, and big data. He is currently focused on the analysis of protein structures, functions, and activities, as well as on the use of novel computation techniques to get insights from biological data, including NGS and proteomics data. He is an author of over 90 papers published in conference proceedings and international journals, two books on the use of big data and high-performance computational solutions in protein bioinformatics published by Springer, a coeditor of 14 other books devoted to databases and data processing, and an editor of four special issues of reputable scientific journals.

Dr. Mrozek is a member of the IEEE Engineering in Medicine and Biology Society, the IEEE Systems, Man, and Cybernetics Society, and the IEEE Cloud Computing Community. He has collaborated with qualified institutions by working in different research projects, e.g., the Imperial College of London (on the Chernobyl Tissue Bank), the V. P. Komisarenko Institute of Endocrinology and Metabolism—the Academy of Medical Sciences of the Ukraine, Medical Radiological Research Centre—the Russian Academy of Medical Sciences, the Helmholtz Zentrum Muenchen Deutsches Forschungszentrum Fuer Gesundheit und Umwelt Gmbh, Microsoft Research, USA, the Institute of Oncology, the Department of Internal Diseases, Diabetology and Nephrology, the Medical University of Silesia, Zabrze, Poland, and the Department of Biochemistry, Medical University of Silesia, Katowice, Poland.

Dr. Mrozek was twice a Laureate of Microsoft Azure for Research Award by Microsoft Research granted in 2014 and 2016.