Received February 27, 2020, accepted March 9, 2020, date of publication March 13, 2020, date of current version March 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980553

Uncertainty Modeling of Object-Oriented Biomedical Information in HBase

LEI ZHANG[1,*], JIALIANG SUN[2,*], SHUHUI SU[2], QIURU LIU[2], AND JIAN LIU[2]
[1]School of Biological and Chemical Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Corresponding author: Jian Liu (jianliu@hit.edu.cn)

*Lei Zhang and Jialiang Sun are co-first authors.

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC1200205 and Grant 2017YFC1200200, and in part by the Zhejiang Province Key Research and Development Program under Grant 2020C03071, Grant 2017C02009, and Grant 2017C03024.

ABSTRACT While object-oriented databases (OODBs) are known to be rich in functionality, HBase database, which is a distributed and scalable big data store, as well as uncertain databases have recently gained a lot of attention in the database community. This paper presents a methodology for handling an important step of knowledge integrations and migrations. In particular, a formal approach for reengineering fuzzy object-oriented databases in HBase is firstly developed. The reengineering approach is based on the technique of rule-based schema mapping, which defines a set of transformation rules involved in the process of schema transformations for mapping a fuzzy object-oriented database schema into a fuzzy HBase database schema. In addition, a formal approach to map the fuzzy object-oriented algebra into fuzzy HBase algebra is proposed. On this basis, we complement the work with a comprehensive set of experiments to show the efficiency of our proposed approach in terms of query time and scalability metrics.

INDEX TERMS Big data store, fuzzy HBase, fuzzy object-oriented database, mapping.

I. INTRODUCTION

In current computer science and medicine fields, since the notion of object and class is ubiquitous in many real-world applications, there is a paradigm shift to the object orientation in the formalisms for representations of the structured knowledge used both in knowledge representations and in databases. Object-oriented databases (OODBs) are widely used for supporting advanced database applications [1] such as such as the systems biology modeling [2], the object-oriented biomedical continuous system modeling [3], and object-oriented user interfaces for the bioinformatics analysis pipeline systems [4]. An object-oriented knowledge base [5] called SENEX is developed for modeling and representing the biomedical information about the neurodegeneration and loss of memory in aging. OODBs have several advantages: increased data representation capabilities including the representations of the abstract data types or meta-data, navigational access to data, encapsulation of procedural and declarative knowledge, managements of class extensions, dynamic class definitions, etc [6]. With the popularity of object-oriented technology, many prototypes and commercial object-oriented databases (OODBs) have been developed by industrial and research laboratories. In [7]–[10], In the real world, an application that involves the analysis and the management of databases often involves imperfect information. In many domains, e.g., the high-throughput DNA sequencing [11], the precision medicine [12], the structural biology research [13] and diagnostic practice [14], it is difficult to state all information with one hundred percent certainty [15]–[17]. Influenced by objective factors such as instrument precision, there is uncertain information in the original sequencing data. Through the quality control of sequencing data, we remove the low-quality data before further analysis. In addition, uncertainty surrounding etiology and diagnosis, as well as treatment and aftercare widely exist in practical applications. These drive researchers to develop specific solutions to provide supports for the uncertain data processing, and the research on fuzzy object-oriented database management is extensively under way [18]–[22].

In recent years, average sizes of corporate databases tend to be in the range of Gigabytes (GB). With the advent of the era for big data, a database with a multi-Terabyte (TB) or even Petabyte (PB) size becomes normal [23]. Massive knowledge representation and management is emerged as a challengeable issue in big data era. Managing and storing knowledge in traditional databases [24]–[26] could no longer fulfill the increasing requirements of dealing with massive data. There are number of projects (e.g., [27], [28]) have been developed as an alternative to traditional database systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Ying Li.

Due to the advantages in distributed and parallel processing, as well as knowledge applications such as massive knowledge managements, etc, the open-source, reliable and scalable distributed database system HBase [29], designed for the large-scale distributed data storage and the high-performance computation, has attracted much attention both in academia and industry [30], [31].

The requirement of the interoperability of autonomous databases leads to the multi-database systems [32]–[37], which often consists of homogenous or heterogeneous databases. In the heterogeneous multi-database system, the transformation between operations of databases with different data models is critical [38]. Actually, there exist many legacy databases (such as object-oriented databases, and so on) that are in need of modernization [39] in order to be compatible and competitive in this new era of big data ubiquity. Now database administrators are faced with the challenge of ensuring their databases which can interface with other big data management systems (such as HBase, etc), and mapping mechanisms from existing legacy databases to databases designed for large-scale data managements are proposed [40]–[42].

As the integration of uncertain and scalable aspects is necessary in the next generation information systems, fuzzy object-object database is a promising database application and currently, HBase is widely used in bioinformatics community and it may be the next generation database to meet advanced big biomedical data management requirements. In order to effectively manage massive biomedical data, approaches concerned with the reengineering traditional databases in HBase are greatly needed. The adoption of the HBase naturally triggers the requirement of the mapping from the historical one to the new one [43], [44]. Although HBase is employed to model the future data explosion, little work focuses on the uncertainty modeling of object-oriented biomedical information in HBase. Developing an effective mapping technique, which can deal with uncertainty modeling and schema mapping from the fuzzy object-object biomedical database model to the HBase model in a uniform way is still an open problem. In order to solve the mapping problems, in this paper, we study the methodology of uncertainty modeling of object-oriented biomedical information in HBase. In particular, we present a novel rule-based approach to transform a fuzzy object-oriented biomedical database schema into an HBase database schema. On this basis, a formal approach to map the fuzzy object-oriented algebra into fuzzy HBase algebra is proposed. To the best of our knowledge, this is the first effort on the construction of the fuzzy HBase database model from fuzzy object-oriented database models.

Our contributions in this paper can be summarized as follows:

- We study the methodology of uncertainty modeling of object-oriented biomedical information in HBase and present a rule-based approach to achieve the schema transformation.
- We propose a novel approach to map the fuzzy object-oriented algebra into fuzzy HBase algebra.
- We present an extensive experimental evaluation which proves the efficiency of our proposal on the tested data.

The rest of the paper is organized as follows. Section 2 gives the preliminaries of fuzzy object-oriented and fuzzy HBase databases. The uncertainty modeling of object-oriented biomedical information in HBase is presented in Section 3. Section 4 contains the experiments and Section 5 concludes the paper.

II. PRELIMINARIES

A. FORMALIZATION OF FUZZY OBJECT-ORIENTED BIOMEDICAL DATABASES

By summarizing the characteristics of fuzzy object-oriented databases in the literature [22], [45], [46], [47], in the following we introduce some basic notions of fuzzy object-oriented biomedical databases.

The basic notions of fuzzy object-oriented biomedical databases consist of fuzzy object, fuzzy class, fuzzy inheritance, integrity constraints and fuzzy object-oriented database (algebraic) operations. An object is fuzzy because of a lack of information. A fuzzy object can be represented by a 3-tuple (o, vi, ρ (vi)), where o is a unique and immutable object identifier, vi is a value set, and ρ (vi) ($0< \rho$ (v_i) ≤ 1) is a possibility connecting with vi. The object identifier of a fuzzy object is unaffected by changes to the object's value, and the possibility connecting with a value can be omitted if the possibility is equal to 1.0. For example, the following object represents a fuzzy genome annotation object:

(o_1, [pos: "14370", REF: "G"], [ρ (14370): "0.9", ρ (G): "0.8"])

Here, o1 is the object identifier, pos and REF are attribute names, 14370 and G are the values of those attributes respectively, 0.9 and 0.8 are the possibilities of the values 14370 and G respectively.

The objects with the same properties are gathered into classes organized into hierarchies. Theoretically, a class could be considered from two viewpoints [22]: i) an extensional class, where the class is defined by a list of its object instances, and ii) an intensional class, where the class is defined by a set of attributes and their values. A class is fuzzy because: i) a class defined by fuzzy objects could be fuzzy. In this scenario, these fuzzy objects belong to the class with a possibility ρ ($\rho \in [0, 1]$). ii) if a class is intensionally defined, the domain of an attribute may be fuzzy and a fuzzy class is formed. For instance, a class "young people" is a fuzzy class because the domain of its attribute "age" may be a set of fuzzy values such as {45/0.2, 40/0.3, 35/0.4}, that is, the possibility of 45 is 0.2, the possibility of 40 is 0.3, and the possibility of 35 is 0.4. iii) the subclass produced by a fuzzy class by means of specialization and the superclass produced by some fuzzy classes by means of generalization are also fuzzy. Following the previous works on uncertainty modeling such as the work [48], a sound uncertainty model usually contains three levels of uncertainty. To be consistent with previous studies, we also use these three levels of uncertainty in our work. We introduce three levels of uncertainty [38] to the classes in fuzzy object-oriented databases as follows:

- At the first level, class and its attribute sets could be fuzzy, i.e., they have a possibility to the model.
- The second one is related to fuzzy occurrences of objects.
- The third one concerns fuzzy values of attributes of special objects.

In order to model the first level of uncertainty, i.e., an attribute or a class with a possibility, the attribute or class name should be followed by a pair of words WITH i DEGREE, where $0 \leq i \leq 1$ is a scalar and it is used to indicate the degree that an attribute belongs to a class or a class belongs to a data model. For example, assume that we have a class *INFO* with an attribute *AA* in a biomedical data model about genomic annotations. This biomedical data model contains several classes, and class *INFO* may or may not be necessary to be included in the data model. Additionally, *INFO* consists of some attributes, and *AA* may or may not be necessary to be included in this class. Assume that we have *INFO WITH 0.8 DEGREE* and *AA WITH 0.6 DEGREE*, which means class *INFO* and attribute *AA* are both with the first level of the uncertainty. Here, *INFO* belongs to the data model with 0.8 degree, and *AA* belongs to the class with 0.6 degree. Generally, "WITH 1.0 DEGREE" can be omitted when the degree of an attribute or a class is 1.0. In order to model the third level of uncertainty (i.e., attribute values are fuzzy), a keyword FUZZY could be introduced and it is added in the attribute. As to the second level of uncertainty, we need indicate the possibility that an object of the class belongs to this class. To this purpose, an additional attribute is introduced to represent the possibility belonging to the class, whose domain is [0,1]. We denote this special attribute with μ in this paper. In order to differentiate the class with the second level of fuzziness, we could use a dashed-outline rectangle to denote this kind of class.

Figure 1 shows a fuzzy class *INFO*. For the sake of simplification, some components such as superclass, method and

```
CLASS INFO
INHERIT …
ATTRIBUTES
  NS: TYPE OF number
  DP: TYPE OF number
  AF: TYPE OF string
  OMIM: TYPE OF string
  AA: FUZZY TYPE OF string
  μ: DOMAIN [0, 1] TYPE OF number
METHODS
…
ENDS
```

FIGURE 1. **A fuzzy class.**

constraints, etc, are not listed. *OMIM* may have fuzzy values, that is, its domain is fuzzy, in this scenario a fuzzy type (depicted as FUZZY TYPE OF) of number is introduced. Since INFO may or may not have AA information, when class *INFO* is given, it is not sure whether *AA* should be included in *INFO*. That is to say, *AA* uncertainly belongs to *INFO*. A possibility ρ could be assigned to *AA* with regard to *INFO*. *WITH* ρ *DEGREE* is used to describe the first level uncertainty in the class definition. Since we may not determine whether an object is the instance of a class because the class is fuzzy, an additional attribute μ whose domain is [0, 1] is added into *INFO*.

The formal description to fuzzy object-oriented biomedical databases contains three elements: fuzzy biomedical class, fuzzy inheritance relationship definition between classes, and algebraic operations. In the following, the definition of fuzzy biomedical class is firstly provided.

Definition 1: A fuzzy biomedical class is defined as a tuple (C_{sup}(c), C_{sub}(c), L_{datt}(c), L_{uatt}(c), L_{ins}(c), L_{pos}(c)), where c is class identifier, C_{sup}(c) is a list of direct superclasses of class c, C_{sub}(c) is a list of direct subclasses of class c, L_{datt}(c) is the set of additional deterministic attributes locally defined in c, L_{uatt}(c) is the set of additional fuzzy attributes locally defined in c, L_{ins}(c) is the set of object identifiers of objects added locally to c (An instance of an object schema maps each class name to a set of OIDs, and maps each OID to a value), and L_{pos}(c) is the set of possibility of an object (or attribute) added locally to c.

For the fuzzy inheritance relationship, we have the following definition. In particular, fuzzy inheritance relationships between classes define a partial ordering called a fuzzy superclass/subclass relationship. Conceptually, if sc is in a fuzzy subclass relationship with sc', then the set of objects represented by c (depicted as s(c)) is a subset of the objects represented by sc' with possibilities. In this scenario, for any object, if the possibility that it belongs to the subclass is no more than the possibility that it belongs to the superclass, and the possibility that it belongs to the subclass is no less than a given threshold β. In particular, let c1 and c2 be fuzzy classes and β be a given threshold, we say c2 is a subclass of c1 if $s(c2) \subseteq s(c2)$, and $(\forall o)(\ \beta \leq \rho_{c2}(o) \leq \rho_{c1}(o))$, where $\rho_{c2}(o)$ is the possibility that it belongs to the subclass, and $\rho_{c1}(o)$ is the possibility that it belongs to the superclass.

For the inheritance hierarchies [49], two constraints are maintained in the fuzzy object-oriented database system:

- Cover: a set of subclasses $C_1, \ldots C_k$ of a class C covers a class C if every instance of C is an instance of one of these subclasses, i.e., $\vartheta(C_1) \cup \ldots \cup \vartheta(C_k) = \vartheta(C)$, and $\rho(C_1), \ldots, \rho(C_k) \leq \rho(C)$, where $\vartheta(C)$ is the set of instances of class C, $\rho(C)$ is the corresponding possibility. The cover constraint is introduced in the declaration of one of these classes with: C_i *inherit* C *cover with* (C_1, $\ldots C_{i-1}$, C_{i+1}, $\ldots C_k$).
- Partition: a set of subclasses $C_1, \ldots C_k$ of a class C partitions a class C if every instance of C can be an instance of one of these subclasses only, i.e., for all i, j, we have

$\vartheta(C_i) \cap \vartheta(C_j) = \emptyset$, and $\rho(C_1), \ldots, \rho(C_k) \leq \rho(C)$. The partition constraint is introduced in the declaration of one of these classes with C_i *inherit* C *partition with* $(C_1, \ldots C_{i-1}, C_{i+1}, \ldots C_k)$.

For the referential integrity in fuzzy object-oriented databases, it requires that any object referenced by another object actually exists. When a deterministic object is deleted, one can handle remaining references to it by either deleting the reference (replacing it with a null), or deleting the referencing object (cascading the deletion process). When a fuzzy object is deleted, one can handle remaining references to it by either deleting the reference and the corresponding possibility, or deleting the referencing object. The referential integrity could be supported by specifying one of these maintenance options for a reference attribute in the attribute declaration.

For the algebraic operations, we have the following formal description. In particular, fuzzy object-oriented databases provide algebraic operations as a basis for database manipulation languages. Primitive algebraic operators are class construction $C_c(C, S_c, S_a, S_p)$, object construction $I_o(C, S_v, S_p)$, object union $U_o(C, S_v, S_p)$, object difference $D_o(C, S_v, S_p)$, selection $S_o(C, F_c, S_a)$, and update $U_o(C, F_c, S_a, V_c)$.

- class construction $C_c(C, S_c, S_a, S_p)$: create class C as a subclass of classes in set S_c with additional attributes in set S_a and possibilities in set S_p.
- object construction $I_o(C, S_v, S_p)$: create an object of class C with values in set S_v and corresponding possibilities in set S_p.
- object union $OU_o(C, S_v, S_p)$: unite two objects o_i and o_j as an object o with the union of each value of the same attribute in set S_v, and the maximal possibility (depicted as $\max(\rho(o_i), \rho(o_j))$, where $\rho(o_i)$ and $\rho(o_j)$ are related possibilities, $\rho(o_i)$, $\rho(o_j)$ $\in(0, 1]$) of the corresponding possibilities in set S_p. It should point out that, for the object union, two objects involved must be union-compatible, i.e., they must have the same set of attributes.
- object difference $D_o(C, S_v, S_p)$: the object difference of object o_j from o_i, is the set of all values of set S_v in o_i but not in o_j, and the minimal possibility of the corresponding possibilities $\rho(o_i)$ and $(1-\rho(o_j))$ in set S_p(depicted as $\min(\rho(o_i), (1-\rho(o_j)))$, where $\rho(o_i)$ and $\rho(o_j)$ are related possibilities, $\rho(o_i)$, $\rho(o_j)$ $\in(0, 1]$). For the object difference, the two objects involved must be union-compatible.
- selection $S_o(C, F_c, S_a)$: select values, satisfying a given fuzzy selection condition specified by a fuzzy expression combining the basic clause $A\theta B$ in set F_c of attributes in set F_c of class C. Since the fuzzy selection condition may be fuzzy, the evaluation of the fuzzy expression can be conducted by using Zadeh's extension principle [22], [50], where $\theta \in \{<_\omega, =_\omega, >_\omega, \leq_\omega, \geq_\omega, \neq_\omega\}$, ω is a threshold.
- update $U_o(C, F_c, S_a, V_c)$: update values which satisfy fuzzy selection condition in set F_c, of attribute S_a of class C with new value V_c.

B. FORMALIZATION OF FUZZY HBase DATABASES

As introduced in [51], from a logical point of view, data in HBase are organized in labeled tables. All table accesses are via the table primary key and any scan of HBase table results into a MapReduce job [30], [52]. By summarizing the characteristics of HBase databases in the literature [53], in the following we will introduce the formalization of fuzzy HBase databases. The basic notions of fuzzy HBase databases consist of fuzzy HBase table, HBase integrity constraints and fuzzy HBase database (algebraic) operations. To model the uncertainty in HBase, three levels of uncertainties occurring in a fuzzy HBase table are considered [53]:

- At the first level, column families or columns may or may not occur, that is, they have some possibility to the given model.
- The second one, a column family may or may have rows associated with its columns, that is, rows may have some possibility belonging to the column family.
- The third one, some columns may have fuzzy values.

To model the first level of uncertainty in HBase, a column (or a column family) depicted by a pair of words with ρ possibility $\rho \in [0, 1]$ is introduced to indicate the possibility of an HBase table having the column family or column is ρ. Note that, ρ can be omitted when the possibility is 1.0. For the second level of uncertainty, a possibility column μ, $\mu \in [0, 1]$ is added to indicate the possibility of a column family having a row i μ. For the third level of uncertainty, a set of possible values (possibility distribution) associated with the column is used to indicate the values' possibilities. For example, the following represents the first level of uncertainty in a fuzzy HBase table:

(cf, [columnFamily: "INFO", column: "AA"], [ρ(INFO):" with 0.8 possibility", ρ(AA):"with 0.3 possibility"]).

Here, cf is the column family or column identifier, *columnFamily* and *column* are column family and column names, *INFO* and *AA* are the values of those column family and column respectively, *with 0.8 possibility* and *with 0.3 possibility* are the possibilities of the values *INFO* and *AA* belonging to the fuzzy HBase table respectively.

The following represents the second level of uncertainty in a fuzzy HBase table:

(r_1,[columnFamily:"INFO"-columnRange:"1-3"], [ρ(r_1:INFO: 1-3):"0.5"]).

Here, r_1 is the row identifier, *columnFamily* and *columnRange* are column family names and column ranges, *car* and "1-3" are the values of the column family and column range respectively, "0.5" depicts that the possibility of this row (columns 1-3) being a member of the column family *car* is 0.5.

The following depicts the third level of uncertainty in a fuzzy HBase table:

(r_1,[columnFamily:"INFO"-column:"OMIM"-value:" DOID:0110493,HP:0000007,HP:0000407"], [ρ(v_1):"0.5, 0.6, 0.3"]).

TABLE 1. An example of a fuzzy HBase table.

Row Key	Timestamp	INFO with 0.8 possibility		
		NS	OMIM-value	μ
r1	t2	3	HP:0000307	0.3
	t1			
r2	t4	2	{DOID:0110493/0.5,HP:0000007/0.6, HP:0000407/0.3}	0.8
	t3			
…	…	…	…	…

Here, r_1 is the row identifier, *columnFamily*, *columnRange* and *value* are column family and column names, *INFO* and *OMIM* are values of the column family and column, and "DOID:0110493,HP:0000007,HP:0000407" are the possible values of this column in r_1, "0.5, 0.6, 0.3" depicts that possibilities of these values are 0.5, 0.6 and 0.3 respectively.

Consider the fuzzy HBase table in Table 1, where μ denotes the possibilities of related rows belonging to the corresponding column family. *INFO with 0.8 possibility* is a column family with the first level of uncertainty. In the first row of table 1, for the NS 3 whose OMIM-value is HP:0000307, the possibility of this row being a member of the column family *INFO* is 0.3. The first row of table 1 is an instance of the second kind of uncertainty in the fuzzy HBase table. For the NS 2, if its OMIM-value is unknown so far, i.e., it has a fuzzy value in the column *cost*, which could be represented by using a possibility distribution, for example, {DOID:0110493/0.5, HP:0000007/0.6, HP:0000407/0.3}. This salary column is an instance of the third kind of uncertainty in the fuzzy HBase table.

Definition 2: A fuzzy HBase table is defined as a tuple (T_{dcf}(t), T_{ucf}(t), T_{dcol}(t), T_{ucol}(t), T_{ins}(t), T_{pos}(t)), where t is object identifier, T_{dcf}(t) is a list of deterministic column families of t, T_{ucf}(t) is a list of fuzzy column families of t, T_{dcol}(t) is the set of additional deterministic column of t, T_{dcol}(t) is the set of additional fuzzy columns of t, T_{ins}(t) is the set of values of column added locally to t, and T_{pos}(t) is the set of possibilities occurring in t.

The main constraints in fuzzy HBase models are domain integrity constraints and cell integrity constraints. The contents of domain integrity constraints in fuzzy HBase are that column values should be the values in the domain. In particular, for all values $v_1, \ldots v_k$ of a column *Col*, every instance of *Col* is one of these values, i.e., $\chi(v_1)\cup\ldots\cup\chi(v_k) = \chi(\text{Col})$, where χ(C) is the set of instances of the values in *Col*. The contents of cell integrity constraints in fuzzy HBase are that each nonempty cell should have an identified key and the value of the identified key should be sole and cannot be null.

Next we will introduce the algebraic operations in fuzzy HBase databases. Primitive algebraic operators are column family construction CF_{cf}(CF, S_{cf}, S_{col}, S_p), row object construction RO_{ro}(RO, S_{cf}, S_{col}, S_v, S_p), mapper construction Map_{map}(M, S_{col}, S_v, S_p), reducer construction Red_{reduce}(R, S_{col}, S_v, S_p), row object union U_{rou}(U, S_{cf}, S_{col}, S_v, S_p), row object difference D_{rod}(D, S_{cf}, S_{col}, S_v, S_p), HBase selection $S_{hselection}$(HO, F_c, S_a), and HBase update $U_{hupdate}$(HO, F_c, S_a, V_c).

- column family construction CF_{cf}(CF, S_{cf}, S_{col}, S_p): create column family CF in set S_{fc} with additional columns in set S_{col} (S_{col} $=S_{ucol}\cup S_{dcol}$, S_{ucol} is the set of fuzzy columns and S_{dcol} is the set of deterministic columns) and corresponding possibilities in set S_p.
- row object construction RO_{ro}(RO, S_{cf}, S_{col}, S_v, S_p): create a row object RO with column families in set S_{cf}, columns in set S_{col}, values in set S_v and corresponding possibilities in set S_p.
- mapper construction Map_{map}(M, S_{cf}, S_{col}, S_p): create a mapper M with columns in set S_{col} and their values of each column family in set S_{cf} associated with the corresponding corresponding possibilities in set S_p. Mapper construction performs filtering and sorting (e.g., sorting teachers by name into queues, one queue for each name).
- reducer construction Red_{reduce}(R, S_{cf}, S_{col}, S_p): create a reducer R with columns in set S_{col} and their values of each column family in set S_{cf} associated with the corresponding possibilities in set S_p. Reducer construction performs a summary operation (such as counting the number of students in queue, yielding name frequencies).
- row object union U_{rou}(U, S_{cf}, S_{col}, S_v, S_p): unite two row objects ro_i and ro_j as a row object ro with the union of each value of the same column family and the same column in set S_v, and the maximal possibility (depicted as $\max(\rho(ro_i), \rho(ro_j))$, where $\rho(ro_i)$ and $\rho(ro_j)$ are related possibilities, $\rho(ro_i), \rho(ro_j) \in (0, 1]$) of the corresponding possibilities in set S_p. It should pointed out that, for the object union, the two row objects involved must be union-compatible, i.e., the two row objects must have the same set of columns.
- Row object difference D_{rod}(D, S_{cf}, S_{col}, S_v, S_p): the row object difference of object ro_j from ro_i, is the set of all values of set S_v in ro_i but not in ro_j, and the minimal possibility of the corresponding possibilities $\rho(ro_i)$ and $(1-\rho(ro_j))$ in set S_p(depicted as $\min(\rho(ro_i), (1-\rho(ro_j)))$, where $\rho(ro_i)$ and $\rho(ro_j)$ are related possibilities, $\rho(ro_i), \rho(ro_j) \in (0, 1]$). For the row object difference, the two row objects involved must be union-compatible.
- HBase selection $S_{hselection}$(HO, F_c, S_{cf}, S_{col}): select values, satisfying a given fuzzy selection condition specified by a fuzzy expression combining the basic clause $A\theta B$ (this fuzzy expression could be conducted by using Zadeh's extension principle [22, 50] computing on the corresponding column family and column, where $\theta \in \{<_\omega, =_\omega, >_\omega, \leq_\omega, \geq_\omega, \neq_\omega\}$, ω is a threshold.) in set F_c of attributes in set F_c of HBase row object HO.
- HBase update $U_{hupdate}$(HO, F_c, S_{cf}, S_{col}, V_c): updates values which satisfy fuzzy selection condition in set F_c of the corresponding column family and column in S_{cf} and S_{col}, respectively, of HBase row object HO with new value V_c.

III. UNCERTAINTY MODELING OF OBJECT-ORIENTED BIOMEDICAL INFORMATION IN HBase

In this section, we concentrate on the formal mapping from fuzzy object-oriented biomedical database to HBase by using transformation rules. In particular, the defined rules are used for the class mapping (Rules 1-7), the inheritance relation mapping (Rule 8) and the database operations mapping (such as class construction associated with Rule 9, object construction associated with Rule 10, the selection associated with Rule 11, the update associated with Rule 12, object union associated with Rule 13, object difference associated with Rule 14), separately. And these rules are not conflicted.

A. TRANSFORMATION OF FUZZY CLASSES

The uncertainty modeling of object-oriented biomedical information and the corresponding schema mapping could be established by a series of mapping rules as follows.

Rule 1: For a fuzzy object-oriented biomedical schema S, a column named *Timestamp* is created in HBase.

Rule 2: For each class C in a fuzzy object-oriented biomedical schema S, a column family named $S.C$ is created in HBase.

Rule 3: For each Object identifier (OID) with the name O_i in class C of fuzzy object-oriented schema S, a column concatenating all (distinct) Object identifiers O_i as the row key column of the (fuzzy) HBase table, and a column with the same name as O_i are created in the column family $S.C$.

Rule 4: For each deterministic non-OID attribute A_i in class C of fuzzy object-oriented schema S, a column with corresponding name A_i is created in the column family $S.C$.

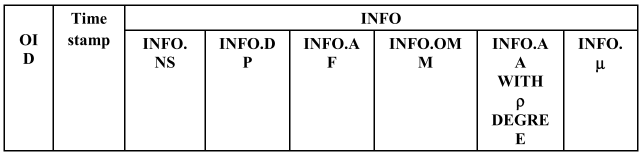
Rule 1-4 could be directly used to transform a deterministic object-oriented biomedical class into a deterministic HBase table when all the classes and attributes are deterministic. As soon as there exists the uncertainty in object-oriented biomedical databases, the following rules (Rule 5-7) can be used to accomplish the fuzzy class schema mapping.

Rule 5: If there exists the first level of uncertainty in object-oriented biomedical databases (recall the discussions of the uncertainty in Section 2.1), for each non-OID fuzzy attribute A_i in class C of fuzzy object-oriented schema S, a column with the same name as A_i *with* ρ *possibility* is created in the column family $S.C$.

Rule 6: If there exists the second level of uncertainty in object-oriented biomedical databases (recall the discussions of the uncertainty in Section 2.1), for each fuzzy class C of fuzzy object-oriented schema S, an additional column with the name as $\rho.\ S.C$ is created in the column family $S.C$.

Rule 7: If there exists the third level of uncertainty in object-oriented biomedical databases (recall the discussions of the uncertainty in Section 2.1), for each non-OID fuzzy attribute A_i in class C of fuzzy object-oriented schema S, an additional column with the same name as A_i, which takes a fuzzy value represented by using a possibility distribution, is created in the column family $S.C$.

Figure 2 shows the transformation of the fuzzy object-oriented biomedical class shown in Figure 1 to the fuzzy HBase table. From Figure 1, we know that fuzzy class *Student* has three levels of uncertainties. Firstly, we use Rule 1 and 2 to create the *Timestamp* column and the column family for the construction of uncertainty class *INFO* in HBase. For the OID attribute and deterministic non-OID attributes of fuzzy class *INFO*, Rule 3 and 4 are used for the fuzzy class construction in HBase. Then for the uncertainty in class *INFO*, Rule 5, 6 and 7 are used to create the corresponding HBase columns for the first, second and third levels of uncertainties in fuzzy class *INFO*, respectively. Finally, we create the row key for the generated HBase table based on Rule 3 to accomplish the construction of fuzzy class *INFO* in HBase.



OID	Time stamp	INFO					
		INFO.NS	INFO.DP	INFO.AF	INFO.OMM	INFO.AA WITH ρ DEGREE	INFO.μ

FIGURE 2. Construction of fuzzy biomedical class INFO in HBase table.

B. TRANSFORMATION OF FUZZY INHERITANCE

In this section, the construction of fuzzy HBase tables based on the fuzzy inheritance (see Section 2.1) in object-oriented biomedical databases is investigated. Following the transformation of classes given in Section 3.1, the superclass and each subclass are firstly transformed and then the extended tag is used to identify the constraints of the fuzzy inheritance. In particular, the following mapping rule could be used to accomplish the transformation of the fuzzy inheritance.

Rule 8. For each subclass $\mathrm{C_i}$ in fuzzy object-oriented biomedical databases, let its superclass be C, (i) if $\mathrm{C_i}$ *inherit* C *cover with* $(\mathrm{C_1}, \ldots \mathrm{C_{i-1}}, \mathrm{C_{i+1}}, \ldots \mathrm{C_k})$, Rules 1-7 are firstly used to transform the classes and column families *S.* $\mathrm{C_i}$ and *S.* C are created for classes $\mathrm{C_i}$ and C respectively, and an additional column $\xi\ .C_i\ \text{-}C$ is created in column family *S.* $\mathrm{C_i}$ to identify the cover constraint between subclass $\mathrm{C_i}$ and its superclass C, and (ii) if $\mathrm{C_i}$ *inherit* C *partition with* $(\mathrm{C_1}, \ldots \mathrm{C_{i-1}}, \mathrm{C_{i+1}}, \ldots \mathrm{C_k})$, Rules 1-7 are used to transform the classes and column families *S.* $\mathrm{C_i}$ and *S.* C are created for classes $\mathrm{C_i}$ and C respectively, and an additional column named $\zeta\ .C_i\ \text{-}C$ is created in column family *S.* $\mathrm{C_i}$ to identify the partition constraint between subclass $\mathrm{C_i}$ and its superclass C.

C. TRANSFORMATION OF FUZZY DATABASE OPERATIONS

In this section, we will introduce the transformation from fuzzy object-oriented algebraic operations to the fuzzy HBase algebraic operations. The detailed transformation could be accomplished by using the following mapping rules.

Rule 9: For the class construction $\mathrm{C_c(C, S_c, S_a, S_p)}$ in the fuzzy object-oriented algebra, it can be mapped into the column family construction of the fuzzy HBase algebra, where the object-oriented class name C in class set $\mathrm{S_c}$, attributes $\mathrm{S_a}$, and possibilities $\mathrm{S_p}$ are mapped into the corresponding HBase column family, columns and possibilities.

Rule 10: For the object construction $I_o(C, S_v, S_p)$ in the fuzzy object-oriented algebra, it can be mapped into the row object construction of the fuzzy HBase algebra, the values S_v and possibilities S_p of the given class C are mapped into the corresponding HBase column values and possibilities of the transformed column family C.

Rule 11: For the selection $S_o(C, F_c, S_a)$ in the fuzzy object-oriented algebra, if the fuzzy selection condition F_c conducts at a single attribute, then the attribute and its corresponding selection condition F_c of the given class C can be mapped into the selection that satisfies the given fuzzy selection condition executed on the corresponding HBase column of the transformed column family C. Otherwise, if the fuzzy selection conditions F_c conduct at multiple attributes, then the attributes and their corresponding selection conditions F_c of the given class C can be mapped into the selections that satisfy the given fuzzy selection conditions executed on the corresponding HBase columns of the transformed column family C.

Rule 12: For the update $U_o(C, F_c, S_a, V_c)$ in the fuzzy object-oriented algebra, the attributes S_a satisfying selection conditions F_c and their updated values V_c of the given class C are mapped into the updates that satisfy the given fuzzy selection conditions executed on the corresponding HBase columns of the transformed column family C.

Rule 13: For the object union $OU_o(C, S_v, S_p)$ in the fuzzy object-oriented algebra, the values S_v and possibilities S_p of the given class C can be mapped into the set union of the corresponding HBase column values and maximal values of related possibilities of the transformed column family C.

Rule 14: For the object difference $D_o(C, S_v, S_p)$ in the fuzzy object-oriented algebra, e.g., the object difference of object o_j from o_i, where the corresponding possibilities of o_i and o_j are $\rho(o_i)$ and $(1-\rho(o_j))$ respectively, the values S_v and possibilities S_p of the given class C can be mapped into the set difference of the corresponding HBase column values and the minimal possibility of the corresponding possibilities $\rho(o_i)$ and $(1-\rho(o_j))$ of the transformed column family C.

Algorithm 1 OO2HB(o)

```
01 while not end(o)
02 $t_{act}$ = getOOSchema(o)
03 $S_{act}$ = getHBaseSchema($t_{act}$, $T_{dcf}(t_{act})$, $T_{ucf}(t_{act})$, $T_{dcol}(t_{act})$, $T_{ucol}(t_{act})$, $T_{ins}(t_{act})$, $T_{pos}(t_{act})$)
   //translating the schema o of $t_{act}$ into the fuzzy HBase schema based on Rules 1-7
04 for each object $object_i$ in $t_{act}$
05 $T_{dcol}(object_i)$ = getValues($L_{ins}(object_i)$)
06 cellOfHBase(determinstic attribute columns) = $T_{dcol}(object_i)$
07 $L_{pos}(object_i)$ = getValues($L_{pos}(object_i)$)
08 cellOfHBase(fuzzy attribute columns) = $L_{pos}(object_i)$
09 end for
10 end while
```

The mapping frameworks operate in two phases. In the first phase (lines 2-3 of Algorithm 1), the corresponding HBase schema is generated. In the second phase (lines 4-9), the contents in fuzzy object-oriented biomedical objects are migrated to the fuzzy HBase cells based on the generated HBase schema. The values are migrated from the fuzzy object-oriented biomedical objects to the corresponding cells of the constructed HBase tables at lines 6 and 8 respectively.

Theorem 1: For each object i in fuzzy object-oriented databases O_i and its transformed object $\gamma(i)$ in fuzzy HBase databases $H(\gamma(O_i))$, two conditions holds: i) there is a mapping Δ from O_i to $H(\gamma(O_i))$, for each instance i in O_i, $\Delta(i)$ is an object of $H(\gamma(O_i))$, and ii) there is a mapping Γ from $H(\gamma(O_i))$ to O_i, for each object j in $H(\gamma(O_i))$, $\Gamma(j)$ is an object of O_i.

Proof: For the first part of Theorem 1, let $(C_{sup}(i), C_{sub}(i), L_{datt}(i), L_{uatt}(i), L_{ins}(i), L_{pos}(i))$ and $(T_{dcf}(\gamma(i)), T_{ucf}(\gamma(i)), T_{dcol}(\gamma(i)), T_{ucol}(\gamma(i)), T_{ins}(\gamma(i)), T_{pos}(\gamma(i)))$ be a fuzzy class i in O_i, and a fuzzy object $\gamma(i)$ in $H(\gamma(O_i))$ respectively.

- According to mapping rules introduced in Section 3.1, it is easy to prove that $\Delta(L_{datt}(i)) = T_{dcol}(\gamma(i))$, $\Delta(L_{uatt}(i)) = T_{ucol}(\gamma(i))$, $\Delta(L_{ins}(i)) = T_{ins}(\gamma(i))$, and $\Delta(L_{pos}(i)) = T_{pos}(\gamma(i))$.
- For the fuzzy inheritance, if C_k *inherit* C *cover/partition with* (C_a, C_m), we have $C_{sup}(i) = C$, $C_{sub}(i) = C_a \cup C_k \cup C_m$. Based on the mapping rules introduced in Section 3.2, i) if C is the deterministic superclass of class (C_a, C_k, C_m) in O_i, and C_a, C_k, C_m are fuzzy subclasses, then $\Delta(C) = T_{dcf}(\gamma(i))$, $\Delta(C_a \cup C_k \cup C_m) = T_{ucf}(\gamma(i))$, ii) if C is the deterministic superclass of class (C_a, C_k, C_m) in O_i, and C_a is a deterministic subclass, C_k and C_m are fuzzy subclasses, then $\Delta(C \cup C_a) = T_{dcf}(\gamma(i))$, $\Delta(C_k \cup C_m) = T_{ucf}(\gamma(i))$, iii) i) if C is the deterministic superclass of class (C_a, C_k, C_m) in O_i, and C_a, C_k, C_m are deterministic subclasses, then $\Delta(C \cup C_a \cup C_k \cup C_m) = T_{dcf}(\gamma(i))$, iv) if C is the fuzzy superclass of class (C_a, C_k, C_m) in O_i, and C_a, C_k, C_m are fuzzy subclasses, then $\Delta(C \cup C_a \cup C_k \cup C_m) = T_{ucf}(\gamma(i))$, v) if C is the fuzzy superclass of class (C_a, C_k, C_m) in O_i, and C_a is a deterministic subclass, C_k and C_m are fuzzy subclasses, then $\Delta(C_a) = T_{dcf}(\gamma(i))$, $\Delta(C \cup C_k \cup C_m) = T_{ucf}(\gamma(i))$; vi) if C is the fuzzy superclass of class (C_a, C_k, C_m) in O_i, and C_a, C_k and C_m are deterministic subclasses, then $\Delta(C) = T_{ucf}(\gamma(i))$, $\Delta(C_a \cup C_k \cup C_m) = T_{dcf}(\gamma(i))$.

As a consequence of the above, for each instance i in O_i, $\Delta(i)$ is an object of $H(\gamma(O_i))$. Similarly, the second part of Theorem 1 is a mutually inverse process of the first one, and could be proved in the same manner, i.e., for each object j in $H(\gamma(O_i))$, $\Gamma(j)$ is an object of O_i.

IV. EXPERIMENTAL EVALUATION

A. EXPERIMENTAL SETUP

In this section, the performance comparisons after the reengineering are discussed based on our experiments in the

TABLE 2. Queries used for the experiments.

Type1-Q1	To retrieve results grouping by an attribute, and containing this attribute and the aggregate sum combination of values from the test datasets.
Type2-Q2	To retrieve results grouping by three attributes, and containing these three attributes and the aggregate sum combination of values from the test datasets.
Type3-Q3	To retrieve results grouping by two attributes, and containing these two attributes and the aggregate sum combination of values from the test datasets.

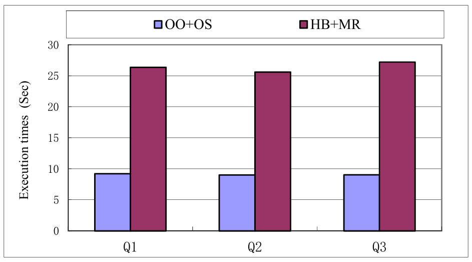
database applications for the queries of massive data, from the perspectives of query time and scalability. For the data in HBase databases, the MapReduce query model is used to evaluate the performance, denoted as HB+MR. For the object-oriented data, the previous OQL (object query language) query model [54]–[56] implemented in SQL-like query specifications is used to evaluate the performance, denoted as OO+OS. In order to simulate a real scenario of massive object-oriented data and offer a comprehensive evaluation of HB+MR and previous approach OO+OS in terms of query time and scalability, we performed our experiments on synthetic datasets which contain a randomly generated fuzzy class and attributes. For the randomly generated class, we randomly generate 6 attributes (numeric data are generated by using uniformly distributed parameters). We then ran the experiments on the different synthetic datasets, depicted D1, D2, D3, D4 and D5, which contains different amount of data records varying from 1, 5, 10, 50 and 100 million. We chose three representative queries listed in Table 2 to evaluate the approaches.

All the experiments are running on a 7-server Hadoop 2.7.1 cluster, including 1 master node, 6 slave nodes and 74TB storages. Each server is equipped with Xeon E5 2680 V4 2.40GHz CPU *2 and 64GB main memory. The approaches are programmed in Java with JDK 1.8.0 and CentOS 6.3 as operating system.

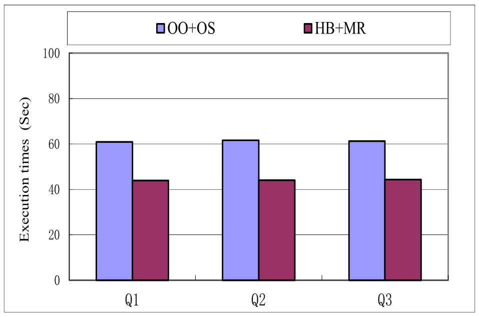
B. EXPERIMENTAL RESULTS

In this section, we show the performance of the proposed approach in terms of query time and scalability metrics by using the following group of experiments.

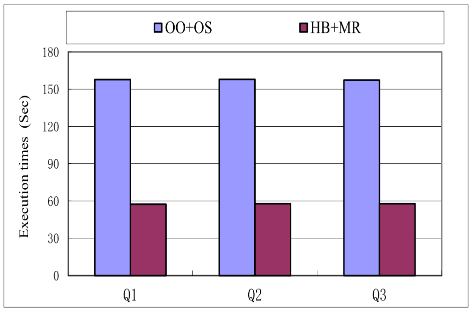
The aim of the first group of experiments is to analyze the query performance on different datasets with the same number of slave nodes. Figure 3 shows the execution times of queries for HB+MR and OO+OS, tested on D1, D2, D3, D4 and D5 with 3 slave nodes. Our first observation is that query times of using OO+OS is less than those of using HB+MR when executing queries on a small size of datasets. For example, in Figure 3(a), HB+MR is about 2 times slower than OO+OS when processing Q2 on D1. The second observation is that query times of using OO+OS is more than those of using HB+MR when executing queries on a large size



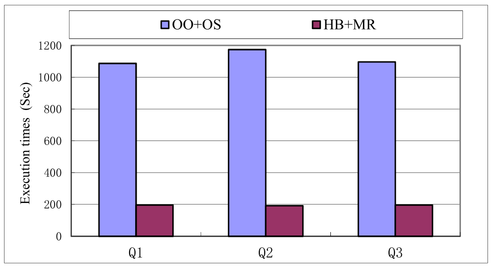
(a) query times on D1



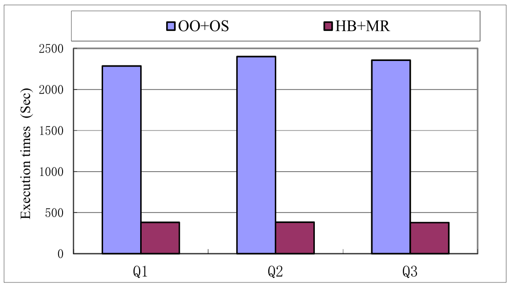
(b) query times on D2



(c) query times on D3



(d) query times on D4



(e) query times on D5

FIGURE 3. Execution time comparisons running on 3 slave nodes.

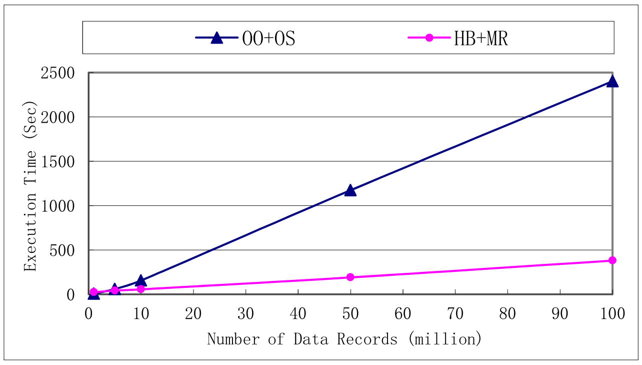


FIGURE 4. Varying the number of data records executing Q2 on 3 slave nodes.

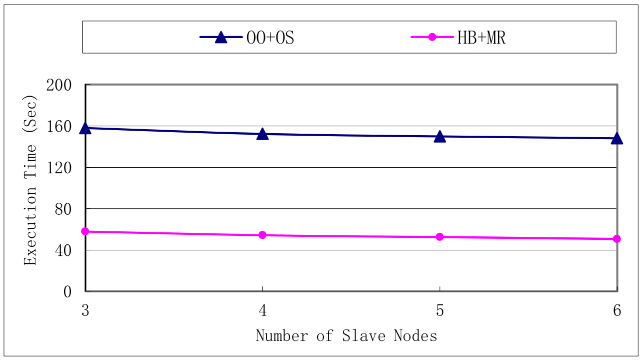


FIGURE 5. Varying the number of slave nodes executing Q2 on D3.

of datasets. For instance, in Figure 3(c), HB+MR is about 2 times faster than OO+OS when processing Q2 on D3.

In the second group of experiments, the scalability of HB+MR and OO+OS is investigated. Figure 4 shows the results by varying the number of data records when executing Q2 on 3 slave nodes. From Figure 4, we can see that, query times increase linearly with the increase of the number of records. In particular, the times consumptions for OO+OS are much higher than those for HB+MR. We also observe that larger size of datasets results in greater gaps of query times between HB+MR and OO+OS, e.g., in Figure 3(e), HB+MR is about 5 times faster than OO+OS when processing Q2 on D5. These results conforms to our previous observations that, HB+MR has the ability of handling the large-scale datasets and better performance when executing queries on a large size of datasets.

Figure 5 reports the results by varying the number of slave node parameters executing Q2 on D3. From Figure 5, we see that, query times of HB+MR and OO+OS decrease with increasing the number of slave nodes, and HB+MR still holds the performance advantage over OO+OS. These results further illustrate that HB+MR has good scalability.

V. CONCLUSION

In this paper, we investigated the uncertainty modeling of object-oriented biomedical information in HBase. After introducing the formalizations of fuzzy object-oriented and HBase models, we develop a set of mapping rules to handle the schema transformation. Meanwhile, we present a formal approach to transform fuzzy object-oriented algebra into fuzzy HBase algebra, and illustrate the transformation approaches by using representative examples. Experimental results indicate that using HBase databases could provide great support for massive fuzzy object-oriented biomedical data with both nice efficiency and scalability.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and suggestions. *(Lei Zhang and Jialiang Sun are co-first authors.)*

REFERENCES

[1] W.-S. Cho, K.-H. Hong, and W.-K. Loh, "Estimating nested selectivity in object-oriented and object-relational databases," *Inf. Softw. Technol.*, vol. 49, no. 7, pp. 806–816, Jul. 2007.
[2] Q. Yan, "Bioinformatics for transporter pharmacogenomics and systems biology: Data integration and modeling with UML," in *Membrane Transporters in Drug Discovery and Development*. Cham, Switzerland: Springer, 2010, pp. 23–45.
[3] M. Hakman and T. Groth, "Object-oriented biomedical system modelling—The language," *Comput. Methods Programs Biomed.*, vol. 60, no. 3, pp. 153–181, Nov. 1999.
[4] M. A. Dietz, A. O. Grant, and C. F. Starmer, "An object oriented user interface for analysis of biological data," *Comput. Biomed. Res.*, vol. 23, no. 1, pp. 82–96, Feb. 1990.
[5] S. Ball, L. Wright, and P. Miller, "SENEX: An object-oriented biomedical knowledge base," in *Proc. Annu. Symp. Comput. Appl. Med. Care, Amer. Med. Informat. Assoc.*, 1989, pp. 85–89.
[6] M. L. Hines, "Conceptual object-oriented database: A theoretical model," *Inf. Sci.*, vol. 105, nos. 1–4, pp. 31–68, Mar. 1998.
[7] I. Dzafic, J. Sofo, E. Halilovic, N. Lecek, and M. Music, "Object-oriented database and user interface design," in *Proc. Eurocon*, Jul. 2013, pp. 558–563.
[8] V. Saxena and A. Pratap, "Performance comparison between relational and object-oriented databases," *Int. J. Comput. Appl.*, vol. 71, no. 22, pp. 1–4, 2013.
[9] P. Suri and M. Sharma, "A comparative study between the performance of relational & object oriented database in Data Warehousing," *Int. J. Database Manage. Syst.*, vol. 3, no. 2, p. 116, 2011.
[10] S. Wang, "Object identity set algebra for object-oriented database systems," in *Proc. 5th IEEE Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Dec. 2012, pp. 1–6.
[11] J. A. O'Rawe, S. Ferson, and G. J. Lyon, "Accounting for uncertainty in DNA sequencing data," *Trends Genet.*, vol. 31, no. 2, pp. 61–66, Feb. 2015.
[12] S. P. Hey and A. S. Kesselheim, "Countering imprecision in precision medicine," *Science*, vol. 353, no. 6298, pp. 448–449, Jul. 2016.
[13] J. P. Mackay, M. J. Landsberg, A. E. Whitten, and C. S. Bond, "Whaddaya know: A guide to uncertainty and subjectivity in structural biology," *Trends Biochem. Sci.*, vol. 42, no. 2, pp. 155–167, Feb. 2017.
[14] A. G. Kennedy, "Managing uncertainty in diagnostic practice," *J. Eval. Clin. Pract.*, vol. 23, no. 5, pp. 959–963, Oct. 2017.
[15] R. Cavallo and M. Pittarelli, "The theory of probabilistic databases," *VLDB*, vol. 87, pp. 1–4, Sep. 1987.
[16] D. Dubois and H. Prade, "Possibility theory and its applications: Where do we stand?" in *Springer Handbook of Computational Intelligence*. Cham, Switzerland: Springer, 2015, pp. 31–60.
[17] J. Liu and D. L. Yan, "Answering approximate queries over XML data," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 2, pp. 288–305, Apr. 2016.
[18] S. Dutta, L. Sahoo, and D. Dwibedy, "An equivalence relation to reduce data redundancy based on fuzzy object oriented database system," *Int. J. Soft Comput. Eng.*, vol. 2, no. 3, pp. 417–421, 2013.
[19] B. Fahimian and A. Harounabadi, "A structure to support queries in object-oriented database using fuzzy XML tags," *Manage. Sci. Lett.*, vol. 4, no. 11, pp. 2379–2386, 2014.

[20] M. Pourbehzadi, A. Haroonabadi, and M. Sadeghzadeh, "A new weighted fuzzy grammar on object oriented database queries," *Manage. Sci. Lett.*, vol. 2, no. 4, pp. 1133–1140, Jul. 2012.

[21] S. Sonia and S. Kumari, "Fuzzy object oriented database versus FRDB for uncertainty management," *Int. J. Comput. Appl.*, vol. 74, no. 17, pp. 22–27, 2013.

[22] L. Yan, Z. Ma, F. Zhang, and Z. Ma, *Fuzzy XML Data Management*. Cham, Switzerland: Springer, 2014.

[23] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.

[24] R. Delhibabu and A. Behrend, "A new rational algorithm for view updating in relational databases," *Int. J. Speech Technol.*, vol. 42, no. 3, pp. 466–480, Apr. 2015.

[25] C. Liu, M. W. Vincent, and J. Liu, "Constraint preserving transformation from relational schema to XML schema," *World Wide Web*, vol. 9, no. 1, pp. 93–110, Mar. 2006.

[26] U. Yun and H. Ryang, "Incremental high utility pattern mining with static and dynamic databases," *Int. J. Speech Technol.*, vol. 42, no. 2, pp. 323–352, Mar. 2015.

[27] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008.

[28] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, p. 205, Oct. 2007.

[29] X. Gao, V. Nachankar, and J. Qiu, "Experimenting lucene index on HBase in an HPC environment," in *Proc. 1st Annu. Workshop High Perform. Comput. Meets Databases (HPCDB)*, 2011, pp. 25–28.

[30] I. Konstantinou, E. Angelou, C. Boumpouka, D. Tsoumakos, and N. Koziris, "On the elasticity of NoSQL databases over cloud management platforms," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2011, pp. 2385–2388.

[31] T. Rabl, M. Sadoghi, H.-A. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankowskii, "Solving big data challenges for enterprise application performance management," 2012, *arXiv:1208.4167*. [Online]. Available: http://arxiv.org/abs/1208.4167

[32] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Netw. Appl.*, vol. 2019, pp. 1–11, Apr. 2019.

[33] H. Gao, Y. Duan, L. Shao, and X. Sun, "Transformation-based processing of typed resources for multimedia sources in the IoT environment," *Wireless Netw.*, vol. 2019, pp. 1–17, Nov. 2019.

[34] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet Things J.*, to be published.

[35] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[36] Y. Jin, X. Guo, Y. Li, J. Xing, and H. Tian, "Towards stabilizing facial landmark detection and tracking via hierarchical filtering: A new method," *J. Franklin Inst.*, to be published.

[37] J. Li, X. Zhang, Z. Wang, X. Chen, J. Chen, Y. Li, and A. Zhang, "Dual-band eight-antenna array design for MIMO applications in 5G mobile terminals," *IEEE Access*, vol. 7, pp. 71636–71644, 2019.

[38] J. Liu and Z. M. Ma, "Formal transformation from fuzzy object-oriented databases to fuzzy XML," *Int. J. Speech Technol.*, vol. 39, no. 3, pp. 630–641, Oct. 2013.

[39] T. N. Jarada, K. Chung, A. Shimoon, P. Karampelas, R. Alhajj, and J. Rokne, "Mapping rules for converting from ODL to XML schemas," in *Proc. 12th Int. Conf. Inf. Integr. Web-Based Appl. Services (iiWAS)*, 2010, pp. 307–316.

[40] C.-C. Huang and W. Liou, "A study on the translation mechanism from relational-based database to column-based database," in *Proc. Int. Conf. Inform. Appl. (ICIA)*, 2012, pp. 480–486.

[41] B. Chattopadhyay, L. Lin, W. Liu, S. Mittal, P. Aragonda, V. Lychagina, Y. Kwon, and M. Wong, "Tenzing a SQL implementation on the mapreduce framework," Google, Menlo Park, CA, USA, Tech. Rep., 2011.

[42] R. Vilaça, F. Cruz, J. Pereira, and R. Oliveira, "An effective scalable SQL engine for NoSQL databases," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.* Cham, Switzerland: Springer, 2013, pp. 155–168.

[43] J. Liu, Q. Liu, L. Zhang, S. Su, and Y. Liu, "Enabling massive XML-based biological data management in HBase," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, to be published.

[44] J. Liu, Z. Qu, M. Yang, J. Sun, S. Su, and L. Zhang, "Jointly integrating VCF-based variants and OWL-based biomedical ontologies in MongoDB," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, to be published.

[45] J. Fong, "Translating object-oriented database transactions into relational transactions," *Inf. Softw. Technol.*, vol. 44, no. 1, pp. 41–51, Jan. 2002.

[46] J. Liu, Z. M. Ma, and X. Feng, "Formal approach for reengineering fuzzy XML in fuzzy object-oriented databases," *Int. J. Speech Technol.*, vol. 38, no. 4, pp. 541–552, Jun. 2013.

[47] T. Naser, K. Kianmehr, R. Alhajj, and M. J. Ridley, "Transforming object-oriented databases into XML," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2007, pp. 600–605.

[48] Z. M. Ma, L. Yan, and F. Zhang, "Modeling fuzzy information in UML class diagrams and object-oriented database models," *Fuzzy Sets Syst.*, vol. 186, no. 1, pp. 26–46, Jan. 2012.

[49] Y. Kornatzky and P. Shoval, "Conceptual design of object-oriented database schemas using the binary-relationship model," *Data Knowl. Eng.*, vol. 14, no. 3, pp. 265–288, Feb. 1995.

[50] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[51] M. Nalin Vora, "Hadoop-HBase for large-scale data," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, vol. 1, Dec. 2011, pp. 601–605.

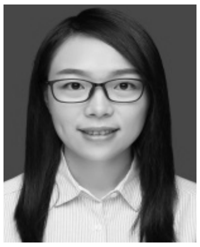
[52] L. Wang, B. Chen, and Y. Liu, "Distributed storage and index of vector spatial data based on HBase," in *Proc. 21st Int. Conf. Geoinform.*, Jun. 2013, pp. 1–5.

[53] J. Liu and X. X. Zhang, "Modeling fuzzy relational database in HBase," *J. Intell. Fuzzy Syst.*, vol. 31, no. 3, pp. 1845–1857, Aug. 2016.

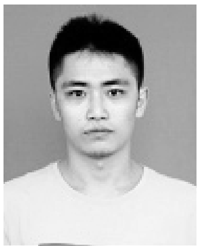
[54] S. Dhande and G. Bamnote, "Improving query processing performance using optimization techniques for object-oriented DBMS," in *Proc. 2nd Int. Conf. Comput. Commun. Technol.* Cham, Switzerland: Springer, 2016, pp. 119–127.

[55] S. Dhande and G. R. Bamnote, "Query optimization in object oriented DBMS: Direct navigation," in *Proc. Int. Conf. Comput. Commun. Control Autom.*, Feb. 2015, pp. 412–416.

[56] T. Niemi, M. Christensen, and K. Järvelin, "Query language approach based on the deductive object-oriented database paradigm," *Inf. Softw. Technol.*, vol. 42, no. 11, pp. 777–792, Aug. 2000.



LEI ZHANG received the Ph.D. degree in microbiology from the College of Life Sciences, Zhejiang University, China. She is currently an Associate Professor with the Zhejiang University of Science and Technology, China. Her current research interests include biological data modeling and bioinformatics.



JIALIANG SUN received the M.S. degree in computer application technology from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His current research interests include genome data analysis and biological database modeling.

SHUHUI SU received the M.S. degree in computer application technology from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. Her current research interests include genome data analysis, biological databases, and ontologies.

QIURU LIU received the M.S. degree in computer application technology from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His current research interests include massive biological database management and genome data analysis.

JIAN LIU received the Ph.D. degree in computer application technology from Northeastern University, China. He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His current research interests include massive biological database management, multiomics data analysis, and bioinformatics.

• • •