Fuzzy request handler for Mongo QL derived from SQL

I F Astachova1, N K Samoilov2 and E I Kiseleva3
1Department of Applied Mathematics, Informatics and Mechanics, Voronezh State University, 1, University Square, Voronezh, 394018, Russia
2Department of Programming and Information Technologies, Voronezh State University, Voronezh, Russia
3Department of Pedagogy and Methods of Preschool and Primary Education, Psychological and Pedagogical Faculty, Voronezh State Pedagogical University, 86, Lenin street, Voronezh, Russia
E-mail: astachova@list.ru

Abstract. Relational DBMS are often used to store fuzzy values, but problems arise with putting such data in a tabular form. Moreover, there appears a problem of storing both the crisp and fuzzy data related to one subject domain in one column of a relational table. This article considers the mechanism of storing crisp and fuzzy values and linguistic variables in the document-oriented Mongo DBMS. The data are stored in the collection as GeoJSON geometry; different geometries are used for different data options. The possibility of storing crisp scalar values, crisp value sets, crisp value intervals and fuzzy values in the collection documents is described. For data processing by means of SQL queries, the context-free grammar of the SQL subset is described, according to which lexer and parser are generated. In order to form the structure of an abstract syntactic tree, a corresponding object model has been implemented. A translator application has been developed, which allows converting SQL queries related to the crisp and fuzzy data into Mongo QL queries. The algorithm of fuzzy queries translation process is suggested; the geometrical interpretation of data comparison operations is described. The examples show the options of fuzzy comparison operations for different value options.

1. Introduction
The theory of fuzzy sets is widely used in artificial intelligence systems [1–5]. The reliability of the data depends on the amount of information stored and processed in the system obtained using fuzzy analysis. Currently, object-relational databases are used for this purpose [6–10]. However, both the relational and the object data models do not adequately meet the needs of storing fuzzy and mixed information. Most databases lack in-built fuzzy data types and mechanisms for working with them.

Database for fuzzy data must meet such requirements as support for fuzzy data types to ensure their integrity. Database for fuzzy sets must perform basic operations working with various t-norms, t-co norms and implications, it must obtain results of defuzzification, the search of the membership function for predefined linguistic values [11, 12].

The two most common approaches to storing and processing fuzzy data are relational design and the use of fields with fuzzy data types [13–15].
Works [16–18] present extensions of relational algebra and SQL language with fuzzy operators of comparison and aggregation, and fuzzy logic operators for relational DBMS. And fuzziness is imported into the crisp data by adding to the table a column with the values of the membership function, or the values of the membership function are stored in a separate table and are associated with the original records by a foreign key. Such an implementation involves a problem of storing mixed, or crisp and fuzzy data, in one table column. Moreover, in these works the authors use a strictly limited set of membership function types: triangular, trapezoidal, Z-shaped and S-shaped. The membership functions are specified in a tabular or an analytical manner. This paper suggests using document-oriented NoSQL DBMS – Mongo for storing mixed data. A Mongo database is a set of named collections (analog of a table in RDBMS); the collection stores data in the form of a set of JSON documents (an analog of RDBMS entry) [19, 20].

In this article, it is necessary to do the following:
1. Develop a mixed data storage mechanism using NOSQL of Mongo DBMS.
2. Implement a crisp and fuzzy query translator from SQL language into Mongo QL language.

![Image](image.png)

**Figure 1.** Representation of a linguistic variable in FuzzyVariable collection

2. Materials and methods
The data that describe the linguistic variable [21] in Mongo DBMS are stored in a FuzzyVariable collection dictionary, that is in the typeUID field is stored the linguistic variable identifier, and in the geometry field is stored geometry, in GeoJSON format, that describes the membership function as a set of broken lines by the points with the coordinates \((x, \mu(x))\). Each broken line corresponds to one of the linguistic variable values. MultiLineString type is used to store a collection of broken lines in GeoJSON format (figure 1).

FuzzyValue collection dictionary is used to store the linguistic variable values separately, in the form of a broken line. The FuzzyValue collection is similar to the FuzzyVariable collection, but into FuzzyValue a valueUID field is added – a fuzzy value identifier; and in the field geometry is stored the geometry that describes the membership function as one broken line in GeoJSON format. Separate storing of the linguistic variable values, in a separate collection, allows accelerating the search for the geometries that describe the membership function during some comparison operations.

Since mixed values can be stored in a Mongo database as GeoJSON geometry, the ordinary collections can store the following value options:
- **Point** – for a crisp value;
- **MultiPoint** – for a set of crisp values;
• Segment (LineString) – for an interval of crisp values;
• Broken line (LineString) – for a fuzzy value;
• Set of broken lines (MultiLineString) – for a fuzzy value.

It is convenient to use SQL language for working with data, but MongoDB uses its own Mongo QL query language [19], whose syntax is significantly different from SQL syntax, so one has to apply a translator [22] from SQL language (or its subset) into Mongo QL. In order to parse a subset of SQL language sentences, LL grammar is proposed (figure 2), which is described by means of ANTLR [23] and contains the following rules:

Blank character: space, tabulation, carriage shift:
WS: \[ \t\r\n]+ − > skip;
Number: integer, floating-point number:
DIGIT: [0-9];
INT_NUMBER: DIGIT+;
FLOAT_NUMBER: (INT_NUMBER \.), INT_NUMBER);
String:
STRING: 'N'? \( \sim \) (\( \sim \) | \( \sim \)) * \( \sim \);
Letter:
LETTER: [a-zA-Z];
Identifier:
ID: LETTER (LETTER | DIGIT)*;
Left parenthesis:
LPAREN : '(';
Right parenthesis:
RPAREN: ')';
Comparison operation:
crispCompareOperation: GT | LT | EQ | NEQ | GTE | LTE;
  GT: '>'; LT: '<'; EQ: '='; NEQ: '!=';
  GTE: '>='; LTE: '<=';
Fuzzy comparison operation:
fuzzyCompareOperation: FGT | FLT | FEQ | FNEQ | FGTE | FLTE;
  FGT: '\sim '>';
  FLT: '\sim <';
  FEQ: '\sim =';
  FNEQ: '\sim !=';
  FGTE: '\sim >=';
  FLTE: '\sim <=';
Logical operations:
notOperation: NOT;
andOperation: AND;
orOperation: OR;
Operand: attributes, constant:
operand: attr | constOperand;
constOperand: INT_NUMBER | FLOAT_NUMBER | STRING;
attr: (collectionName'.')? ID;
collectionName: ID;
The expression may include both crisp and fuzzy operations:
expression: operand (crispCompareOperation | fuzzyCompareOperation) operand;

The predicate is representable by an expression, negation, conjunction, disjunction; the
operation’s priority may be defined by parentheses:
predicate:
expression
| notOp predicate
| predicate andOp predicate
| predicate orOp predicate
| LPAREN predicate RPAREN;

Phrase Where:
allAttrsClause: (collectionName’.’)?ALL_ATOMS;
The rules for the list of selected attributes:
allAttrsClause: (collectionName’.’)?ALL_ATOMS;
attrList: allAttrsClause | attr | attr*;
The root rule for Select query:
select: SELECT attrList FROM collectionName whereClause?;

Using this grammar and ANTLR utility, were generated: classes of lexical analyzer (lexer)
MongoGramLexer, parser MongoGramParser, template interface MongoGramVisitor and class-
template MongoGramBaseVisitor for processing of the parse tree. The MongoASTBuilder class
was implemented in order to form the structure of the AST tree. The MongoQueryResolver
class was developed for pre-processing of fuzzy queries to Mongo DBMS. The diagram of the
abstract syntactic tree classes is shown in figure 3.

In order to form the structure of the abstract syntactic tree [22], the following classes are
used:
– NotOpNode, AndOpNode, OrOpNode – represent logical operations Not, And, Or,
respectively;
– PredicateNode – base class for predicates;
– ExpressionPredicateNode – predicate node that contains one logical expression;
– NotOpPredicateNode – predicate node that contains negation of a logical expression;
Figure 3. The diagram of the abstract syntactic tree classes

- AndOpPredicateNode – predicate node that contains conjunction of logical expressions;
- OrOpPredicateNode – predicate node that contains disjunction of logical expressions;
- ParenthesesPredicateNode – predicate node that contains a predicate in parentheses;
- BaseExpressionNode – base class for logical expressions;
- CrispExpressionNode – logical expression node that contains one of the crisp comparison operations (\(=\), \(!=\), \(<\), \(\geq\), \(\leq\), \(\geq\));
- FuzzyExpressionNode – logical expression node that contains one of the fuzzy comparison operations (\(\sim=\), \(\sim! =\), \(\sim<\), \(\sim\geq\), \(\sim\leq\), \(\sim\geq\));
- ColectionNameNode – node that contains the name of the collection from which the data will be sampled;
- AttrListNode – node that contains the list of selected attributes or "*" for all attributes;
- SelectNode – root node of the query.

3. Fuzzy query translation process

Let the linguistic variable (LV) be defined as \(\{X, T(X), U, G, M\}\) [21], where \(X\) — the name of the LV, \(T(X)\) — term-set of the LV, \(U\) — the universal set, \(G\) — the syntactic rule that generates the names \(x\) of the values of the variable \(X\), \(M\) — the semantic rule that associates each fuzzy variable \(x\) with its sense \(M(x)\), where \(M(x)\) is the representation of the broken line, \(h = <\;a_1, a_2, \ldots, a_n, a_{n+1}\;>\) (1), where \(n \geq 2\), \(a_i \in A\) — the set of points on the plane. In this case, the LV can be represented in the form of a set of broken lines \(H = <\;h_1, \ldots, h_n\;>\) (2), where \(h_i\) — the broken line defined in (1). Let us consider the fuzzy-equal operation \(\mu = (\nu, h_{\text{const}}^\alpha)\) (fuzzy equals, feq), where \(\nu\) — the value obtained from the DB (crisp value, interval, fuzzy value), \(h_{\text{const}}^\alpha\) — the value of the fuzzy variable (fuzzy constant) represented in the form of a broken line limited by the \(\alpha\)-level:

- Let \(v\) be the crisp value of \(\nu_{\text{crisp}}\), then \(\mu = (\nu_{\text{crisp}}, h_{\text{const}}^\alpha) = \text{true}\) if \([(\nu_{\text{crisp}}, 0), (\nu_{\text{crisp}}, 1)] \cap h_{\text{const}}^\alpha \neq \emptyset\);
• Let $v$ be the interval of the values $(v_1, v_2)$, specified on the plane as $[(v_1, 1), (v_2, 1)]$, and $p^a_{\text{const}}$ be the polygon formed by the broken line $h^a_{\text{const}}$, then $\mu = (v, h^a_{\text{const}}) = \text{true}$, if $\text{len}(v \cap p^a_{\text{const}}) \geq \frac{\text{len}(v)}{2}$;

• Let $v$ be the fuzzy value on level $\alpha$, specified by the broken line $h^\alpha$, $p^\alpha$ be the polygon formed by the broken line $h^\alpha$, $\text{const}$ be the polygon formed by the broken line $p^\alpha_{\text{const}}$ $S(p^\alpha_{\text{const}})$ be the area of the polygon $p^\alpha_{\text{const}}$, then $\mu = (v, h^\alpha_{\text{const}}) = \text{true}$, if $S(p^\alpha \cap p^\alpha_{\text{const}}) \geq S(p^\alpha_{\text{const}})$. Let us consider the fuzzy-greater than operation $\mu = (v, h^\alpha_{\text{const}})$. Let $h^\alpha_{\text{const}} = \langle c_1, c_2 \rangle, \ldots, \langle c_n, c_n + 1 \rangle > (3)$, $n \geq 2$, $c_i \in A$ – the set of points on the plane, then $H^\alpha = \{h \in H | h \succ h^\alpha_{\text{const}}\}$. In this case, the operation $\mu = (v, H^\alpha)$ reduces to the operation $\mu = (v, h^\alpha_{\text{const}})$. Similarly, we can define the comparison operations: fuzzy-less than, fuzzy-greater than or equal, fuzzy-less than or equal.

Let us consider the fuzzy-not equal operation $\mu \neq (v, h^\alpha_{\text{const}})$. Let $h^\alpha_{\text{const}}$ be defined as (3), then $H^\alpha = \{h \in H | h \neq h^\alpha_{\text{const}}\}$. In this case, the operation $\mu \neq (v, h^\alpha_{\text{const}})$ reduces to the operation $\mu = (v, H^\alpha)$.

4. Steps of the fuzzy query translation process

1. Parsing the text of the SQL query; obtaining the AS tree.
2. Searching for the comparison operations and the fuzzy constants in the AS tree.
3. Transforming the string presentation of fuzzy constants into the geometric one suing the data from FuzzyValue and FuzzyVariable collections, depending on the comparison operation.
4. Forming a query to the Mongo DB using the transformed fuzzy values and calls of the stored functions.

An index algorithm is used for computing the overlay of the polygons [24].

Results of studies and their discussion

The examples of translation of crisp SQL queries into MongoQL queries

| SQL Query | MongoQL Query |
|-----------|---------------|
| select * from employee; where age > 25 | db.getCollection('employee').find({age: {$gt: 25}}) |
| select * from employee; where age > 25 and salary = 1000 | db.getCollection('employee').find({age: {$gt: 25}, salary: 1000}) |
| select * from employee; where salary ~='big' | db.getCollection('employee').find({'feq': {this.salary, bigGeoJSON}}) |

5. Examples of fuzzy query operation (in Mongo DB)

The simplest case of comparison for equality of the fuzzy constant ‘big’ with the crisp value $x_0$ from the collection ‘employee’, figure 4. The fuzzy constant is presented in the form of the broken line $h_{\text{big}}$ from (1), which is obtained from FuzzyValue collection and is contained in the variable bigGeoJSON as GeoJSON geometry. The crisp value from the collection is representable in the form of the segment $AB$, where $A = (x_0, 0)$, $B = (x_0, 1)$. Let the intersection point $C(x_r, y_r) = h_{\text{big}} \cap AB$; in this case, the document of the collection falls within the resultant sampling, if $y_r \geq \alpha$, where $\alpha$ — the value of the alpha-level.
The case of comparison for equality of the fuzzy constant 'big' with the fuzzy value $f_0$ from the collection 'employee' (figure 5). Represent the fuzzy constant in the form of the polygon $p_{big}$ formed by the intersection of the straight line of the alpha-level and the broken line $h_{big}$, which is obtained from FuzzyValue collection and is contained in the variable bigGeoJSON as GeoJSON geometry. The fuzzy value from the collection is also representable in the form of a polygon $p_0$, formed by the intersection of the straight line of the alpha-level and the fuzzy value itself, specified as the broken line $h_0$. Let the intersection region be the polygon $P = p_{big} \cap p_0$; in this case, the document of the collection falls within the resultant sampling, if $S_P - S_{P_0} \geq \frac{1}{2}$.

```sql
select * from employee
where salary ~='big'
db.getCollection('employee')
    .find({'feq(this.salary, bigGeoJSON)'}))
```

The case of comparison for greater than of the fuzzy constant 'small' with the fuzzy value $f_0$ from the collection 'employee' (figure 6). Transform the representation of the fuzzy constant "small" in the form of a polygon into the multi-polygon $M = \cup p_i$, where $p_i \succ p_{small}$, and the geometric representation is selected from FuzzyVariable collection. In this case, for execution of the comparison operation $\succ 'small'$, it is sufficient to execute the comparison operation $\sim M$. The fuzzy value from the collection is also representable in the form of the polygon $p_0$, formed by the intersection of the straight line of the alpha-level and the fuzzy value itself, specified as the broken line $h_0$. Let the intersection region be the multi-polygon $P = M \cap p_0$; in this case, the document of the collection falls within the resultant sampling, if $S_P - S_{P_0} \geq \frac{1}{2}$.

![Figure 4. Fuzzy-equal for a crisp value and a fuzzy constant](image1)

![Figure 5. Fuzzy-equal for a fuzzy value and a fuzzy constant](image2)


6. Conclusion
The present work introduces data structures for storing crisp and fuzzy data in document-oriented MongoDB, context-free LL(\*) grammar for translation of SQL queries into MongoQL language, whereby a translator is implemented, and an algorithm for forming fuzzy queries to MongoDB.

References
[1] Cuevas E, Gámez J and Avalos O 2020 Optimization Algorithm. In: Recent Metaheuristics Algorithms for Parameter Identification. Fuzzy Logic Based. Studies in Computational Intelligence vol 854 (Springer, Cham)
[2] Zile M 2020 Intelligent and Adaptive Control. In: Mahdavi Tabatabaei N., Kabalci E., Bizon N. (eds) Microgrid Architectures, Control and Protection Methods (Springer, Cham)
[3] Trujillo-Cabezas R and Verdegay J L 2020 Fuzzy Optimization and Reasoning Approaches. In: Integrating Soft Computing into Strategic Prospective Methods. Studies in Fuzziness and Soft Computing vol 387 (Springer, Cham)
[4] Oueltati H, Azzouma N B and Said L B 2019 A Fuzzy Logic Based Trust-ABAC Model for the Internet of Things. In: Barolli L., Takizawa M., Xhafa F., Enokido T. (eds) Advanced Information Networking and Applications. AINA 2019. Advances in Intelligent Systems and Computing vol 926 (Springer, Cham)
[5] Swapna Rekha H, Nayak J, Naik B and Pelusi D 2019 Soft Computing in Robotics: A Decade Perspective. In: Nayak J., Balas V., Favorskaya M., Choudhury B., Rao S., Naik B. (eds) Applications of Robotics in Industry Using Advanced Mechanisms. ARIAM 2019. Learning and Analytics in Intelligent Systems vol 5 (Springer, Cham)
[6] Cadenas J T, Marín N and Vila M A 2011 Fuzzy Domains with Adaptable Semantics in an Object-Relational DBMS. In: Christiansen H., De Tré G., Yazici A., Zadrozny S., Andreasen T., Larsen H.L. (eds) Flexible Query Answering Systems. FQAS 2011. Lecture Notes in Computer Science vol 7022 (Springer, Berlin, Heidelberg)
[7] Han S, Wang H, Gao H, Li J and Huang S 9932 Fuzzy Keywords Query. In: Li F., Shim K., Zheng K., Liu G. (eds) Web Technologies and Applications. APWeb 2016. Lecture Notes in Computer Science vol 5 (Springer, Cham)
[8] Randhawa S and Jain S 2020 Energy-Efficient Fuzzy-Logic-Based Data Aggregation in Wireless Sensor Networks. In: Tuba M., Aksashe S., Joshi A. (eds) Information and Communication Technology for Sustainable Development. Advances in Intelligent Systems and Computing vol 933 (Springer, Singapore)
[9] Ferraro M B and Giordani P 2018 International conference series on soft methods in probability and statistics 87–90
[10] Behrend A and other 2017 New Trends in Databases and Information Systems: Contributions from ADBIS 2017 Workshops. In: Kirikova M. et al. (eds) New Trends in Databases and Information Systems. ADBIS 2017. Communications in Computer and Information Science vol 767 (Springer, Singapore)

[11] Beliakov G, James S and Wu J Z 2020 Learning Fuzzy Measures. In: Discrete Fuzzy Measures. Studies in Fuzziness and Soft Computing vol 382 (Springer, Cham)

[12] Deedar M H and noz Hernández S M 2019 User-Friendly Interface for Introducing Fuzzy Criteria into Expressive Searches. In: Bi Y., Bhatia R., Kapoor S. (eds) Intelligent Systems and Applications. IntelliSys 2019. Advances in Intelligent Systems and Computing vol 1037 (Springer, Cham)

[13] Sumangala K and Sathappan S 2020 A Hybrid Ant–Fuzzy Approach for Data Clustering in a Distributed Environment. In: Haldorai A., Ramu A., Mohanram S., Onn C. (eds) EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing. EAI/Springer Innovations in Communication and Computing (Springer, Cham)

[14] Sabo C, Pop P C, lean H V and nciulescu D D 2016 An Innovative Approach to Manage Heterogeneous Information Using Relational Database Systems. In: Madureira A., Abrahum A., Gamboa D., Novais P. (eds) Intelligent Systems Design and Applications. ISDA 2016. Advances in Intelligent Systems and Computing vol 557 (Springer, Cham)

[15] Pokorný J 2018 Integration of Relational and NoSQL Databases. In: Nguyen N., Hoang D., Hong TP., Pham H., Truviski B. (eds) Intelligent Information and Database Systems. ACHDS 2018. Lecture Notes in Computer Science vol 10752 (Springer, Cham)

[16] Pivert O and Bosc P 2012 Preference Queries to Relational Databases (London: Imperial College Press)

[17] Pivert O and Bosc P 1995 IEEE T. Fuzzy Systems 3 1–17

[18] Urrutia A, Tineo L and Gonzalez C 2008 Handbook of Research on Fuzzy Information Processing in Databases (London: Imperial College Press)

[19] Copeland R 2013 MongoDB Applied Design Patterns (Sebastopol CA: O’Reilly Media)

[20] Hows D, Membrey P and Plugge E 2015 The definitive guide to MongoDB (Apress)

[21] Zadeh L 1975 Information Sciences 8 199–249

[22] Astaxova I F and Maliyov A A 2013 Vestnik Voronezhskogo gosuniversiteta. Sistemnyj analiz i informacionnye tehnologii 134–139

[23] Part T 2013 The Definitive ANTLR 4 Reference (Dallas: The Pragmatic Bookshelf)

[24] Tyukachyov N A 2009 Vestnik VGTU 141–144