
СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

УДК 004.066

ТРАНСЛЯЦИЯ ЗАПРОСОВ С ЯЗЫКА SQL В ЯЗЫК MONGO QL

Н. К. Самойлов

Воронежский государственный университет

Поступила в редакцию 25.06.2019 г.

Аннотация. Для хранения нечётких значений часто используются реляционные СУБД, но при этом возникают проблемы размещения таких данных в табличной форме. Кроме того, появляется проблема хранения как чётких, так и нечётких данных, относящихся к одной предметной области, в одном столбце реляционной таблицы. В данной статье рассматривается механизм хранения чётких и нечётких значений и лингвистических переменных в документо-ориентированной СУБД Mongo. Данные хранятся в коллекции как GeoJSON геометрия, для различных вариантов данных используются различные геометрии. Описана возможность хранения в документах коллекции чётких скалярных значений, наборов чётких значений, интервалов чётких значений и нечётких значений. Для обработки данных посредством запросов языка SQL описывается контекстно-свободная грамматика подмножества языка SQL, по которой генерируются лексический анализатор и синтаксический анализатор. Для формирования структуры абстрактного синтаксического дерева реализована соответствующая объектная модель. Разработано приложение транслятора, которое позволяет преобразовывать SQL-запросы по чётким и нечётким данным в запросы на языке Mongo QL. Предложен алгоритм процесса трансляции нечётких запросов, описана геометрическая интерпретация операций сравнения данных. В примерах приведены варианты операций нечёткого сравнения для различных вариантов значений.

Ключевые слова: транслятор, грамматика, нечёткое значение, лингвистическая переменная, геометрия, mongodb, линейная ломаная.

ВВЕДЕНИЕ

В работах [1–3] представлены расширения реляционной алгебры и языка SQL нечёткими операторами сравнения, агрегации, нечёткими логическими операторами для реляционных СУБД. При этом нечеткие данные формируются путём добавления к таблице столбца со значениями функции принадлежности либо значения функции принадлежности хранятся в отдельной таблице и связаны с исходными записями внешним ключом. В такой реализации существует проблема хранения смешанных данных в одном столбце таблицы.

Кроме того, в работах [1–3] авторы используют строго ограниченный набор видов функций принадлежности: треугольную, трапецевидную, Z-образную, S-образную. Функции принадлежности задаются таблично или аналитически.

В данной работе предлагается использовать для хранения смешанных данных документо-ориентированную NoSQL СУБД – Mongo. База данных Mongo представляет собой набор именованных коллекций (аналог таблицы в РСУБД), коллекция хранит данные в виде набора JSON-документов (аналог записи в РСУБД) [4–5].

В данной статье предложен механизм хранения смешанных данных используя NoSQL СУБД MongoDB и описана реализация транслятора четких и нечетких запросов из языка SQL в язык Mongo QL.

1. МАТЕРИАЛЫ И МЕТОДЫ

Данные, описывающие лингвистическую переменную [6] в СУБД Mongo, хранятся в коллекции-словаре FuzzyVariable. В поле typeUID хранится идентификатор лингвистической переменной, в поле geometry хранится геометрия, в формате GeoJSON. Геометрия описывает функцию принадлежности, как набор ломаных линий, по точкам в координатах $(x, \mu(x))$. Каждая ломаная линия соответствует одному из значений лингвистической переменной. Для хранения набора ломаных линий в формате GeoJSON используется тип MultiLineString.

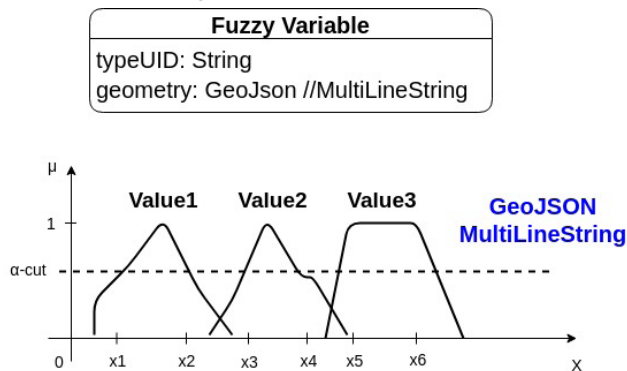


Рис. 1. Представление лингвистической переменной в коллекции FuzzyVariable

Для хранения значений лингвистической переменной по отдельности в виде ломаной линии используется коллекция-словарь FuzzyValue. Коллекция FuzzyValue подобна коллекции FuzzyVariable, но в FuzzyValue добавлено поле valueUID – идентификатор нечёткого значения и в поле geometry хранится геометрия, описывающая функцию принадлежности, как одна ломаная линия в формате GeoJSON. Хранение значений лингвистической переменной по отдельности в отдельной коллекции позволяет ускорить поиск геометрий описывающих функцию принадлежности при некоторых операциях сравнения.

Поскольку смешанные значения могут храниться в БД Mongo как GeoJSON-геометрия [7], то mongo-коллекция может содержать следующие варианты значений:

- точка (Point) – для чёткого значения;
- набор точек (MultiPoint) – для набора чётких значений;

- отрезок (LineString) – для интервала чётких значений;
- ломаная (LineString) – для нечёткого значения;
- набор ломаных (MultiLineString) – для нечёткого значения.

Для работы с данными удобно использовать язык SQL, но в MongoDB используется свой язык запросов – Mongo QL [4], синтаксис которого значительно отличается от синтаксиса SQL [8]. Преобразование подмножества предложений языка SQL в MongoQL-запросы осуществляется транслятором. Для генерации исходного кода транслятора используется LL-грамматика, описанная средствами ANTLR [9] и содержащая следующие правила:

пробельный символ: пробел, табуляция, перевод каретки:

```
WS: [ \t\r\n]+ -> skip;
```

число: целое число, число с плавающей точкой:

```
DIGIT: [0-9];
```

```
INT_NUMBER: DIGIT+;
```

```
FLOAT_NUMBER: (INT_NUMBER '.'
```

```
INT_NUMBER);
```

строка:

```
STRING: 'N'? '\\' (~'\\' | '\\\\' | '\\\'' ) * '\\\'';
```

литера:

```
LETTER: [a-zA-Z_];
```

идентификатор:

```
ID: LETTER (LETTER|DIGIT)*;
```

левая скобка:

```
LPAREN : '(';
```

правая скобка:

```
RPAREN: ')';
```

операции сравнения:

```
crispCompareOperation: GT | LT | EQ | NEQ | GTE | LTE;
```

```
GT: '>';
```

```
LT: '<';
```

```
EQ: '=';
```

```
NEQ: '!=';
```

```
GTE: '>=';
```

```
LTE: '<=';
```

операции нечёткого сравнения:

```
fuzzyCompareOperation: FGT | FLT | FEQ | FNEQ | FGTE | FLTE;
```

```

FGT:    '~>';
FLT:    '<~';
FEQ:    '~=';
FNEQ:   '~!=';
FGTE:   '~>=';
FLTE:   '<~=';
логические операции:
notOperation: NOT;
andOperation: AND;
orOperation: OR;
операнды (атрибут, константа):
operand: attr | constOperand;
constOperand: INT_NUMBER |
FLOAT_NUMBER | STRING;
attr: (collectionName'.')?ID;
collectionName: ID;
выражение (может включать в себя как
операцию чёткого сравнения, так и нечёткого):
expression: operand
(crispCompareOperation |
fuzzyCompareOperation) operand;
предикат (может быть представлен вы-
ражением, отрицанием, конъюнкцией, дизъ-
юнкцией, приоритет операций может опреде-
ляться скобками):
predicate:
    expression
    | notOp predicate
    | predicate andOp predicate
    
```

```

    | predicate orOp predicate
    | LPAREN predicate RPAREN;
фраза Where:
allAttrsClause:
(collectionName'.')?ALL_ATTRS;
правила для списка выбираемых атрибутов:
allAttrsClause:
(collectionName'.')?ALL_ATTRS;
attrList: allAttrsClause | attr
(', ' attr)*;
корневое правило для Select-запроса:
select: SELECT attrList FROM
collectionName whereClause?.
    
```

Из данной грамматики генератором нисходящих анализаторов ANTLR были построены классы лексического анализатора `MongoGramLexer`, синтаксического анализатора `MongoGramParser`, шаблонный интерфейс `MongoGramVisitor` и класс-шаблон `MongoGramBaseVisitor` для обработки дерева разбора. Для формирования структуры AST-дерева был реализован класс `MongoASTBuilder`. Разработан класс `MongoQueryResolver`, предназначенный для предварительной обработки нечётких запросов к СУБД Mongo. Диаграмма классов абстрактного синтаксического дерева представлена на рис. 3.

Для формирования структуры абстрактного синтаксического дерева [2] используются классы:

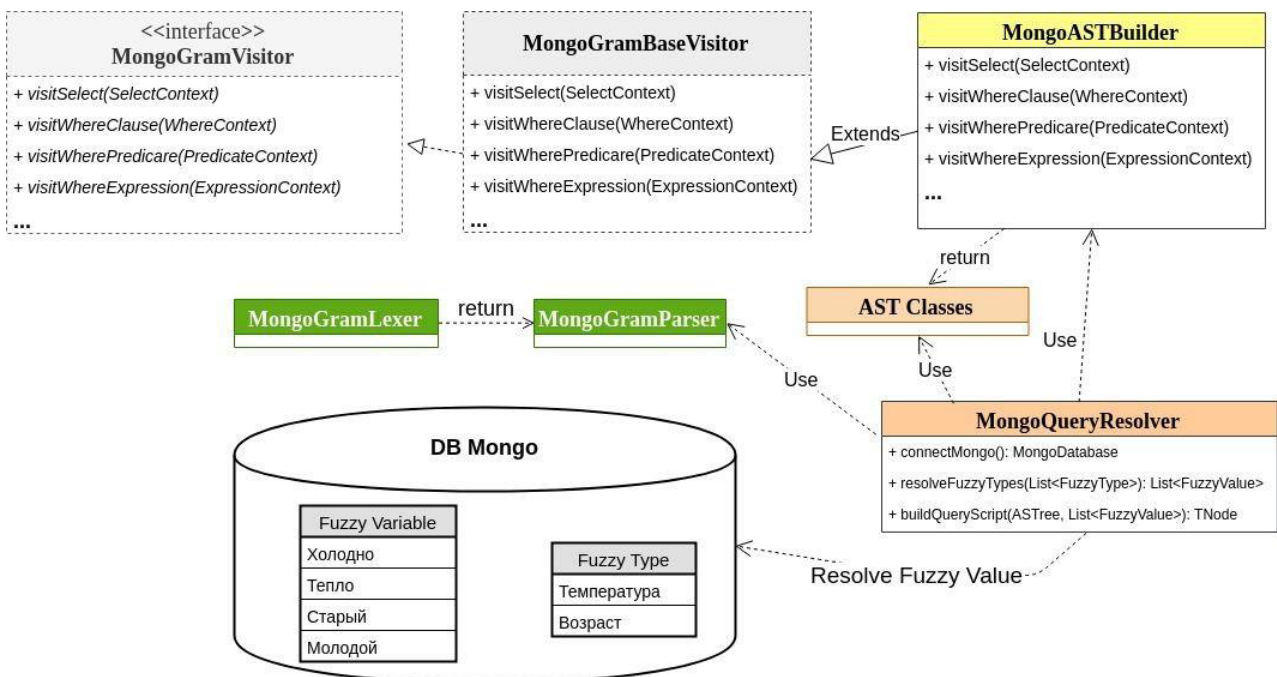


Рис. 2. Структура приложения транслятора

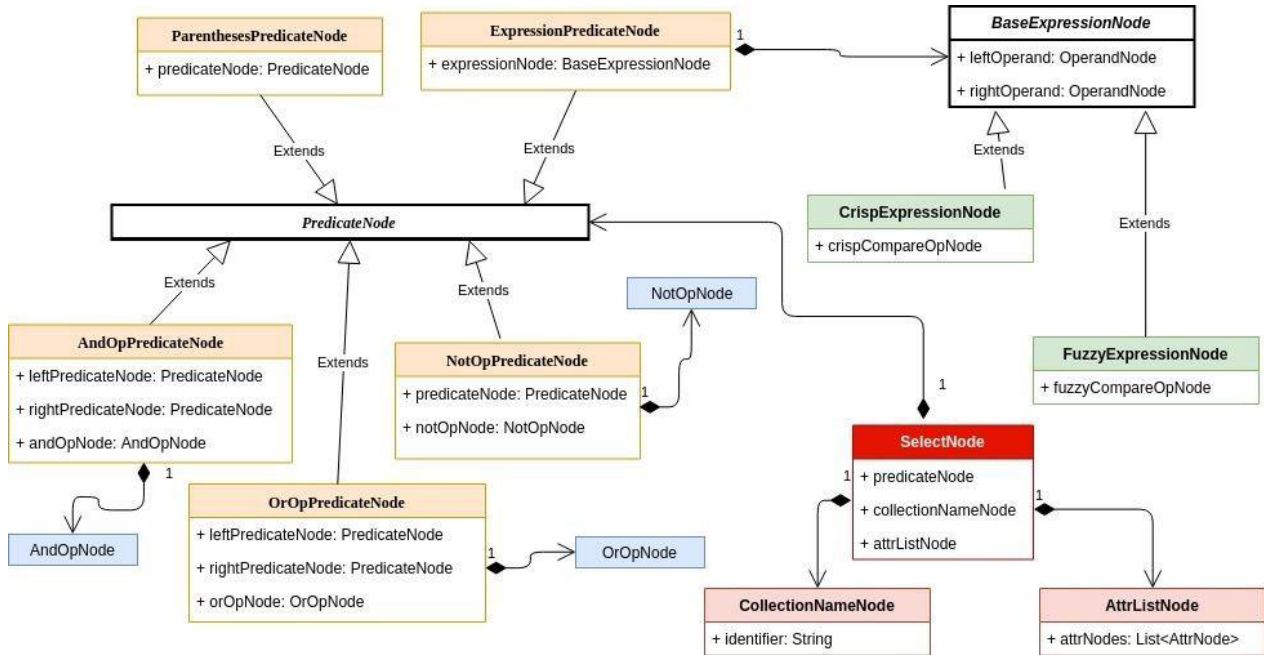


Рис. 3. Диаграмма классов абстрактного синтаксического дерева

- NotOpNode, AndOpNode, OrOpNode – представляют логические операции Not, And, Or, соответственно;
- PredicateNode – базовый класс для предикатов;
- ExpressionPredicateNode – узел предиката, который содержит одно логическое выражение;
- NotOpPredicateNode – узел предиката, который содержит отрицание логического выражения;
- AndOpPredicateNode – узел предиката, который содержит конъюнкцию логических выражений;
- OrOpPredicateNode – узел предиката, который содержит дизъюнкцию логических выражений;
- ParenthesesPredicateNode – узел предиката, который содержит предикат в скобках;
- BaseExpressionNode – базовый класс для логических выражений;
- CrispExpressionNode – узел логического выражения, который содержит одну из чётких операций сравнения (=, !=, <, >, <=, >=);
- FuzzyExpressionNode – узел логического выражения, который содержит одну из нечётких операций сравнения (~=, ~!=, ~<, ~>, ~<=, ~>=);
- CollectionNameNode – узел содержащий имя коллекции, из которой будет происходить выборка данных;

- AttrListNode – узел содержащий список выбираемых атрибутов или “*” для всех атрибутов;
- SelectNode – корневой узел запроса.

2. ОБРАБОТКА ГЕОМЕТРИЧЕСКИХ ДАННЫХ НЕЧЕТКИХ ЗАПРОСОВ

Пусть лингвистическая переменная (ЛП) определяется как $\{X, T(X), U, G, M\}$ [6], где X – название ЛП, $T(X)$ – терм-множество ЛП, U – универсальное множество, G – синтаксическое правило, порождающее названия для x – значений переменной X , M – семантическое правило, которое ставит в соответствие каждой нечёткой переменной x её смысл $M(x)$, где $M(x)$ – представление линейной ломаной $h = \langle [a_1, a_2], \dots, [a_n, a_{n+1}] \rangle (1)$, где $n \geq 2$, $a_i \in A$ – множество точек на плоскости. ЛП в таком случае представима в виде набора линейных ломаных $H = \langle h_1, \dots, h_n \rangle$, где h_i – ломаная, определенная в (1). Пусть h_{const}^α – значение нечёткой переменной (нечёткая константа) представленное в виде линейной ломаной из (1) ограниченной, α – уровнем [10], $h_{const}^\alpha = \langle [b_1, b_2], \dots, [b_n, b_{n+1}] \rangle$, где $n \geq 2$, $b_i \in A$, $b_i = (x_i, y_i)$, $y_i \geq \alpha$.

Рассмотрим операцию «нечётко-равно» $\mu_{=} (v, h_{const}^\alpha)$:

• пусть v – есть чёткое значение v_{crisp} , то $\mu_{=}(v_{crisp}, h_{const}^{\alpha}) = true$, если

$$\left[(v_{crisp}, 0), (v_{crisp}, 1) \right] \cap h_{const}^{\alpha} \neq \emptyset;$$

• пусть v – есть интервал значений (v_1, v_2) , заданный на плоскости как $\left[(v_1, 1), (v_2, 1) \right]$, p_{const}^{α} – полигон образованный ломаной h_{const}^{α} , то $\mu_{=}(v, h_{const}^{\alpha}) = true$, если

$$len(v \cap p_{const}^{\alpha}) \geq \frac{len(v)}{2};$$

• пусть v – есть нечёткое значение по уровню α , задаваемое ломаной h_v^{α} , p_v^{α} – полигон образованный ломаной h_v^{α} , p_{const}^{α} – полигон образованный ломаной h_{const}^{α} , $S(p_v^{\alpha})$ – площадь полигона p_v^{α} , то $\mu_{=}(v, h_{const}^{\alpha}) = true$, если

$$S(p_v^{\alpha} \cap p_{const}^{\alpha}) \geq \frac{S(p_v^{\alpha})}{2}.$$

Рассмотрим операцию «нечётко-больше» $\mu_{>}(v, h_{const}^{\alpha})$. Пусть $P_{>} = \{h \in H \mid h > h_{const}^{\alpha}\}$ – полигон образованный линейными ломаными нечётких значений следующих после h_{const}^{α} и прямой α – уровня, тогда операция «нечётко-больше» $\mu_{>}(v, h_{const}^{\alpha})$ сводится к операции «нечётко-равно» $\mu_{=}(v, P_{>})$. Аналогично, можно определить операции сравнения «нечётко-меньше», «нечётко-больше равно», «нечётко-меньше или равно».

Рассмотрим операцию «нечётко-неравно» $\mu_{\neq}(v, h_{const}^{\alpha})$. Пусть h_{const}^{α} определена как (3), тогда $P_{\neq} = \{h \in H \mid h \neq h_{const}^{\alpha}\}$. Таким образом, P_{\neq} – полигон образованный линейными ломаными нечётких значений неравных h_{const}^{α} и прямой α – уровня, тогда операция «нечётко-неравно» $\mu_{\neq}(v, h_{const}^{\alpha})$ сводится к операции «нечётко-равно» $\mu_{=}(v, P_{\neq})$.

Таким образом, трансляция нечёткого SQL-запроса в нечёткий MongoQL-запрос состоит из следующих шагов:

1. Разбор текста SQL-запроса, получение AS-дерева.

2. Поиск нечётких операций сравнения и нечётких констант в дереве.

3. Преобразование строчного представления нечётких констант в геометрическое, используя данные коллекций FuzzyValue или FuzzyVariable, в зависимости от операции сравнения.

4. Формирование запроса к БД Mongo используя преобразованные нечёткие значения и вызовы хранимых функций. Для обработки данных запроса используется операция map-reduce. В фазе reduce генерируется код для вычисления пересечения прямой и линейной ломаной или код для вычисления оверлея многоугольников [11–15].

3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ И ИХ ОБСУЖДЕНИЕ

3.1. Пример трансляции чётких SQL-запросов в MongoQL-запросы представлен в табл. 1

3.2. Простейший случай сравнения на равенство нечёткой константы 'big' с чётким значением x_0 из коллекции employee

Нечёткая константа представлена в виде линейной ломаной h_{big} из (1), которая получена из коллекции FuzzyValue и содержится в переменной bigGeoJSON как GeoJSON геометрия. Чёткое значение из коллекции представимо в виде отрезка AB , где $A = (x_0, 0)$, $B = (x_0, 1)$. Пусть точка пересечения $C(x_r, y_r) = h_{big} \cap AB$, в этом случае документ коллекции попадает в результирующую выборку, если $y_r \geq \alpha$, где α – значение альфа-уровня (рис. 4).

Таблица 1

select * from employee	db.getCollection('employee').find({})
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({\$and: [{age: {\$gt: 25}}, {salary: 1000}]})

<code>select * from employee where salary ~= 'big'</code>	<code>db.getCollection('employee').find({ 'feq(this.salary, bigGeoJSON)' })</code>
---	---

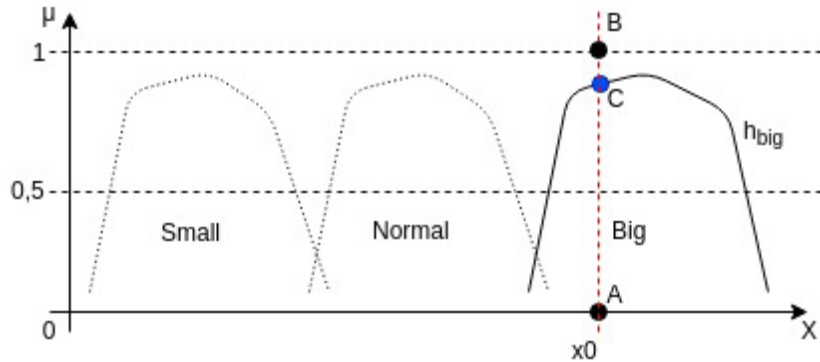


Рис. 4. Нечётко-равно для чёткого значения и нечёткой константы

3.3. Случай сравнения на равенство нечёткой константы 'big' с нечётким значением f_0 из коллекции employee

Представим нечёткую константу в виде многоугольника p_{big} образованного пересечением прямой альфа-уровня и линейной ломаной h_{big} , которая получена из коллекции FuzzyValue и содержится в переменной bigGeoJSON как GeoJSON геометрия. Нечёткое значение из коллекции также представимо в виде многоугольника p_0 образованного пересечением прямой альфа-уровня и самим нечётким значением заданным как линейная ломаная h_0 . Пусть область пересечения многоугольников $P = p_{big} \cap p_0$, в этом случае документ коллекции попадает в результирующую выборку, если $S_p - S_{p_0} \geq \frac{1}{2}$. (рис. 5).

3.4. Случай сравнения на больше нечёткой константы 'small' с нечётким значением f_0 из коллекции employee

Преобразуем представление нечёткой константы "small" в виде полигона в мультиполигона $M = \bigcup p_i$, где $p_i \succ p_{small}$, при этом геометрическое представление выбирается из коллекции FuzzyVariable. В этом случае для выполнения операции сравнения $\sim >$ "small" достаточно выполнить операцию сравнения $\sim = M$. Нечёткое значение из коллекции также представимо в виде многоугольника p_0 образованного пересечением прямой альфа-уровня и самим нечётким значением заданным как линейная ломаная h_0 . Пусть область пересечения – мультиполигон $P = M \cap p_0$, в этом случае документ коллекции попадает в результирующую выборку, если $S_p - S_{p_0} \geq \frac{1}{2}$ (рис. 6).

<code>select * from employee where salary ~= 'big'</code>	<code>db.getCollection('employee').find({ 'feq(this.salary, bigGeoJSON)' })</code>
---	---

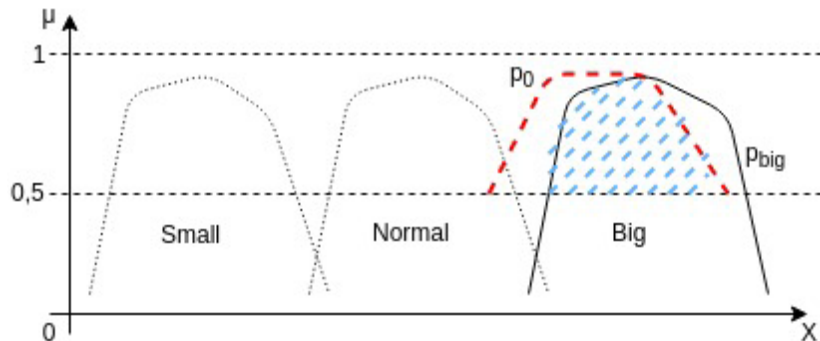


Рис. 5. Нечётко-равно для нечёткого значения и нечёткой константы

<pre>select * from employee where salary ~> 'small'</pre>	<pre>db.getCollection('employee').find({'feq(this.salary, gtSmallGeoJSON)' })</pre>
--	--

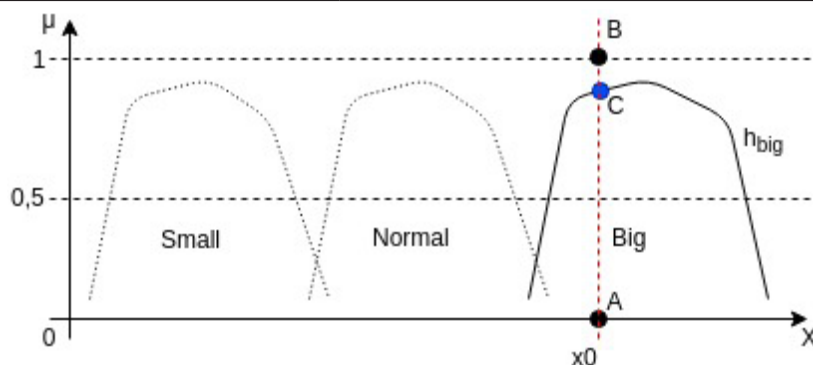


Рис. 6. Нечётко-больше для нечёткого значения и нечёткой константы

ЗАКЛЮЧЕНИЕ

В данной работе предложен способ хранения смешанных данных как геометрии в GeoJSON формате в коллекциях документо-ориентированной СУБД Mongo, описана контекстно-свободная LL-грамматика, на основе которой сгенерированы классы транслятора (лексический анализатор, синтаксический анализатор), классы абстрактного-синтаксического дерева, описан механизм трансляции и выполнения нечётких SQL-запросов в СУБД Mongo.

В дальнейшем планируется интегрировать транслятор нечетких запросов в ядро СУБД Mongo, добавить в Mongo поддержку индексов по нечетким данным и добавить в транслятор поддержку нечетких операций in, exists, нечеткой группировки и несимметричных операций над предикатами таких, как “P1 и возможно P2”, “P1 или иначе P2”.

СПИСОК ЛИТЕРАТУРЫ

1. Pivert, O. Fuzzy Preference Queries to Relational Databases / O. Pivert, P. Bosc – London : Imperial College Press, 2012. – 347 p.
2. Bosc, P. SQLf: a relational database language for fuzzy querying / Pivert O., Bosc P. // IEEE T. Fuzzy Systems. – 1995. – Vol. 3. – P. 1–17.
3. Urrutia, A. FSQL and SQLf: Towards a standard in fuzzy databases / A. Urrutia, L. Tineo, C. Gonzalez // Handbook of Research on Fuzzy Information Processing in Databases. – 2008. – Vol. 1. – P. 270–298.

4. Copeland, R. MongoDB Applied Design Patterns / R. Copeland – Sebastopol CA: O’Reilly Media, 2013. – 160 p.

5. Hows, D. The definitive guide to MongoDB / Hows D., Membrey P., Plugge E. – Apress, 2015. – 361 p.

6. Zadeh, L. The concept of a linguistic variable and its application to approximate reasoning / L. Zadeh // Information Sciences. – 1975. – Vol. 8. – P. 199–249.

7. The GeoJSON Format. – Режим доступа: <https://tools.ietf.org/html/rfc7946>. – (Дата обращения: 06.06.2019).

8. Толстобров, А. П. Управление данными : учебное пособие / А. П. Толстобров. – Воронеж : ИПЦ ВГУ, 2007. – 205 с.

9. Parr, T. The Definitive ANTLR 4 Reference / T. Parr – Dallas: The Pragmatic Bookshelf, 2013. – 305 p.

10. Системы искусственного интеллекта. Практический курс : Учеб. пособие / под ред. И. Ф. Астаховой. – М. : Бинوم. Лаборатория знаний, 2008. – 292 с.

11. Тюкачёв, Н. А. Алгоритм построения оверлея многоугольников и многогранников / Н. А. Тюкачёв // Вестник ВГТУ. – 2009. – № 6. – С. 141–144.

12. Margalit, A. An algorithm for computing the union, intersection or difference of two polygons / A. Margalit, G. D. Knott // Computers & Graphics. – 1989. – Vol. 13, No. 2. – P. 167–183.

13. O’Rourke, J. A new linear algorithm for intersection convex polygons / J. O’Rourke, C. B. Chien, T. Olson // Computer graphics and Image Processing. – 1982. – Vol. 19. – P. 384–391.

14. Скворцов, А. В. Построение объединения, пересечения, и разности произвольных многоугольников в среднем за линейное время с помощью триангуляции / А. В. Скворцов // Вычислительные методы и программирование. – 2002. – Т. 3. – С. 116–123.

15. Скворцов, А. В. Линейно-узловой алгоритм построения оверлеев двух полигонов / А. В. Скворцов // Вестник ТГУ. – 2002. – № 275. – С. 99–103.

Самойлов Николай Константинович – ассистент кафедры программирования и информационных технологий Воронежского государственного университета, e-mail: nk.samoylov@gmail.com

FUZZY QUERY TRANSLATION FROM SQL TO MONGO QL

N. K. Samoilov

Voronezh State University

Annotation. Relational DBMS are often used to store fuzzy values, but there are problems with placing such data in a tabular form. Besides, there appears the problem of storing both the clear and fuzzy data related to one subject area in one relational table column. This article considers the mechanism of storing clear and fuzzy values and linguistic variables in the documented Mongo DBMS. The data are stored in the collection as GeoJSON geometry, different geometries are used for different data variants. The possibility of storing clear scalar values, clear value sets, clear 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, which generates lexer and parser. To form the structure of the abstract syntactic tree the corresponding object model is implemented. A translator application has been developed which allows converting SQL queries based on clear and fuzzy data into Mongo QL queries. The algorithm of fuzzy queries translation process is offered, the geometrical interpretation of data comparison operations is described. The examples show the variants of fuzzy comparison operations for different variants of values.

Keywords: translator, grammar, fuzzy value, linguistic variable, geometry, mongodb, broken line.

Samoilov N. K. – assistant of the department Programming and Information Technologies Department, Voronezh State University, Voronezh, e-mail: nk.samoylov@gmail.com