https://doi.org/10.53656/math2022-4-5-pra

ПРАКТИЧЕСКИЙ МЕТОД РЕАЛИЗАЦИИ НЕЧЕТКИХ ЗАПРОСОВ ДЛЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Dr. Alexander Aleksandrovich Rybanov, Assoc. Prof.
Волжский политехнический институт – Волгоград (Россия)

Аннотация. Информационные системы, использующие базы данных, являются гибкими в той мере, в какой они позволяют пользователям запрашивать необходимые им данные. Язык SQL ограничен точной обработкой данных и не позволяет напрямую выразить нечеткие понятия естественного языка. Следовательно, придание SQL некоторой гибкости может помочь пользователям улучшить взаимодействие с информационными системами, не требуя от них изучения совершенно нового языка. Актуальной является задача снижения трудоемкости процесса интеграции механизмов нечетких запросов к уже действующим информационным системам. В статье показана ограниченность четких запросов, рассмотрены различные формы нечетких запросов. Проанализированы известные подходы к реализации нечетких запросов к четким реляционным базам данных. Представлен подробный анализ нечетких запросов, а также их преобразование в стандартные SQL-запросы с помощью MySQL. Предлагается метод реализации запросов к реляционным базам данных, объединяющий теорию нечетких множеств и SQL. Предлагаемый метод реализации возможности работы с нечеткими запросами основан на расширении четкой базы данных хранимыми функциями, без изменения структуры и состава ее таблиц. Преимуществами метода являются: повышение удобочитаемости и понимания SQL-запросов; простота интеграции с уже существующими базами данных информационных систем; гибкая настройка функции принадлежности лингвистических переменных в соответствии с потребностями пользователя базы данных. Применение метода показано на примере адаптации базы данных MySQL. Предложенный метод адаптации может быть широко использован для реализации нечетких запросов к базам данных различных реляционных СУБД, поддерживаюших работу с хранимыми функциями.

Ключевые слова: реляционные базы данных; нечеткая логика; нечеткие запросы; SQL; MySQL

Введение. Классический язык структурированных запросов SQL позволяет формулировать условия, которым должны соответствовать ин-

тересующие пользователя данные. Эти условия выбора должны быть заданы однозначно, т.к. точность является одним из основных требований при указании критериев отбора записей в SQL-запросах. Поскольку для человека основным средством коммуникации является естественный язык, это вызывает трудности при преобразовании нечетких и расплывчатых ограничений на поиск информации в язык SQL.

Например, если арендатор в поисках дешевой недвижимости точно укажет диапазон для цены, которая его устраивает, то, независимо от того, как заданы пределы этого диапазона, недвижимость, цена которой несущественно превышает установленный предел, не будет соответствовать условиям запроса. Причина этого в том, что рассматриваемые ограничения являются следствием необходимости точного определения условий, которые изначально были выражены неточными терминами на естественном языке.

Простой четкий запрос на зыке SQL можно представить следующим образом:

SELECT *атрибут*$_1$, …, *атрибут*$_n$
FROM *таблица*
WHERE *атрибут*$_p$ > *P* AND *атрибут*$_r$ < *R*;

Здесь значения *P* и *R* ограничивают множество интересующих пользователя данных. Предполагаемый результат выполнения данного запроса показан на рисунке 1, где маркерами обозначены записи таблицы.

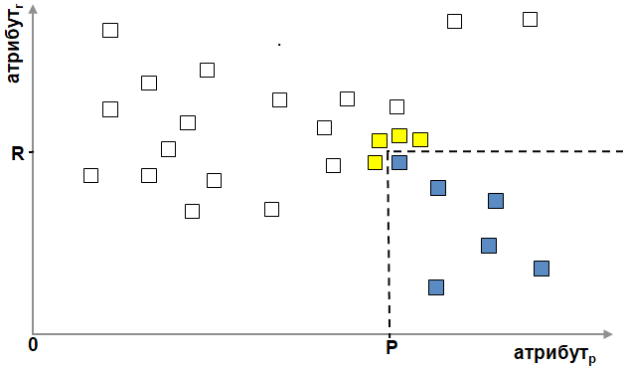


Рисунок 1. Результат выполнения четкого запроса

На рисунке 1 показано, что четыре записи (маркеры желтого цвета) очень близки к удовлетворению условия запроса. Но SQL использует четкую логику в процессе выполнения запроса, что означает четкий выбор. Следовательно, эти записи таблицы не будут присутствовать в

результатах запроса. По мере усложнения критериев фильтрации, мощность множества записей, соответствующих условию оператора WHERE четкого запроса, стремится к нулю.

Нечеткость в четкий запрос SQL может быть введена следующим образом:

SELECT $атрибут_1$, …, $атрибут_n$
FROM *таблица*
WHERE $атрибут_p > P\text{-}p$ AND $атрибут_r < R+r$;

Здесь p и r используются для расширения начальных критериев запроса P и R с целью выбора записей, которые почти соответствуют критериям четкого запроса. Но этот подход имеет следующий недостаток: в таблице базы данных также могут присутствовать записи, которые очень близки к удовлетворению уже нового, расширенного условия запроса, и как следствие необходимо выполнить еще одно расширение запроса. В итоге, в результатах выполнения запроса будет больше записей из таблицы базы данных, но пользователь потеряет точность своего запроса.

Проблема может быть решена путем использования в запросах к базам данных лингвистических терминов, представленных в виде нечетких множеств в соответствующем, обычно числовом, пространстве. Таким образом, вводится понятие степени соответствия данных запросу, и предполагается, что значение этой степени определяет принадлежность данных нечеткому множеству, представляющему условие запроса.

Анализ известных методов реализации нечетких запросов к четким реляционным базам данных. Сферы применения и преимущества нечетких запросов перед традиционным способом формирования запросов показаны в работах (Estefeev & Balashova 2016; Rybanov 2019). Среду поиска информации можно представить в виде кортежа:

<база данных, запрос>.

В табл. 1 приведена классификация нечеткости в среде поиска информации.

Таблица 1. Нечеткость в среде поиска информации

Тип нечеткости	База данных	Запрос
1	Четкая	Четкий
2	Четкая	Нечеткий
3	Нечеткая	Четкий
4	Нечеткая	Нечеткий

Рассмотрим подходы, ориентированные на расширение четких баз данных возможностью работы с нечеткими запросами.

Существующие в настоящее время практические методы реализации нечетких запросов к четким реляционным базам данных основаны на дополнительном информационном и программном слоях, которые необходимы для перевода нечетких запросов на уровень классической системы управления базами данных (СУБД). В работе (Smolka & Bradac 2017) рассмотрен подход реализации нечетких запросов, основанный на расширении схемы реляционной базы данных дополнительной таблицей для фиксации определения отдельных нечетких наборов. В работе (Hudec 2009) для обеспечения поддержки запросов, основанных на лингвистических выражениях, предлагается нечеткое обобщенное логическое условие для спецификации WHERE SQL, что позволяет использовать встроенные в СУБД обработку и выполнение SQL-запросов. В работе (Mama & Machkour 2021) для записи нечетких запросов, без необходимости их последующего перевода, предложен подход к расширению языка SQL, основная идея которого состоит в применении объекта БД VIEW (представление) для манипулирования степенями удовлетворенности, связанными с определяемыми пользователем нечеткими предикатами. В работе (Kilic, Abdullayev & Alakbarov 2016) рассматриваются вопросы организации, обработки и программной реализации нечетких запросов, основанной на языке SQL в виде специальной библиотеки функций. В работе (Sabour, Gadallah & Hefny 2014) предлагается гибкий нечеткий подход для запросов к реляционным базам данных путем оценки каждого кортежа непосредственно с использованием хранимых объектов базы данных. В работе (Trefilov 2017) поддержку нечетких запросов предлагается реализовать на стороне приложения. Недостатком данного похода является сильная зависимость программного кода приложения от модификаций схемы базы данных. В работе (Ventsov, Dolgov, & Podkolzina 2015) предлагается расширение спецификации WHERE оператора SQL условным выражением, полученным в результате преобразования процесса вычисления значения функции принадлежности в SQL-выражение. В работе (Nowakowski 2018) рассмотрен детальный анализ нечетких запросов и предлагаются способы конвертация их в стандартные SQL-запросы с помощью Oracle 11g XE. В работе (Muminov 2016) предлагается использовать хранимые процедуры для вычисления минимального и максимального значения функции принадлежности по ключевым словам лингвистической переменной.

Рассмотренные выше подходы к реализации нечетких запросов имеют следующие общие недостатки:

– снижение удобочитаемости и понимания SQL-запросов;

– сложность интеграции с уже существующими базами данных информационных систем;

– отсутствие возможности гибкой настройки функции принадлежности лингвистических переменных в соответствии с потребностями пользователя базы данных.

Адаптация четкой базы данных к реализации работы с нечеткими запросами. Предлагаемый метод реализации возможности работы с нечеткими запросами основан на расширении четкой базы данных, без изменения структуры и состава ее таблиц, следующими хранимыми функциями (stored function):

– хранимыми функциями для описания лингвистических переменных (по одной хранимой функции на каждую лингвистическую переменную).

– хранимыми функциями для описания функций принадлежности нечетких переменных (по одной хранимой функции на каждый терм лингвистической переменной);

Обобщенная сигнатура хранимой функции SQL, описывающей функцию принадлежности нечеткой переменной, может быть представлена в расширенной форме Бэкуса-Наура (РФБН) следующим образом:

<имя ФП>(***<нечеткая*** *переменная>*,

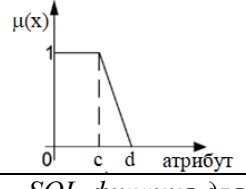
<параметр $_1$***>***{,***<параметр*** $_n$***>***}) RETURNS DOUBLE;

где ***<имя*** *ФП>* – имя функции принадлежности нечеткой переменной; ***<параметр*** $_1$***>,…,<параметр*** $_n$***>*** – значения параметров, описывающих вид функции принадлежности; ***<нечеткая*** *переменная>* – значение нечеткой переменной, для которой вычисляется степень принадлежности.

Для описания функций принадлежности нечетких переменных будем использовать кусочно-линейных функции, реализации которых на языке SQL представлены в табл. 2-4, где a, b, c, d – границы нечетких чисел x, используемые для задания ядра и носителя термов лингвистических переменных.

Таблица 2. SQL-реализация Z-образной функции принадлежности

График линейной Z-образной функции принадлежности



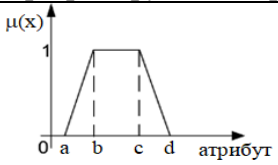
$$\mu(x)=\begin{cases}1 & ; \quad x \le c \\ \dfrac{d-x}{d-c} & ; \quad c < x \le d \\ 1 & ; \quad x > d\end{cases}$$

SQL-функция для Z-образной функции принадлежности

```
CREATE FUNCTION z_fig(x INT, c INT, d INT)
  RETURNS double
BEGIN
  CASE
    WHEN x <= c THEN RETURN 1;
    WHEN x > c AND x <= d THEN RETURN (d - x) / (d - c);
    WHEN x > d THEN RETURN 0;
  END CASE;
END
```

Таблица 3. SQL-реализация трапецивидной функции принадлежности

График функции принадлежности трапециевидной формы



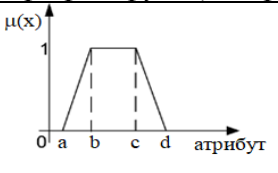
$$\mu(x)=\begin{cases}0 & ; \quad x<a\\ \dfrac{x-a}{b-a} & ; \quad a\le x<b\\ 1 & ; \quad b\le x\le c\\ \dfrac{d-x}{d-c} & ; \quad c<x\le d\\ 0 & ; \quad x>d\end{cases}$$

SQL-функция для трапециевидной функции принадлежности

```
CREATE FUNCTION.trap(x INT, a INT, b INT, c INT, d INT)
  RETURNS double
BEGIN
  CASE
    WHEN x < a THEN RETURN 0;
    WHEN x >= a AND x < b THEN RETURN (x - a) / (b - a);
    WHEN x >= b AND x <= c THEN RETURN 1;
    WHEN x > c AND x <= d THEN RETURN (d - x) / (d - c);
    WHEN x > d THEN RETURN 0;
  END CASE;
END
```

Таблица 4. SQL-реализация S-образной функции принадлежности

График функции принадлежности трапециевидной формы



$$\mu(x)=\begin{cases}0 & ; \quad x<a\\ \dfrac{x-a}{b-a} & ; \quad a\le x<b\\ 1 & ; \quad b\le x\le c\\ \dfrac{d-x}{d-c} & ; \quad c<x\le d\\ 0 & ; \quad x>d\end{cases}$$

SQL-функция для трапециевидной функции принадлежности

```
CREATE FUNCTION.trap(x INT, a INT, b INT, c INT, d INT)
  RETURNS double
BEGIN
  CASE
    WHEN x < a THEN RETURN 0;
    WHEN x >= a AND x < b THEN RETURN (x - a) / (b - a);
    WHEN x >= b AND x <= c THEN RETURN 1;
    WHEN x > c AND x <= d THEN RETURN (d - x) / (d - c);
    WHEN x > d THEN RETURN 0;
  END CASE;
END
```

Обобщенная сигнатура хранимой функции SQL, описывающей лингвистическую переменную, может быть представлена в РФБН следующим образом:

<имя** ЛП>*(<атрибут** БД>*, ***<терм** ЛП>*) RETURNS DOUBLE;

где ***<имя** ЛП>* – имя лингвистической переменной; ***<атрибут** БД>* – значение атрибута базы данных; ***<терм** ЛП>* – терм лингвистической переменной. Хранимая функция возвращает степень принадлежности значения ***<атрибут** БД>* терму лингвистической переменной <терм ЛП>.

Применение предлагаемого подхода рассмотрим на примере четкой базы данных «Регистрация договоров аренды», логическая схема которой приведена на рис. 2. В таблице Квартиры представлена информация по объектам недвижимости, которые предоставляются в аренду.

Рисунок. 2. Схема базы данных «Учет договоров аренды»

Определим лингвистические переменные следующих атрибутов базы данных: стоимость аренды и площадь квартиры.

Лингвистическую переменную, описывающую стоимость аренды однокомнатной квартиры в сутки опишем в виде 3-х термовой модели:

– b – «стоимость аренды»;

– $T=\{a_1,a_2,a_3\}$, a_1= «дешевая», a_2= «средняя», a_3=«дорогая». Терм «дешевая» имеет носитель [0;1050] и ядро [0;950]. Терм «средняя» имеет носитель [1000;1500] и ядро [1150;1350]. Терм «дорогая» имеет носитель [1400; ∞] и ядро [1500;∞].

SQL реализация функций принадлежности лингвистической переменной «стоимость аренды» приведена в табл. 5. Функция лингвистической переменной это комбинация функций принадлежности нечетких переменных. При описании лингвистической переменной использованы хранимые функции для Z-образной, трапецивидной и S-образной функций принадлежности:

– z_fig(attrib, 950, 1050);
– trap(attrib, 1000, 1150, 1350, 1500);
– s_fig(attrib, 1400, 1500).

Таблица 5. SQL-реализация лингвистической переменной «стоимость аренды»

Хранимая функция описания лингвистической переменной «стоимость аренды»
CREATE FUNCTION fprice(attrib INT, term VARCHAR(10)) RETURNS DOUBLE BEGIN CASE term WHEN 'дешевая' THEN RETURN z_fig(attrib, 950, 1050); WHEN 'средняя' THEN RETURN trap(attrib, 1000, 1150, 1350, 1500); WHEN 'дорогая' THEN RETURN s_fig(attrib, 1400, 1500); ELSE RETURN 0; END CASE; END

Лингвистическую переменную, описывающую площадь квартиры в виде 3-х термовой модели:

– b – «площадь квартиры»;

– T={g_1,g_2,g_3}, g_1= «маленькая», g_2= «средняя», g_3=«большая». Терм «маленькая» имеет носитель [0;30] и ядро [0;24]. Терм «средняя» имеет носитель [29;40] и ядро [33;38]. Терм «дорогая» имеет носитель [42; ∞] и ядро [39;∞].

SQL реализация функций принадлежности лингвистической переменной «площаль квартиры» приведена в таблице 6.

Таблица 6. SQL-реализация лингвистической переменной «площадь квартиры»

Хранимая функция описания лингвистической переменной «площадь квартиры»

```
CREATE FUNCTION farea(attrib INT, term VARCHAR(10))
  RETURNS DOUBLE
BEGIN
  CASE term
    WHEN 'маленькая' THEN RETURN z_fig(attrib, 24, 30);
    WHEN 'средняя' THEN RETURN trap(attrib, 29, 33, 38, 40);
    WHEN 'большая' THEN RETURN s_fig(attrib, 39, 42);
  END CASE;
END
```

Описать подобным образом можно любые термовые модели и функции принадлежности нечетких переменных.

Продемонстрируем применение метода адаптации реляционных баз данным к реализации нечетких запросов. Текущее состояние таблицы apartments (квартиры) представлено на рис. 3.

id	address	area	rooms	balcony	price
1	ул. Мира, 31	31	1	0	1300
2	ул. Балканская, 51	24	1	0	950
3	б-р. Ладыгина, 17	28	1	0	1380
4	ул. Труда, 27	28	1	0	1000
5	б-р. Профсоюзов, 42	32	1	1	1550
6	ул. Танковая, 34	34	1	1	1520
7	ул. Пушкина, 21	32	1	1	1450
8	ул. Дружбы, 10	42	1	1	1750
9	ул. Архитекторов, 77	39	1	1	1630
10	ул. Гагарина, 27	34	1	1	1300
11	пл. Чехова, 16	42	1	0	1480
12	ул. Домодедовская, 95	28	1	0	1370

Рисунок 3. Текущее состояние таблицы apartments (квартиры)

На рис. 4. приведены результаты вычисления степеней принадлежности атрибутов price (стоимость аренды) и area (площадь квартиры) с использованием соответствующих хранимых функций для лингвистических переменных (табл.5-6).

id	дешевая(price)	средняя(price)	дорогая(price)	маленькая(area)	средняя(area)	большая(area)
1	0	1	0	0	0,5	0
2	1	0	0	1	0	0
3	0	0,8	0	0,33	0	0
4	0,5	0	0	0,33	0	0
5	0	0	1	0	0,75	0
6	0	0	1	0	1	0
7	0	0,33	0,5	0	0,75	0
8	0	0	1	0	0	1
9	0	0	1	0	0,5	0
10	0	1	0	0	1	0
11	0	0,13	0,8	0	0	1
12	0	0,87	0	0,33	0	0

Рисунок 4. Значения функций принадлежности лингвистических переменных «стоимость аренды» и «площадь квартиры»

Рассмотрим примеры нечетких запросов.

Пример 1. Пусть требуется получить сведения об однокомнатных квартирах со стоимостью аренды 1350 руб. в сутки и общей площадью 35 кв.м. Четкий SQL-запрос будет иметь следующий вид:

```
SELECT * FROM  apartments
WHERE rooms=1 AND area=35 AND price<=1300;
```

Квартира со стоимостью аренды 1300 руб. и площадью 34 кв.м. не попадет в результат запроса, хотя ее характеристики практически соответствуют требованиям запроса. Результат выполнения данного запроса будет пустым.

Аналогичный нечеткий SQL-запрос будет иметь следующий вид:

```
"SELECT * FROM apartments
WHERE rooms = 1 AND f area(area, 'средняя') AND price(price, 'средняя')
ORDER BY LEAST(area(area, 'средняя'), price(price, 'средняя')) DESC;"
```

Т.к. в нечетком запросе два параметра запроса farea(area, ‘средняя’) и fprice(price, ‘средняя’) определены степенью принадлежности, то для сортировки результирующих строк по релевантности запросу используется спецификация ORDER BY (табл. 7).

Таблица 7. Правила формирования спецификации ORDER BY

Оператор	Выражение для сортировки
AND	LEAST(<*нечеткий параметр*$_1$> , …, <*нечеткий параметр*$_n$>)
OR	GREATEST(<*нечеткий параметр*$_1$>, …, <*нечеткий параметр*$_n$>)

LEAST() и GREATEST() стандартные функции MySQL для выбора минимального и максимального значения и списка.

В результате выполнения запроса будут отобраны записи с id равными 1, 7 и 10 (рис. 5).

id	address	area	rooms	balcony	price	средняя(area)	средняя(price)
10	ул. Гагарина, 27	34	1	1	1300	1	1
1	ул. Мира, 31	31	1	0	1300	0,5	1
7	ул. Пушкина, 21	32	1	1	1450	0,75	0,33

Рисунок 5. Пример 1: Результат выполнения нечеткого запроса

Пример 2. Пусть требуется найти однокомнатную квартиру с дешевой стоимостью аренды. Нечеткий SQL-запрос будет иметь следующий вид:

```
SELECT * FROM apartments
WHERE  rooms = 1 AND fprice(price, 'дешевая')
ORDER BY  fprice(price, 'дешевая') DESC;
```

В результате выполнения запроса будут отобраны записи с id равными 2 и 4 (рис. 6).

id	address	area	rooms	balcony	price	дешевая(price)
2	ул. Балканская, 51	24	1	0	950	1
4	ул. Труда, 27	28	1	0	1000	0,5

Рисунок 6. Пример 2: Результат выполнения нечеткого запроса

Расширим базовое терм-множество новыми лингвистическими термами. Для этого воспользуемся лингвистическими модификаторами very (очень) и more-or-less (более-или-менее), которые усиливают и ослабляют термы базового множества. Нечеткое множество усиливающего модификатора very определим функцией принадлежности вида:

$$\mu_{very}(x) = (\mu(x))^2 .$$

Нечеткое множество ослабляющего модификатора more-or-less определим функцией принадлежности вида:

$$\mu_{more-or-less}(x) = \sqrt{\mu(x)} .$$

Операцию нечеткого отрицания (*NOT*) определим как:

$$\overline{\mu(x)} = 1 - \mu(x) .$$

Пример 3. Необходимо найти более-или-менее средние по стоимости аренды однокомнатные квартиры с балконом. Нечеткий SQL-запрос будет иметь следующий вид:

```
SELECT * FROM apartments
WHERE rooms = 1  AND balcony AND sqrt(fprice(price, ‘средняя’))
ORDER BY  sqrt(fprice(price, ‘средняя’)) DESC;
```

Пример 4. Необходимо найти недорогие по стоимости аренды жилья однокомнатные квартиры. Нечеткий SQL-запрос будет иметь следующий вид:

```
SELECT * FROM apartments
WHERE rooms = 1 AND 1 - fprice(price, ‘дорогая’)
ORDER BY1 - fprice(price, ‘дорогая’) DESC;
```

Рассмотренные выше примеры нечетких запросов подтверждают простоту интеграции предлагаемого метода реализации механизмов нечетких запросов к уже действующим информационным системам, созданными на базе MySQL.

Заключение. На основе предлагаемого метода адаптации реляционных баз данных к реализации нечетких запросов показано, что создаваемые в рамках метода хранимые функции являются инвариантными по отношению к предметной области четкой базы данных. Приведенные примеры преобразования нечетких запросов в стандартные SQL-запросы с помощью MySQL указывают на простые в реализации методы получения нечеткой информации из базы данных и тем самым расширяют ее функциональные возможности.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

ЕСТЕФЕЕВ, В.И. & БАЛАШОВА, Ю.И., 2016. Нечеткие запросы к реляционной базе данных в информационной системе подбора персонала. *Образование и наука в современных условиях*, **2 – 2**(7), 69 – 72.

Муминов Б.Б., 2016. Формирование запросов для интеллектуального поиска в нечеткой информационной среде. *Наука и мир*, **3 – 1** (31), 79 – 85.

(2)РЫБАНОВ, А.А., 2019. Метрики разнообразия типов данных в физической схеме базы данных MySQL. *Вестник Адыгейского государственного университета*. Серия 4: Естественно-математические и технические науки, **4** (251), 87 – 90.

ТРЕФИЛОВ, П.А., 2017. Нечеткие запросы в реляционных базах данных. *Некоторые вопросы анализа, алгебры, геометрии и математического образования*, **6**, 197 – 198.

ВЕНЦОВ, Н.Н., ДОЛГОВ, Л.А. & ПОДКОЛЗИНА, Л.А., 2015. Об одном способе построения запросов к базе данных на основе аппарата нечеткой логики. *Инженерный вестник Дона*, **3** (37), 44.

REFERENCES

ESTEFEEV, V.I. & BALASHOVA, I. YU., 2016. Fuzzy queries to relational database in recruitment information system. *Education and science in modern settings,* **2 – 2** (7), 69 – 72.

HUDEC, M., 2009. An approach to fuzzy database querying, analysis and realization, *Computer Science and Information Systems,* **6**, 127 – 140, doi: 10.2298/CSIS0902127H.

KILIC, K., ABDULLAYEV, T., ALAKBAROV, R. & KILIC N., 2016. Processing of fuzzy queries and software implementation to a relational database of wholesale and retail commercial enterprises, *Paper presented at the Procedia Computer Science*, **102**, 490 – 494, doi:10.1016/j.procs.2016.09.

MAMA, R. & MACHKOUR, M., 2021. Fuzzy querying with SQL: Fuzzy view-based approach, *Journal of Intelligent and Fuzzy Systems*, **40**(5), 9937 – 9948, doi: 10.3233/JIFS-202551.

MUMINOV, B.B., 2016. Formation of queries for intelligent retrieval in fuzzy information environment, *Science and world*, **3 – 1** (31), 79 – 85.

(10)NOWAKOWSKI, G., 2018. Fuzzy queries on relational databases, *Paper presented at the 2018 International Interdisciplinary PhD Workshop*, IIPhDW, 293 – 299, doi: 10.1109/IIPHDW.2018.8388376.

RYBANOV, A.A., 2019. Diversity metrics of data types for physical schema of data base MySQL, *The Bulletin of the Adyghe State University, the series "Natural-Mathematical and Technical Sciences"*, **4** (251), 87 – 90.

SABOUR, A.A., GADALLAH, A.M. & HEFNY, H.A., 2014. *Flexible querying of relational databases: Fuzzy set based approach*, doi: 10.1007/978-3-319-13461-1_42.

SMOLKA, P. & BRADAC, V., 2017. Fuzzy queries above relational database, *Paper presented at the AIP Conference Proceedings*, **1906**, doi:10.1063/1.5012350.

TREFILOV, P.A., 2017. Fuzzy query in relational databases, *Some questions of analysis, algebra, geometry and mathematical education*, **6**, 197 – 198.

VENTSOV, N.N., DOLGOV, V.V. & PODKOLZINA, L.A., 2015. One method of constructing database queries based on fuzzy logic, *Ingineering journal of Don*, **3** (37), 44.

A PRACTICAL METHOD FOR IMPLEMENTING FUZZY QUERIES FOR RELATIONAL DATABASES

Abstract. Information systems that use databases are flexible to the extent that they allow users to request the data they need. SQL is limited to precise data processing and does not directly express fuzzy concepts of natural language. Therefore, giving SQL some flexibility can help users improve interaction with information systems without requiring them to learn a completely new language. The task of reducing the labor intensity of the process of integrating the mechanisms of fuzzy requests to existing information systems is urgent. The article shows the limitations of clear queries, considers various forms of fuzzy queries. Known approaches to implementing fuzzy queries to relational databases were analyzed. Provides a detailed analysis of fuzzy queries, as well as their conversion to standard SQL queries using MySQL. The proposed method for implementing the ability to work with fuzzy queries is based on expanding a clear database with stored functions, without changing the structure and composition of its tables. The advantages of the method are: increased readability and understanding of SQL queries; ease of integration with existing databases of information systems; flexible adjustment of the linguistic variable membership function in accordance with the needs of the database user. The application of the method is shown by the example of adapting a MySQL database. The proposed adaptation method can be widely used to implement fuzzy queries to databases of various DBMS that support work with stored functions.

Keywords: relational databases; fuzzy logic; fuzzy queries; SQL; MySQL

✉ **Dr. Alexander Aleksandrovich Rybanov, Assoc. Prof.**
ORCID ID: 0000-0002-8638-9998
Department of Informatics and ProgrammingTechnologyVolzhsky
Polytechnic Institute, branch of the VolgGTU, Volzhsky
Volgograd, Russia
E-mail: rybanoff@yandex.ru