УДК 681.3.07

DOI: 10.15827/0236-235X.120.609-618

Дата подачи статьи: 18.09.17

2017. Т. 30. № 4. С. 609–618

ХРАНЕНИЕ И ЭФФЕКТИВНАЯ ОБРАБОТКА НЕЧЕТКИХ ДАННЫХ В СУБД PostgreSQL

*В.Е. Сорокин, к.т.н., ведущий научный сотрудник, sorokinve@yandex.ru
(НИИ «Центрпрограммсистем», просп. 50 лет Октября, 3а, г. Тверь, 170024, Россия)*

Всего за полвека теория нечетких множеств прошла путь от разработки теоретических основ до широкого практического применения в системах искусственного интеллекта таких областей деятельности, как управление промышленным оборудованием и транспортными средствами, медицинская диагностика и экспертные системы, в том числе для оценки различных рисков, от экономических до экологических. Во многих из них требуются надежное хранение и эффективная обработка больших объемов информации. С этой целью, как правило, применяются СУБД, наиболее совершенными из которых в настоящее время являются объектно-реляционные. Однако нечеткие данные плохо согласуются как с объектной, так и с реляционной моделью данных, и большинство промышленных СУБД не содержат встроенных типов нечетких данных и механизмов работы с ними. К таковым относятся и кроссплатформенная свободно распространяемая в исходных кодах объектно-реляционная СУБД PostgreSQL, с применением которой во многом связывается импортозамещение в инфраструктурном ПО.

Возможности реляционного моделирования нечетких данных и развитый механизм расширения типов за счет создания требуемых пользовательских типов наряду с мощными процедурными языками в СУБД PostgreSQL позволяют реализовывать различные альтернативные подходы к хранению и обработке нечетких данных.

В статье выполнен сравнительный анализ таких подходов с точки зрения поддержания целостности данных и эффективности их обработки, приведены результаты экспериментов на созданных по различным подходам моделях нечетких данных.

Предложены проектные решения, повышающие эффективность поиска подобных нечетких данных, сформулированы рекомендации по выбору подхода к моделированию нечетких данных.

Ключевые слова: лингвистическая переменная, нечеткое множество, система управления базами данных, PostgreSQL, треугольные нормы, операции нечеткой логики, целостность данных, эффективность SQL-запросов.

Сформулированные Л. Заде определения нечеткого множества как объекта с функцией принадлежности элемента к множеству, принимающей любые значения в интервале от 0 до 1, основных логических операций над нечеткими множествами и лингвистической переменной как переменной, значениями которой являются нечеткие множества, положили начало теории нечетких множеств. Ее основным отправным отличием от классической теории множеств является использование для определения принадлежности множества элемента универсального множества функции принадлежности с любыми значениями от 0 до 1 вместо характеристической функции с фиксированными значениями 0 или 1. Лингвистические переменные принимают лингвистические, а не числовые значения [1]. Например, для лингвистической переменной Возраст значениями могут быть «очень молодой», «молодой», «немолодой», «старый» и т.д. Совокупность значений лингвистической переменной составляет ее терм-множество. Числовая переменная Возраст, принимающая значения 0, 1, 2, ..., 120, является базовой для лингвистической переменной Возраст. При этом значение «молодой» интерпретируется как название нечеткого ограничения на базовую переменную, смысл которого определяется его функцией совместимости. Так, совместимость возраста 27 лет со значением «молодой» может быть равна 0,9, а 35 лет – 0,6. Если значения численной переменной изображают графически точками на плоскости, то значения лингвистической переменной Л. Заде стал изображать как пло-

щадки с нечетко очерченными границами. Такая интерпретация позволяет использовать их как средство приближенного описания явлений, которые настолько сложны или недостаточно определены, что не поддаются точному описанию.

Довольно быстро стала очевидной пригодность практического применения теории нечетких множеств в различных системах искусственного интеллекта, экспертных системах. Например, для управления работой прокатного стана вводится функция принадлежности нагрева, а для управления автомобилем разрабатывается контроллер педалей газа и тормоза. Показателями оценки автовождения поезда метрополитена на основе нечеткого управления являются скорость и безопасность его движения, время в пути и точность остановки. Типичным применением теории нечетких множеств является построение системы медицинской диагностики, базирующейся на определении достоверности утверждений о связи между болезнями и симптомами. В ней информация, получаемая от врача и больного, включает нечеткость, выражаемую лингвистическими значениями истинности (ЛЗИ) семи уровней, которые следует преобразовать в числовые значения истинности для выполнения необходимых вычислений. Типичные значения семи уровней ЛЗИ: «очень правдивое» (VT), «довольно правдивое» (RT), «возможно правдивое» (PT), «возможно ложное» (PF), «довольно ложное» (RF), «очень ложное» (VF), «неизвестное» (UN). При этом с применением так называемых α -сечений устанавливается, каким образом

по функции принадлежности каждого ЛЗИ выбираются значения принадлежности. Выводы получаются с применением нечеткого «модус поненс» ($\{x_1 \Rightarrow x_2, x_1\} \Rightarrow x_2$) как нечеткого варианта правила дедуктивного вывода с использованием базы знаний для функций принадлежности, правил, диапазонов входных и выходных значений и указаний к окончательным выходным данным. Предполагаемое Л. Заде широкое практическое применение теории нечетких множеств благодаря возможности развивать приближенное исчисление лингвистических переменных все более подтверждается [2].

Качество практического применения теории нечетких множеств во многом зависит от объема хранимой и обрабатываемой в системе информации, для чего в настоящее время, как правило, используются объектно-реляционные СУБД. Однако нечеткие данные плохо согласуются как с объектной, так и с реляционной моделью данных, и отсутствие в большинстве промышленных СУБД встроенных типов нечетких данных и механизмов работы с ними представляет определенную проблему их эффективного использования. Указанная проблема касается и кроссплатформенной свободно распространяемой в исходных кодах объектно-реляционной СУБД PostgreSQL, возглавляющей направление СУБД в «Протоколе экспертной оценки проектов по импортозамещению инфраструктурного программного обеспечения» Минкомсвязи России от 2 июня 2015 года.

Особенности представления и обработки нечетких данных

Как отмечалось, обычное четкое множество A в полном пространстве X (совокупности элементов x предметной области) определяется характеристической функцией $h_A(x)$, принимающей значение 1 для каждого входящего в множество элемента и значение 0 для всех остальных элементов предметной области. В отличие от него нечеткое множество B в полном пространстве X определяется функцией принадлежности $m_B(x)$, принимающей значение от 0 до 1 и обозначающей субъективную оценку степени принадлежности x множеству B , то есть $m_B: X \rightarrow L$, где $L = [0, 1]$. Таким образом, нечеткое множество – это пара (B, m) из имени нечеткого множества и функции принадлежности. Обычно оно имеет некоторую лингвистическую метку, соответствующую его содержательной интерпретации, и может рассматриваться как значение некоторой лингвистической переменной.

Степень принадлежности может трактоваться по-разному в зависимости от задачи, в которой используется нечеткое множество. Примерами такой трактовки могут быть степень соответствия понятию, вероятность, возможность, полезность, истинность, правдоподобность и другие, включая значения каких-либо функций. Для каждой трактовки

степени принадлежности используются свои методы построения функций принадлежности, в том числе их задание в параметрическом виде с последующей настройкой в процессе отладки с целью уменьшения ошибки рассогласования между нечеткой моделью и моделируемой системой.

Если четкое множество ограниченной мощности можно описать перечнем входящих в него элементов, то для описания нечеткого множества в ограниченном полном пространстве необходимо привести все отличные от 0 значения функции принадлежности. Например, для полного пространства натуральных чисел меньше 8 четкое множество A четных чисел описывается как $A = \{2, 4, 6\}$, а нечеткое множество B малых чисел, к которым на 100 % принадлежат числа 1 и 2, число 3 принадлежит на 70 % и число 4 на 30 %, описывается как $B = 1/1 + 1/2 + 0,7/3 + 0,3/4$. При этом могут существовать нечеткие множества C и D малых чисел, определенные другими экспертами как $C = 1/1 + 0,5/2 + 0,1/3$ и $D = 1/1 + 0,7/2$.

Очевидно, что в неограниченном полном пространстве как четкие, так и нечеткие множества определяются не перечислительно. Например, четкое множество может определяться как диапазон вещественных чисел, а функция принадлежности нечеткого множества может задаваться графически. Наиболее распространенными в приложениях теории нечетких множеств являются треугольные, трапециевидные, гауссовские и колоколообразные функции принадлежности. Если гауссовские функции принадлежности задаются двумя параметрами (c, s) : $m(x) = \exp(-0,5(x-c)^2/s^2)$, то колоколообразные – тремя (a, b, c) : $m(x) = 1/(1+(x-c/a)^{2b})$, а трапециевидные – четырьмя (a, b, c, d) :

$$m(x) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & a < x \leq b, \\ 1, & b < x \leq c, \\ \frac{d-x}{d-c}, & c < x \leq d, \\ 0, & x > d. \end{cases}$$

Треугольные функции принадлежности являются частным случаем трапециевидных при $b = c$ и задаются, соответственно, тремя параметрами. Однако ввиду часто используемого оценочного характера нечетких данных и связанной с участием человека их антропометричности, прежде всего эргономическими ограничениями количества контролируемых человеком объектов или значений, на практике целесообразно использовать ограниченное полное пространство, в котором определяются нечеткие множества ограниченной мощности. С учетом данного фактора можно считать приемлемым ограничение задания нечетких множеств перечислительным способом, особенно для систем искусственного интеллекта и экспертных систем, в которых наиболее востребованы нечеткие данные.

Подобно тому, как операциям пересечения, объединения и дополнения обычных четких множеств взаимно однозначным образом ставятся в соответствие операции над их характеристическими функциями, определяемые поэлементно (для всех $x \in X$), те же операции над нечеткими множествами определяются поэлементно над их функциями принадлежности: $(B \cap C)(x) = B(x) \wedge C(x)$, $(B \cup C)(x) = B(x) \vee C(x)$, $(\neg B)(x) = \neg B(x)$. В качестве операций конъюнкции, дизъюнкции и отрицания Л. Заде предложил следующее обобщение булевых функций: $(x \in X, y \in X): x \wedge y = \min(x, y)$, $x \vee y = \max(x, y)$, $\neg x = 1 - x$. В общем случае операции и отношения на множестве нечетких множеств определяются также поэлементно с помощью операций и отношений на элементах из X . Отношение включения $\subseteq$ нечетких множеств является отношением частичного порядка, поскольку удовлетворяет условиям рефлексивности $B \subseteq B$, антисимметричности $\{B \subseteq C, C \subseteq B\} \Rightarrow B = C$ и транзитивности $\{B \subseteq C, C \subseteq D\} \Rightarrow B \subseteq D$. На множестве всех нечетких подмножеств множества X операции конъюнкции, дизъюнкции и отрицания удовлетворяют тождествам идемпотентности $B \cap B = B$, коммутативности $B \cap C = C \cap B$, ассоциативности $B \cap (C \cap D) = (B \cap C) \cap D$, и аналогично ... для операции $\cup$, поглощения $B \cap (B \cup C) = B$, ..., дистрибутивности $B \cap (C \cup D) = (B \cap C) \cup B \cap D$, ..., инволютивности $\neg(\neg B) = B$, законам Де Моргана $\neg(B \cap C) = \neg B \cup \neg C$, ... и граничным условиям $B \cap \emptyset = \emptyset$, $B \cap U = B$, $B \cup \emptyset = B$, $B \cup U = U$, где $\emptyset(x) = 0$ и $U(x) = 1$ для всех $x \in X$ [3].

Введенные Л. Заде операции конъюнкции, дизъюнкции и отрицания обладают почти всеми свойствами соответствующих булевых операций, что позволяет легко обобщать на нечеткий случай многие понятия «четкой» логики. Однако наряду с такими преимуществами, как адекватность в порядковых шкалах, в которых обычно измеряются лингвистические оценки, операциям $\min$ и $\max$ присущи такие существенные во многих приложениях недостатки, как идентичность результата одному из операндов в существенном диапазоне изменения второго оператора.

Наиболее распространенным расширением класса операций конъюнкции и дизъюнкции в двухзначной логике и операций $\min$ и $\max$ в нечеткой логике Л. Заде являются ассоциативные операции триангулярной нормы T (t -нормы) и триангулярной конормы S (t -конормы) соответственно, в общем случае недистрибутивные. Они определяются как функции $T, S: [0, 1]^2 \rightarrow [0, 1]$, для которых выполняются тождества коммутативности, ассоциативности и монотонности. Основное отличие t -нормы $T(x, y)$ от t -конормы $S(x, y)$ заключается в граничных условиях: $T(0, x) = T(x, 0) = 0$, $T(1, x) = T(x, 1) = x$, $S(0, x) = S(x, 0) = x$, $S(1, x) = S(x, 1) = 1$. Для t -нормы T двойственная к ней t -конорма S задается как $S(x, y) = 1 - T(1 - x, 1 - y) = N(T(N(x), N(y)))$, где N – операция отрицания. Несмотря на то,

что отсутствие свойства дистрибутивности не позволяет выполнять эквивалентные преобразования логических форм из дизъюнктивной в конъюнктивную форму и обратно, ассоциативность дает возможность построения сложных логических связей, достаточных для моделирования многих областей практического применения.

Простейшими основными и широко распространенными t -нормами и t -конормами являются операции минимум T_M и максимум S_M логики Заде, операции произведение T_P и вероятностная сумма S_P вероятностной логики, операции ограниченная t -норма T_L и ограниченная t -конорма S_L логики Лукасевича, слабая t -норма T_W и сильная t -конорма S_W (драстические или радикальные произведение и сумма), определяемые следующим образом [4, 5]:

$$\begin{aligned} T_M(x, y) &= \min(x, y); \\ S_M(x, y) &= \max(x, y); \\ T_P(x, y) &= x * y; \\ S_P(x, y) &= x + y - x * y; \\ T_L(x, y) &= \max(x + y - 1, 0); \\ S_L(x, y) &= \min(x + y, 1); \\ T_W(x, y) &= \min(x, y), \text{ если } \max(x, y) = 1, \text{ иначе } 0; \\ S_W(x, y) &= \max(x, y), \text{ если } \min(x, y) = 0, \text{ иначе } 1. \end{aligned}$$

Для них выполняются условия:

$$\begin{aligned} T_W(x, y) &\leq T_L(x, y) \leq T_P(x, y) \leq T_M(x, y); \\ S_M(x, y) &\leq S_P(x, y) \leq S_L(x, y) \leq S_W(x, y). \end{aligned}$$

При этом для любых t -норм и t -конорм выполняются следующие неравенства: $T_W(x, y) \leq T(x, y) \leq T_M(x, y) \leq S_M(x, y) \leq S(x, y) \leq S_W(x, y)$.

Существенно большей гибкостью моделирования по сравнению с приведенными простейшими t -нормами и t -конормами обладают параметрические t -нормы и t -конормы, такие как семейства t -норм и t -конорм Домби, Франка, Хамахера, Ягера. Большая гибкость достигается за счет их большей вычислительной сложности. Например, едва ли не наименее вычислительно сложные из них семейства t -норм T_Y^λ и t -конорм S_Y^λ Ягера определяются как

$$\begin{aligned} T_Y^\lambda(x, y) &= \begin{cases} T_W(x, y), & \text{если } \lambda = 0, \\ T_M(x, y), & \text{если } \lambda = +\infty, \\ \max(0, 1 - ((1 - x)^\lambda + (1 - y)^\lambda)^{1/\lambda}) & \text{иначе,} \end{cases} \\ S_Y^\lambda(x, y) &= \begin{cases} S_W(x, y), & \text{если } \lambda = 0, \\ S_M(x, y), & \text{если } \lambda = +\infty, \\ \min(1, (x^\lambda + y^\lambda)^{1/\lambda}) & \text{иначе.} \end{cases} \end{aligned}$$

Все t -нормы и t -конормы коммутативны. Для моделирования операций, результат выполнения которых зависит от порядка следования операторов, предлагаются параметрические классы обобщенных операций конъюнкции и дизъюнкции, для которых не выполняются тождества коммутативности. Может быть сгенерировано большое количество семейств таких операций [3], однако наибольший практический интерес представляют операции относительно низкой вычислительной сложности. Примерами таких операций с параметрами p и q могут служить конъюнкции:

$$\begin{aligned}
T(x, y) &= \min(x, y), \text{ если } p \leq x \text{ или } q \leq y, \text{ иначе } 0; \\
T(x, y) &= \min(x, y) * \max(1-p(1-x), 1-q(1-y), 0); \\
T(x, y) &= \min(\min(x, y), \max(x^p, y^q)); \\
T(x, y) &= \min(x, y) * \max(x^p, y^q).
\end{aligned}$$

В нечетких продукционных системах определенная t -норма ставится вместо оператора конъюнкции на степенях неопределенности, задаваемых значениями функций принадлежности, двух или более условий в одном и том же продукционном правиле. Соответственно, t -конорма вычисляет степень неопределенности (значение функции принадлежности) заключения, выведенного из двух или более продукционных правил, или степень неопределенности (значение функции принадлежности) условной части продукционного правила, в которой antecedentes соединены логической связкой «ИЛИ».

Принцип расширения как одна из основных идей теории нечетких множеств позволяет обобщить определения операций «*», таких как алгебраические операции сложения, вычитания, умножения или деления, на нечеткие множества $z = x * y$ как $m_z(z) = \max\{\min[m_x(x), m_y(y)]\}$. Поскольку теоретически существует бесконечное число комбинаций x и y , которые дают одну и ту же величину z , и на практике поиск комбинаций, обеспечивающих максимальную функцию принадлежности, может иметь очень большую вычислительную сложность, обычно применяют такие эвристические методы, как метод вершин. Его ключевым утверждением является то, что для операции «*» над двумя нечеткими числами $K = I * J$ оптимальная пара (x, y) получается в предположении $m_I(x) = m_J(y)$, что вместе с алгебраическим соотношением $x * y = z$ определяет функцию принадлежности решения. Для упрощения вычисления значений $m_K(z)$ возможна обработка с использованием α -срезов для K в сопоставлении их с соответствующими оптимальными α -срезами для I и J .

Начало развития теории многозначных логик на основе триангулярных норм положили работы Лукасевича 1920-х годов. В них, как и в классической булевой логике, где выражение «если H , то G » эквивалентно выражению «не H или G », можно построить импликацию I через отрицание N , конъюнкцию T и дизъюнкцию S как $I(x, y) = S(N(x), y) = N(T(x, N(y)))$. В этом случае закон противоположности $I(x, y) = I(N(y), N(x))$ всегда выполняется.

Для рассмотренных четырех простейших основных t -норм получаются следующие импликации:

$$\begin{aligned}
I_M(x, y) &= \max(1-x, y); \\
I_P(x, y) &= 1-x+x*y; \\
I_L(x, y) &= \min(1-x+y, 1); \\
I_w(x, y) &= \begin{cases} 1-x, & \text{если } y = 0, \\ y, & \text{если } x = 1, \\ 1 & \text{иначе.} \end{cases}
\end{aligned}$$

Другой способ расширения классического оператора двоичной импликации на $[0, 1]$ использует понятие границы R_T непрерывной слева t -нормы T : $R_T(x, y) = \sup\{z \in [0, 1] \mid T(x, z) \leq y\}$. В этом случае для непрерывных слева t -норм T_M , T_P и T_L получаются выражения:

$$\begin{aligned}
R_M(x, y) &= 1, \text{ если } x \leq y, \text{ иначе } y; \\
R_P(x, y) &= \min(y/x, 1); \\
R_L(x, y) &= \min(1-x+y, 1).
\end{aligned}$$

Выражение $\min(1-x+y, 1)$ для импликации введено Лукасевичем, поэтому соответствующие широко распространенные T_L и S_L называют t -нормой и t -конормой Лукасевича [4].

На импликативных правилах основаны простейшие широко используемые системы нечеткого логического вывода. Прежде всего это модели Мамдани и Сугено, основанные на правилах R_i и R_j соответственно, вида:

$$\begin{aligned}
R_i: & \text{ Если } X \text{ есть } A_i \text{ и } Y \text{ есть } B_i, \text{ то } Z \text{ есть } C_i; \\
R_j: & \text{ Если } X \text{ есть } A_j \text{ и } Y \text{ есть } B_j, \text{ то } z = f_j(x, y),
\end{aligned}$$

где X, Y, Z – нечеткие переменные одного типа, например Возраст; A_i, B_i, C_i – их нечеткие значения, например «молодой»; x, y, z – численные значения соответствующих переменных; f_j – некоторые вещественные функции. Для заданных значений x и y сила срабатывания правила w_i вычисляется как $w_i = T_1(m_{A_i}(x), m_{B_i}(y))$, где T_1 – некоторая операция конъюнкции; $m_{A_i}(x)$ и $m_{B_i}(y)$ – значения принадлежности x и y нечетким множествам A_i и B_i . Заключение правил может быть вычислено как $m_{C_i}(z) = T_2(w_i, m_{C_i}(z))$ или $z_i = T_2(w_i, f_i(x, y))$, где T_2 – операция конъюнкции, используемая в операции импликации и, возможно, отличная от T_1 . Для агрегирования заключений, полученных по всем правилам, может использоваться некоторая операция дизъюнкции. Кроме того, в моделях Мамдани используется процедура преобразования нечеткого множества, полученного в результате логического вывода, в число, называемая дефазификацией.

Оптимизация нечетких моделей традиционно основана как на настройке функций принадлежности нечетких множеств, так и на настройке операций, в том числе на оптимизации параметров параметрически заданных операций. Последнее может оказаться достаточно трудоемким ввиду сложного вида известных параметрических классов t -норм и t -конорм, используемых в качестве операций конъюнкции и дизъюнкции. С этой точки зрения преимущества имеют более простые параметрические классы операций конъюнкции и дизъюнкции [3].

Возможности СУБД PostgreSQL по хранению и обработке нечетких данных

К предназначенной для работы с нечеткими данными СУБД должны предъявляться следующие специфические требования. Она должна поддерживать типы данных, позволяющих хранить нечеткие данные с обеспечением их целостности, а также

эффективно выполнять такие основные операции работы с ними, как разнообразные t -нормы, t -ко-нормы и импликации, получение результатов α -срезов и дефазификации, поиск наиболее соответствующих задаваемым значениям функции принадлежности предопределенных значений лингвистических переменных и т.п. Несмотря на наличие в СУБД PostgreSQL наряду с базовыми типами достаточного количества различных встроенных типов данных, операторов и функций работы с ними, как в последней на момент написания статьи версии 9.6.4 [6], так и в планируемой к выпуску версии 10.1 [7] отсутствует какая-либо поддержка нечетких типов данных. Это касается и подключаемых к системе в виде созданных по предопределенным правилам программных модулей – расширений, как собственных, созданных в рамках проекта PostgreSQL его разработчиками, так и внешних.

Как отмечалось выше, в данной работе рассматривается определение нечетких множеств ограниченной мощности в ограниченных полных пространствах перечислительным способом. Поэтому возможна организация их хранения в БД классическим реляционным способом путем создания отдельных таблиц с полями базовых для реляционных СУБД типов данных для каждого полного пространства и определенных на нем нечетких множеств (возможно, также лингвистических переменных и их значений) в соответствии с нормализуемой моделью атрибутов сущностей. Классические реляционные механизмы, включая ключи и внешние ссылки, являются надежным инструментом поддержания целостности данных. Их эффективность поддерживается B-tree индексами, которые могут быть простыми или уникальными, составными, частичными, по выражениям или функциям, а также конкурентными без блокировок на запись. Использование базовых типов данных делает возникновение конфликтов блокировок изменяемых полей и записей маловероятным. Однако при классическом реляционном подходе переход с четких данных на нечеткие только в одном поле таблицы приводит к необходимости при запросе этих данных вместо обращения к единственному полю выполнять соединение минимум с одной таблицей, резко увеличивая вычислительную сложность запроса.

Избежать «проклятия» множественных соединений позволяет использование полей с типами нечетких данных, которые могут быть созданы с помощью механизма расширения системы типов подобно тому, как в СУБД PostgreSQL встроены специализированные типы данных, например, геометрические типы и типы сетевых адресов. Поскольку хранение в БД нечеткого множества сводится к хранению его имени (его лингвистической метки) и функции принадлежности, тип нечетких данных может быть построен как композитный тип, состоящий из строкового типа и массива или

поля типа `hstore`, с соответствующими проверочными ограничениями, а также как сложный тип `xml`, `json` или `jsonb` с соответствующими спецификациями. Для эффективной работы с такими типами наряду с классическими индексами типов B-tree и Hash также используются индексы типов GiST и SP-GiST с обобщенным деревом поиска и инверсные индексы типа GIN, которые являются инфраструктурой для различных стратегий поиска.

Однако при этом нельзя не учитывать такие недостатки объектной модели, реализуемой сложными типами данных, по сравнению с нормализуемой моделью атрибутов сущностей, как ограниченность механизмов поддержания целостности данных, избыточность данных и большая конкурентность при параллельном доступе. Ввиду отсутствия поддержки ссылочной целостности для поддержания целостности данных (спецификации схемы) следует использовать ограничения, вызывающие функции от данных сложного типа, проверяющие, например, наличие заданного ключа в `hstore`, правильность сформированного `xml` документа или заданного пути в `json`. Проверочные выражения для сложных типов могут быть весьма не тривиальны, и полноценное поддержание целостности данных сложного типа в настоящий момент возможно только созданием наборов соответствующих триггеров и написанием вызываемых ими оригинальных триггерных функций. Отсутствие нормализации в общем случае приводит к значительному дублированию данных в объектной модели. Кроме того, это увеличивает накладные расходы при корректировке таких данных, а также вероятность блокировок или конфликтов изменения при параллельном доступе к ним. Хранение сложного объекта в одном поле записи БД приводит к невозможности одновременного изменения различными пользователями любых его свойств [8].

Широкий набор встроенных операторов и функций для обработки данных различных встроенных типов и разрешение пользователям создавать агрегатные функции, операторы и типы данных, хранимые процедуры и функции на различных процедурных языках, в том числе для работы с созданными типами данных, предоставляют достаточно возможностей для выполнения основных операций (функций) с нечеткими данными. Эффективность их выполнения при этом в основном определяется типами (и структурами) данных их хранения.

Примеры реализации хранения и обработки нечетких данных в СУБД PostgreSQL

Очевидно, что двумя наиболее принципиально различающимися подходами к хранению и обработке нечетких данных в БД, работающей под управлением СУБД PostgreSQL, являются классическое реляционное проектирование и использова-

ние полей с типами нечетких данных. Поскольку различие между типами нечетких данных, основанных на массивах или различных сложных типах, существенно менее значительно, чем между этими двумя подходами, для получения оценки сравнения подходов достаточно ограничиться рассмотрением массивов в качестве основы типа нечетких данных. Выбор массивов обуславливается их большей, по сравнению со сложными типами, структурной простотой, в то же время достаточной для хранения функции принадлежности, а также наличием необходимых операторов и функций работы с ними. Эффективность работы с массивами поддерживается GIN-индексами. Заметим лишь, что среди сложных типов в настоящий момент целесообразно отдать предпочтение jsonb, определяемому как структурированные декомпозированные двоичные json-данные. Он дополняет гибкость и универсальность стандартных текстовых json-данных возможностью использования GIN-индексов в операторах поиска, включая оператор @@ соответствия в языке JQuery запросов к данным типа jsonb, предоставляющий широкие возможности как для поиска и обработки данных, так и для поддержания целостности.

Общим для обоих подходов к хранению и обработке нечетких данных является представление нечетких данных в БД. Нечеткое множество, определенное в ограниченном полном пространстве, описывается множеством пар «элемент множества–значение его функции принадлежности» и в таком виде хранится в БД. Пары со значениями функции принадлежности, равными 0, не хранятся, но при этом хранятся (однократно) все элементы ограниченного полного пространства, что необходимо как для поддержания целостности данных, так и для выполнения таких функций, как нечеткое отрицание. Нумерация элементов ограниченного полного пространства позволяет включать в пару со значением функции принадлежности вместо наименования значения (элемента множества) его номер. Кроме компактности хранения, это позволяет (при необходимости) отражать присущую многим множествам упорядоченность их элементов, включая нечеткие множества, определенные на лингвистических переменных, использующих не числовые, а словесные оценки. Такое хранение не усложняет нечеткие вычисления над элементами множеств, поскольку при них выполняется только проверка совпадения элементов. Для нумерации элементов используется наиболее компактный для необходимого количества значений упорядочиваемый тип данных, например, char при не более нескольких десятков различных значений.

Вычислительный эксперимент по сравнению подходов к хранению и обработке нечетких данных на основе классического реляционного проектирования и использования полей с типами нечетких данных, основанных на массивах для хранения

функции принадлежности, далее для краткости называемых вариант 1 и вариант 2, производился в схеме fuzzy БД, работающей под управлением СУБД PostgreSQL 9.5 с настройками по умолчанию, на системе с процессором Intel Core i5-4430 3.0 GHz и 16 GB оперативной памяти.

Полное пространство предметной области описывается в таблице complete_space атрибутами его идентификатора и именования, а элементы пространства – в таблице element с атрибутами ссылки на пространство, номера и именования элемента. Для варианта 1 нечеткие множества в таблице set описываются в идентичной элементам структуре, но по отношению к нечеткому множеству. Первичный ключ составляют ссылки на пространство и номер элемента или нечеткого множества соответственно. Для элементов и нечетких множеств не введены идентификаторы, поскольку это позволяет уменьшить количество полей ссылок на них за счет усложнения поддержания целостности. При этом в таблице adjective для функции принадлежности имеются атрибуты ссылок на пространство, нечеткое множество и элемент, составляющие первичный ключ, а также атрибут значения функции принадлежности. В случае многократного повторения одних и тех же значений функции принадлежности, например, как значений лингвистической переменной, может быть целесообразным добавление соответствующей таблицы с изменением внешних ссылок. При определенных условиях могут быть целесообразными и другие проектные решения в рамках этого подхода, но принципиально преимущества и недостатки варианта 1 по сравнению с вариантом 2 они не изменяют.

Для варианта 2 и функций, выполняющих основные операции (функции) для работы с нечеткими данными и принимающих в качестве параметра или возвращающих функции принадлежности, создан составной тип adj4 элемента функции принадлежности из номера элемента в полном пространстве и значения функции принадлежности, послуживший основой типа массива adj4[] функции принадлежности. Вместо таблиц set и adjective варианта 1 в варианте 2 создается одна таблица seta с теми же атрибутами ссылки на пространство, номера и именования нечеткого множества, а также с атрибутом функции принадлежности типа adj4[]. Невозможность использования доменов при создании типа adj4[] и невозможность поддержания ссылочной целостности при его использовании в таблице требует создания специальных триггерных механизмов поддержания целостности данных.

Для проведения вычислительного эксперимента создано полное пространство X предметной области с 9 элементами с номерами от 1 до 9. На нем сгенерированы 10 000 нечетких множеств с различным распределением функции принадлежности. Учитывая типичность оценочного характера нечетких данных и связанную с участием человека

их антропометричность, указанные количества достаточны для практической оценки эффективности рассматриваемых вариантов. Для первых 1 000 нечетких множеств задавалось равномерное распределение функции принадлежности, для остальных – нормальное с левым, средним и правым пиками вероятности в равных долях (по 3 000 нечетких множеств). Таблицы для обоих вариантов заполнены идентичными данными.

Введение типа `adj4[]` позволило создать на языке SQL, без применения процедурных языков типа PL/pgSQL, несколько десятков функций, выполняющих основные операции (функции) для работы с нечеткими данными и принимающих этот тип в качестве параметра и/или возвращающих его. Если на «разогревом» кэше при достаточно большом количестве (тысячах) выполняемых операций время выполнения таких встроенных в СУБД операторов, как возведение в степень или деление, составляет порядка 0,01 мс на операцию, то время выполнения таких созданных функций, как дефаззификация различными методами, редукция по мере α -сопоставления или импликация Лукасевича, составляет порядка 0,05 мс в аналогичных условиях. Этот результат говорит о пригодности разработанных функций для эффективной работы с нечеткими данными.

Данный вывод подтверждают выполненные примеры расчета классического нечеткого вывода при предположении, что полное пространство X предметной области является как полным пространством предпосылок, так и полным пространством заключений, а сгенерированные данные представляют собой базу экспертных данных. Считается заданным соответствие нормализованных нечетких множеств aa в предпосылке нормализованным нечетким множествам bb в заключении (упрощенно через соответствие их номеров). Необходимо по нечеткому множеству a_a в данных наблюдения получить результат вывода b_b . Для этого по максимальной мере α -сопоставления aa и a_a выполняется редукция bb , соответствующего aa максимальной меры α -сопоставления с a_a , с последующей дефаззификацией полученного нечеткого множества, в данном примере методом центра тяжести. При выполнении примеров расчетов нечеткое множество a_a задавалось случайной генерацией его номера. Время выполнения этих расчетов для варианта 1 не превышало 650 мс, а для варианта 2 – 450 мс. Поскольку редукция по мере α -сопоставления и дефаззификация выполняются над единственным нечетким множеством на порядки быстрее данного времени, практически все время выполнения этих расчетов затрачивается на вычисление максимальной меры α -сопоставления нечеткого множества a_a в данных наблюдения нормализованным нечетким множествам aa в предпосылке.

Следует заметить, что максимальная мера α -сопоставления является не единственным возможным критерием выбора наиболее совпадающего с заданным нечеткого множества, хотя и очень эффективным для нормализованных нечетких множеств. Альтернативным критерием может выступать минимальное расстояние между соответствующими нечетким множествам точками в многомерном пространстве, координатами которого являются элементы полного пространства со значениями функции принадлежности элемента множеству. Этот критерий, в отличие от меры α -сопоставления, учитывает значения функции принадлежности всех элементов, а не только максимально совпадающие у сопоставляемых множеств. К тому же он применим и к ненормализованным нечетким множествам, нормализация которых перед сравнением может приводить к неприемлемым семантическим искажениям. Например, он может использоваться для поиска predetermined значений лингвистических переменных, наиболее соответствующих задаваемым значениям функции принадлежности. Поскольку возведение в квадрат отрицательных чисел сохраняет упорядоченное отношение их линейного порядка, вместо расстояния между соответствующими нечетким множествам точками целесообразно использовать вычислительно более легкий квадрат расстояния, равный сумме квадратов разностей координат точек.

Повторно выполнены примеры расчета нечеткого вывода с тем отличием, что вместо определения максимальной меры α -сопоставления aa и a_a выполнялся поиск aa с наименьшим квадратом расстояния от a_a и мера их α -сопоставления использовалась для редукции соответствующего bb . Их результаты в основном совпали с выполненными ранее примерами расчета классического нечеткого вывода с очень редкими незначительными отклонениями. Среднее время выполнения этих расчетов для варианта 1 составило 750 мс, а для варианта 2 – 230 мс. То есть вариант 2 оказался существенно эффективнее варианта 1.

Для расчета нечеткого вывода, использующего поиск aa с наименьшим квадратом расстояния от a_a , в отличие от расчета классического нечеткого вывода с максимальной мерой α -сопоставления, удалось реализовать способ существенного уменьшения времени его выполнения, основанный на предварительной фильтрации данных с использованием опорных данных. Под опорными здесь понимаются данные, однократно вычисляемые независимо для каждой из записей и позволяющие исключить из дальнейшего, вычислительно существенно более сложного сопоставления записи по сравнению их опорных данных. Реализация этого способа в данном случае основывается на том факте, что близко расположенные в многомерном пространстве точки имеют незначительно различа-

ющиеся расстояния от них до любой фиксированной точки, в том числе до точки начала координат, расчет квадрата расстояния до которой наиболее эффективен, поскольку не требует учета нулевых значений функции принадлежности. Для каждой точки эта величина равна сумме квадратов ее координат. Она вычисляется в триггерах вставки или изменения данных и сохраняется в добавленном в таблицы *set* и *seta* атрибуте вещественного типа.

Чрезвычайно важна настройка фильтра, заключающаяся в задании допустимого отклонения суммы квадратов координат от заданной величины. С одной стороны, чем меньше допустимое отклонение, тем эффективнее фильтр. С другой – в результате фильтрации не должно быть исключено искомое решение, поскольку точка, ближайшая к заданной, может иметь расстояние от начала координат, значительно отличающееся от заданного, например, при расположении с заданной точкой и началом координат на одной прямой. При выполнении примеров расчета нечеткого вывода и установленном допустимом отклонении суммы квадратов координат $\pm 0,1$ от заданного ни одно искомое решение не было потеряно, что подтверждено сравнением с результатами расчета без фильтрации. Среднее время выполнения расчетов с фильтрацией составило 125 мс и 75 мс для вариантов 1 и 2 соответственно, что существенно лучше, чем по максимальной мере α -сопоставления и по квадрату расстояния без фильтрации для обоих вариантов.

Следующей по эффективности применения после точки начала координат является единичная точка (все координаты равны 1), квадрат расстояния до которой для любой точки равен сумме количества ее нулевых (незаданных) координат и квадратов единиц, уменьшенных на ее координаты. При большом количестве данных добавление к фильтрации по квадрату расстояния от начала координат фильтрации по квадрату расстояния от единичной точки может быть достаточно эффективным. Выбор других точек в качестве опорных в общем случае практически нецелесообразен. В то же время для значительно удаленных от начала координат точек эффективными опорными данными является направление на них из начала координат. На рисунке 1 для упрощенного примера двумерного пространства (люди по возрасту относятся только к молодым или старым) проиллюстрирована совместная фильтрация точек относительно изображенной звездочкой искомой точки по квадрату расстояния и направлению от начала координат.

Для наиболее распространенных распределений функции принадлежности в качестве одномерной и легко вычисляемой оценки направления от начала координат предлагается использовать номер элемента с максимальным значением функции принадлежности (максимальный номер в случае нескольких таких номеров). В качестве фильтра

при этом может использоваться, например, отклонение номера элемента от искомого не более чем на 1,5, как в выполненных примерах расчета нечеткого вывода. В результате среднее время выполнения расчетов с совместной фильтрацией по квадрату расстояния и направлению от начала координат еще более сократилось и составило 37 мс и 20 мс для вариантов 1 и 2 соответственно.

Полученные результаты являются следствием основного принципа построения эффективных запросов в БД: чем к меньшему количеству данных осуществляется обращение, то есть чем на более ранних стадиях выполнения запроса отсекается наибольшая часть ненужных данных, тем быстрее выполняется запрос. Этим же, наряду с упорядоченностью внутренних данных индексов, объясняется тот факт, что, начиная с определенного количества записей в таблицах, последовательное сканирование значительно уступает в быстродействии индексному сканированию, поскольку количество страниц памяти, занимаемое индексами, существенно меньше занятого таблицами. Максимально разница в быстродействии проявляется, когда появляется возможность, особенно в многоколоночных (составных) индексах, получения данных непосредственно из индекса без обращения к таблице, тем более при одновременном доступе и сортировке по ключам [9]. С целью еще большей эффективности использования составных индексов при выполнении SQL-запросов применяется совместная оптимизация логического выражения раздела выборки WHERE и списка выражений раздела сортировки ORDER BY команды запроса SELECT со списком выражений составного индекса [10].

Таким образом, дальнейшее повышение эффективности выполнения расчета нечеткого вывода достигается созданием так называемого опорного составного индекса, удовлетворяющего условиям фильтрации и сортировки по опорным данным. В результате среднее время выполнения примеров расчета нечеткого вывода с использованием опорного составного индекса составило 17 мс для варианта 1 и 11 мс для варианта 2, то есть практически в 40 и 20 раз меньше первоначального, что наглядно показано на рисунке 2.

Рис. 2. Сравнение вариантов расчета нечеткого вывода

Fig. 2. Comparison of fuzzy inference calculation options

В принципе, введение полей с опорными данными не является необходимым для применения опорных индексов, которые могут создаваться не по хранимым опорным данным, а по выражениям их вычисления. Однако при этом каждый раз при фильтрации с использованием индексов придется использовать эти выражения. Поэтому практически всегда, кроме тех редких случаев, когда команды UPDATE выполняются чаще, чем команды SELECT, рекомендуется создавать поля с опорными данными, вычисляемыми в триггерных функциях.

Наконец, следует отметить, что вычислительный эксперимент проводился над единственным нечетким множеством. При наличии нескольких нечетких множеств расчет нечеткого вывода для каждого из них выполняется независимо даже при их построении на едином полном пространстве предметной области. Зависимость времени выполнения одновременного расчета нечеткого вывода для нескольких нечетких множеств от рассмотренных вариантов подходов к хранению и обработке нечетких данных и способов дополнительной

фильтрации носит тот же характер, что и для расчета нечеткого вывода для единственного нечеткого множества. Различия во времени выполнения расчета увеличиваются при увеличении количества нечетких множеств, что было подтверждено отдельными вычислительными экспериментами с двумя и тремя нечеткими множествами, построенными на том же полном пространстве предметной области. Использование полей с типами нечетких данных, основанных на массивах для хранения функции принадлежности, существенно эффективнее классического реляционного проектирования. При этом фильтрация по опорным данным, особенно с применением опорных индексов, влияет на эффективность значительно, чем выбор варианта используемой модели нечетких данных.

Таким образом, несмотря на отсутствие в настоящий момент в СУБД PostgreSQL встроенных типов нечетких данных и, соответственно, процедур и функций работы с ними, возможны эффективные хранение и обработка в ней нечетких данных. Применение классического реляционного проектирования для нечетких данных имеет смысл только при небольшом количестве нечетких множеств с часто изменяемыми данными и необходимости простыми средствами поддерживать целостность данных. В остальных случаях целесообразно создание пользовательских типов нечетких данных на основе композитных типов, содержащих массив, или на основе сложных типов, таких как jsonb. Выделение для предварительной фильтрации опорных данных и создание опорных индексов позволяют резко повысить эффективность выполнения нечеткого вывода как одного из вычислительно сложных основных направлений использования нечетких данных.

Литература

1. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений; [пер. с англ. Н.И. Ринго; под ред. Н.Н. Моисеева и С.А. Орловского]. М.: Мир, 1976. 165 с.
2. Асаи К., Ватада Д., Иваи С. и др. Прикладные нечеткие системы; [пер. с япон.; под ред. Т. Тэрано, К. Асаи, М. Сугэно]. М.: Мир, 1993. 368 с.
3. Батыршин И.З. Основные операции нечеткой логики и их обобщения. Казань: Отечество, 2001. 100 с.
4. Аверкин А.Н., Костерев В.В. Триангулярные нормы в системах искусственного интеллекта // Изв. РАН: Теория и системы управления. 2000. № 5. С. 107–119.
5. Батыршин И.З., Недосекин А.О., Стецко А.А., Тарасов В.Б., Язенин А.В., Ярушкина Н.Г. Нечеткие гибридные системы. Теория и практика; [под ред. Н.Г. Ярушкиной]. М.: Физматлит, 2007. 208 с.
6. PostgreSQL 9.6.4 documentation. The PostgreSQL Global Development Group. URL: <https://www.postgresql.org/files/documentation/pdf/9.6/postgresql-9.6-A4.pdf> (дата обращения: 29.08.2017).
7. What's New In PostgreSQL 10. URL: https://wiki.postgresql.org/wiki/New_in_postgres_10 (дата обращения: 29.08.2017).
8. Сорокин В.Е. Об эффективности наследования таблиц в СУБД PostgreSQL // Программные продукты и системы. 2016. Т. 29. № 3. С. 15–23.

9. Фролков И. Оптимизация запросов в PostgreSQL. PgConf.Russia 2016. URL: <https://pgconf.ru/media/2016/02/19/Фролков.pdf> (дата обращения: 05.04.2016).

10. Сорокин В.Е. Метод искусственного соответствия SQL-запросов индексам реляционных баз данных // Программные продукты и системы. 2013. № 2. С. 47–54.

Software & Systems
DOI: 10.15827/0236-235X.120.609-618

Received 18.09.17
2017, vol. 30, no. 4, pp. 609–618

FUZZY DATA STORING AND EFFICIENT PROCESSING IN PostgreSQL DBMS

V.E. Sorokin¹, Ph.D. (Engineering), Leading Researcher, sorokinve@yandex.ru

¹ R&D Institute Centerprogramsystem, 50 let Oktyabrya Ave. 3a, Tver, 170024, Russian Federation

Abstract. In just half a century the theory of fuzzy sets has developed from theoretical foundations to practical use in artificial intelligence systems in such areas as industrial equipment and vehicles control, medical diagnosis and expert systems, including risk assessment, from economic to environmental. Many of them need efficient and reliable storage and processing of large volumes of information. For this purpose, there are DBMS, the most advanced of them are the object-relational DBMS. However, fuzzy data are difficult to reconcile with both object and relational data models. Moreover, the majority of industrial DBMS do not contain built-in fuzzy data types and mechanisms to work with them. These DBMS include cross-platform and freely distributed in source object-relational DBMS called PostgreSQL. Import substitution in infrastructure software is largely associated with application of PostgreSQL.

Opportunities of fuzzy data relational modeling and a developed mechanism for extending types through the creation of the required user-defined types along with powerful procedural languages in PostgreSQL allow implementing various alternative approaches to fuzzy data storage and handling.

The article shows a comparative analysis of these approaches from the point of view of maintaining data integrity and processing efficiency. There are the experimental results at various fuzzy data models depending on approaches. The paper proposes design solutions that increase the efficiency of fuzzy data search, and formulates recommendations on choosing an approach to modeling fuzzy data.

Keywords: linguistic variable, fuzzy set, database management system, PostgreSQL, triangular norms, fuzzy logic operations, data integrity, SQL query efficiency.

References

1. Zadeh L.A. *The Concept of a Linguistic Variable and Its Application to Approximate Reasoning*. Part 1, 2, 3. Information Sciences. 1975, no. 8, pp. 199–249, no. 8, pp. 301–357, no. 9, pp. 43–80 (Russ. ed.: Ringo N.I. Moscow, Mir Publ., 1976, 165 p.).
2. Asai K., Vatada D., Ivai S. *Applied Fuzzy Systems*. T. Tehrano, K. Asai, M. Sugeno (Eds.). (Russ. ed.: Moscow, Mir Publ., 1993, 368 p.).
3. Batyrshin I.Z. *Osnovnye operatsii nechetkoy logiki i ikh obobshcheniya* [Basic Operations of Fuzzy Logic and Their Generalizations]. Kazan, Otechestvo Publ., 2001, 100 p.
4. Averkin A.N., Kosterev V.V. Triangular norms in artificial intelligence systems. *Izvestiya RAN: Teoriya i sistemy upravleniya* [Proc. of RAS: Theory and Control Systems]. 2000, no. 5, pp. 107–119 (in Russ.).
5. Batyrshin I.Z., Nedosekin A.O., Stecko A.A., Tarasov V.B., Yazenin A.V., Yarushkina N.G. *Nechetkie gibridnye sistemy. Teoriya i praktika* [Fuzzy Hybrid Systems. Theory and Practice]. N.G. Yarushkina (Ed.). Moscow, Fizmatlit Publ., 2007, 208 p.
6. *PostgreSQL 9.6.4 Documentation*. The PostgreSQL Global Development Group Publ., 1996–2017. Available at: <https://www.postgresql.org/files/documentation/pdf/9.6/postgresql-9.6-A4.pdf> (accessed August 29, 2017).
7. *What's New In PostgreSQL 10*. Available at: https://wiki.postgresql.org/wiki/New_in_postgres_10 (accessed August 29, 2017).
8. Sorokin V.E. About table inheritance effectiveness in PostgreSQL DBMS. *Programmnye produkty i sistemy* [Software and Systems]. 2016, no. 3 (29), pp. 15–23 (in Russ.).
9. Frolov I. *Optimizatsiya zaprosov v PostgreSQL* [PostgreSQL Query Optimization]. PgConf.Russia 2016. Available at: <https://pgconf.ru/media/2016/02/19/%D0%A4%D1%80%D0%BE%D0%BB%D0%BA%D0%BE%D0%B2.pdf> (accessed 05 April 2016).
10. Sorokin V.E. A method of artificial matching of sql query to relational databases indexes. *Programmnye produkty i sistemy* [Software and Systems]. 2013, no. 2, pp. 47–54 (in Russ.).