Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка программного обеспечения для работы с базой данных нечетких экспертных систем

Студент группы **ИУ7-65Б** _____ (Подпись, дата) **М.Н. Серова** (И.О. Фамилия)

Руководитель _____ (Подпись, дата) **Ю.В. Строганов** (И.О. Фамилия)

Консультант _____ (Подпись, дата) **О.В. Яковлева** (И.О. Фамилия)

***2022** г.*

РЕФЕРАТ

Расчетно-пояснительная записка 53 с., 22 рис., 1 табл., 22 ист., 1 прил.

В работе представлена разработка программного обеспечения для работы с базой данных нечетких экспертных систем.

Ключевые слова: нечеткая логика, экспертные системы, базы данных, Мамдани, Сугено.

Проведен анализов существующих подходов к хранению информации о нечетких системах. Разработана база данных экспертных систем типа Мамдани и Сугено, приведены алгоритмы для получения результата логического вывода. Реализована программа на языке Matlab для генерации экспертных систем на основе файлов с примерами входных и выходных данных, описан интерфейс приложения для взаимодействия с разработанной БД. Проведено исследование результатов работы системы и времени выполнения запросов для получения логического вывода с кэшированием и без.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные системы автоматического управления широко используют методы нечеткой логики. Основная идея экспертных систем заключается в имитации процесса мыслительной деятельности человека и принятия решений на основе некоторых входных данных. Аппарат теории нечетких множеств позволяет моделировать плавный переход от одного множества к другому при определении степени принадлежности числовых параметров, выражать степень уверенности в процессе принятия решений [1].

Впервые нечеткие множества были описаны Л. Заде в работе «Fuzzy sets», но наибольшее распространение нечеткая логика получила после доказательства в 1993г. Бартоломеем Коско FAT-теоремы (Fuzzy Approximation Theorem), согласно которой любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике [2].

Нечеткие экспертные системы могут состоять из большого количества правил, например, система MYCIN, спроектированная для диагностирования тяжелых бактериальных инфекций и рекомендации лекарств, оперирует базой знаний из 600 правил [3]. В связи с этим появляется необходимость хранения информации в базе данных.

Цель: Разработать базу данных нечетких экспертных систем и программное обеспечение для работы с ней.

Задачи:

1. Провести анализ существующих подходов к хранени..
2. Спроектировать базу данных экспертных систем.
3. Разработать программное обеспечение для работы с базой данных.
4. Исследовать скорость обработки запросов к базе данных при кэшировании и без.

1 Аналитическая часть

1.1 Нечеткая логика

1.1.1 Нечеткие множества

Характеристическая функция, описывающая принадлежность элемента четкому множеству, выглядит следующим образом:

$$\mu_A^*(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases} \tag{1.1}$$

Графическая форма принадлежности x четкому множеству A представлено на рисунке 1.1.

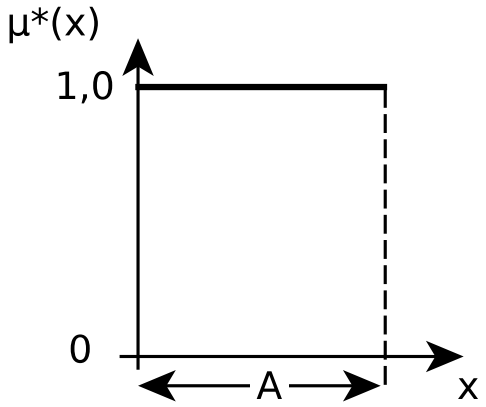


Рисунок 1.1 – Принадлежность элемента четкому множеству

В теории нечетких множеств $\mu_A(x)$ – одномерная функция принадлежности, если область значений одномерного отображения $\mu_A(x) \in [0, 1]$, пример ее графической формы представлен на рисунке 1.2.

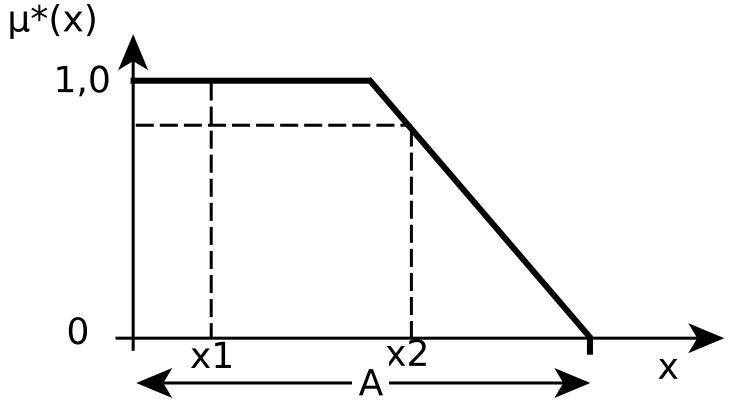


Рисунок 1.2 – Принадлежность элемента нечеткому множеству

Наиболее часто используемыми функциями принадлежности являются следующие:

– трапециевидная:

$$\mu(x;a,b,c,d) = \begin{cases} 0, x \leqslant a, \\ \dfrac{x-a}{b-a}, a < x \leqslant b, \\ 1, b < x \leqslant c, \\ \dfrac{d-x}{d-c}, c < x \leqslant d, \\ 0, d < x; \end{cases} \tag{1.2}$$

– треугольная:

$$\mu(x;a,b,c) = \begin{cases} 0, x \leqslant a, \\ \dfrac{x-a}{b-a}, a < x \leqslant b, \\ \dfrac{c-x}{c-b}, b < x \leqslant c, \\ 0, c < x; \end{cases} \tag{1.3}$$

– S-функция:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 0, x \leqslant \alpha, \\ 2 \cdot \left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2, \alpha < x \leqslant \beta, \\ 1 - 2 \cdot \left(\dfrac{\gamma-x}{\gamma-\alpha}\right)^2, \beta < x \leqslant \gamma, \\ 1, \gamma < x; \end{cases} \tag{1.4}$$

– П-функция:

$$\Pi(x;\beta,\gamma) = \begin{cases} S(x;\gamma-\beta,\gamma-\dfrac{\beta}{2},\gamma), x \leqslant \gamma, \\ 1 - S(x;\gamma,\gamma+\dfrac{\beta}{2},\gamma+\beta), \gamma < x; \end{cases} \tag{1.5}$$

– лингвистическая функция (результат применения лингвистического ба-

рьера («Очень», «Не», «Более или менее», прочее) к функции принадлежности множества), например:

– Очень А:

$$\mu_{VeryA}(x) = CON(A) = \mu_A(x)^2; \tag{1.6}$$

– Более или менее А:

$$\mu_{MoreorLessA}(x) = DIL(A) = \mu_A(x)^{0.5}; \tag{1.7}$$

– Не А:

$$\mu_{NotA}(x) = 1 - \mu_A(x); \tag{1.8}$$

– Не Очень А:

$$\mu_{NotVeryA}(x) = 1 - CON(A). \tag{1.9}$$

Нечеткой переменной называется тройка

$$\{X, U, \widetilde{A}\}, \tag{1.10}$$

где X – вербальное название переменной, U – область ее определения (универсальное множество), $\widetilde{A}$ – нечеткое множество универсального множества, описывающее возможные значения нечеткой переменной [1].

Основные операции над нечеткими множествами [4]–[5]:

– объединение:

$$A \cup B : \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)); \tag{1.11}$$

– пересечение:

$$A \cap B : \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)); \tag{1.12}$$

– отрицание:

$$\bar{A} : \mu_{\bar{A}}(x) = 1 - \mu_A(x); \tag{1.13}$$

– концентрация:

$$CON(A) : \mu_{CON(A)}(x) = (\mu_A(x))^2; \tag{1.14}$$

– растворение:

$$DIL(A) : \mu_{DIL(A)}(x) = (\mu_A(x))^{0.5}. \tag{1.15}$$

1.1.2 Лингвистические переменные

Для количественной оценки смысла предложений естественного языка применяются нечеткие множества и лингвистические переменные, после чего появляется возможность манипулировать этими предложениями. Лингвистическим переменным присваиваются значения, представляющие собой такие выражения, как слова, фразы или предложения естественного или искусственного языка.

Лингвистическая переменная определяется следующим образом:

$$\{X, T(X), U, V, S\}, \tag{1.16}$$

где X – вербальное название переменной, $T(X) = \{X_i, i = \overline{1, m}\}$ – терм-множество переменной X, т. е. множество термов, или названий лингвистических значений (каждое из этих значений – нечеткая переменная со значениями из универсального множества U), V – синтаксическое правило, которое ставит в соответствие каждой нечеткой переменной с названием из $T(X)$ нечеткое подмножество универсального множества U, S – семантическое правило, которое ставит в соответствие каждой нечеткой переменной с названием из $T(X)$ нечеткое подмножество универсального множества U [1].

1.1.3 Экспертные системы

Общая схема работы нечетких экспертных систем представлена на рисунках 1.3, 1.4.

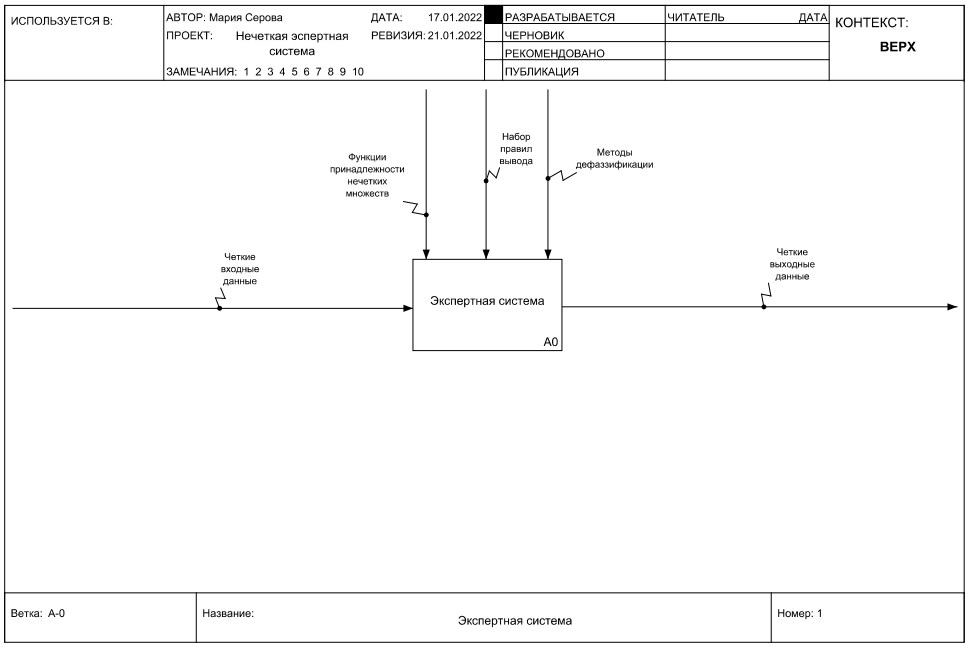


Рисунок 1.3 – Схема работы экспертной системы

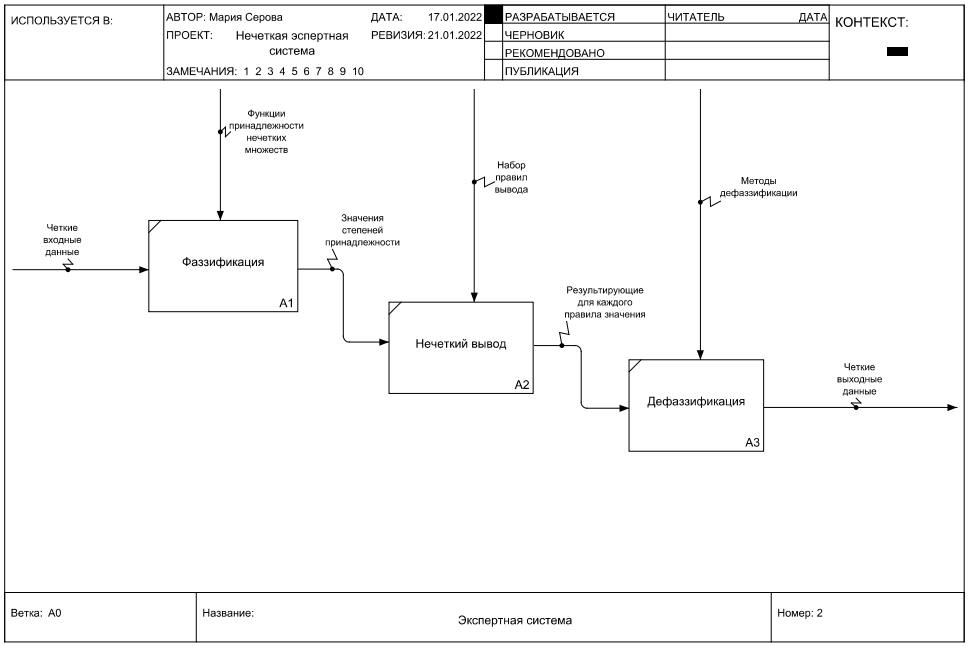


Рисунок 1.4 – Этапы работы экспертной системы

Работа любой нечеткой экспертной системы состоит из трех основных этапов [6]:

– фаззификация: четкие значения входных переменных x приводятся к нечетким значениям посредством вычисления их степени истинности выражения: x_u есть A_u^j как вычисления значения соответствующего значе-

ния входной функции принадлежности $\mu_A(x_u)$, где u – номер входной переменной, j – номер правила нечеткого логического вывода.

– нечеткий вывод: используя полученные на первом этапе значения степеней истинности, вычисляются результирующие значения для каждого j-го правила, вид которых определяется используемым механизмом нечеткого вывода, из набора в m правил.
– дефаззификация: получение четкого значения.

Наиболее распространенные механизмы нечеткого логического вывода, которые легли в основу соответствующих типов экспертных систем [6]–[7]:

– метод Мамдани:

$$if(x_1 \; is \; A_1^j) \; ... \; and \; (x_n \; is \; A_n^j) \; then \; y \; is \; B_k^j, \tag{1.17}$$

где and – операция пересечения нечетких множеств, k – номер функции принадлежности.

– метод Такаги–Сугено:

$$if(x_1 \; is \; A_1^j) \; ... \; and \; (x_n \; is \; A_n^j) \; then \; y = f(x_1, x_2, ..., x_n) = p_0 + \sum_{i=1}^{n} p_i x_i. \tag{1.18}$$

Существует несколько методов для определения истинности консеквентов правила (дефаззификации) [4]–[5]:

– метод максимума, при котором осуществляется выбор элемента с максимальной степенью принадлежности;
– метод моментов, при котором значения истинности консеквентам правил присваиваются с применением способа, аналогичному аналогичного способу вычисления первого момента инерции материального объекта в физике.
– метод центра тяжести;
– метод среднего максимума;
– метод центра области.

1.2 Существующие решения

При работе с нечеткими экспертными системами необходимо хранить переменные, функции принадлежности и продукционные правила. Можно выделить 2 подхода к хранению лингвистических переменных, функций принадлеж-

ности и продукционных правил:

– в оперативной памяти;
– в базе данных.

Первый подход применим только для небольших стабильных систем, поскольку увеличение/изменение продукционных правил может привести к значительному изменению кода и ухудшению его восприятия, а увеличение количества лингвистических переменных и функций принадлежности будет ограничено размером оперативной памяти. В связи с этим для работы с крупными экспертными системами прелоагается использовать базу данных.

В существующих решениях предлагается хранить в базах данных только лингвистические переменные и их функции принадлежности. Одни авторы [8] предлагают хранить функции в виде точек (получаемая таким образом ломаная линия позволяет задавать функции любого вида), что не позволяет учитывать такие функции, как лингвистические, и требует ручного ввода точек. Другие авторы [9] отмечают возможность хранения информации о нечетких множествах в виде xml/json полей в таблице, однако в таком случае существует риск нарушения целостности базы данных из-за невозможности автоматической проверки, также подобный подход усложняет процесс работы с информацией.

В работе [10] предлагается хранить нечеткие числа в виде кортежей вида 1.19.

$$< PK, FK_1, FK_2, ..., FK_n, Data_1, Data_2, ..., Data_m, T, Max, x1, x2, x3, x4 >, \tag{1.19}$$

где $PK, FK_1, FK_2, ..., FK_n$ – первичные и внешние ключи базы данных, $Data_1$, $Data_2$, ..., $Data_m$ – атрибуты отношения (четкие данные), T – тип функции принадлежности, Max – максимальное значение функции принадлежности, x_1, x_2, x_3, x_4 – аргументы функции принадлежности.

Данный подход не позволяет учитывать лингвистические и зависящие от других переменных функции. Также объединение нечетких переменных с функциями принадлежности, задающими нечеткое множество, приводит к дублированию информации.

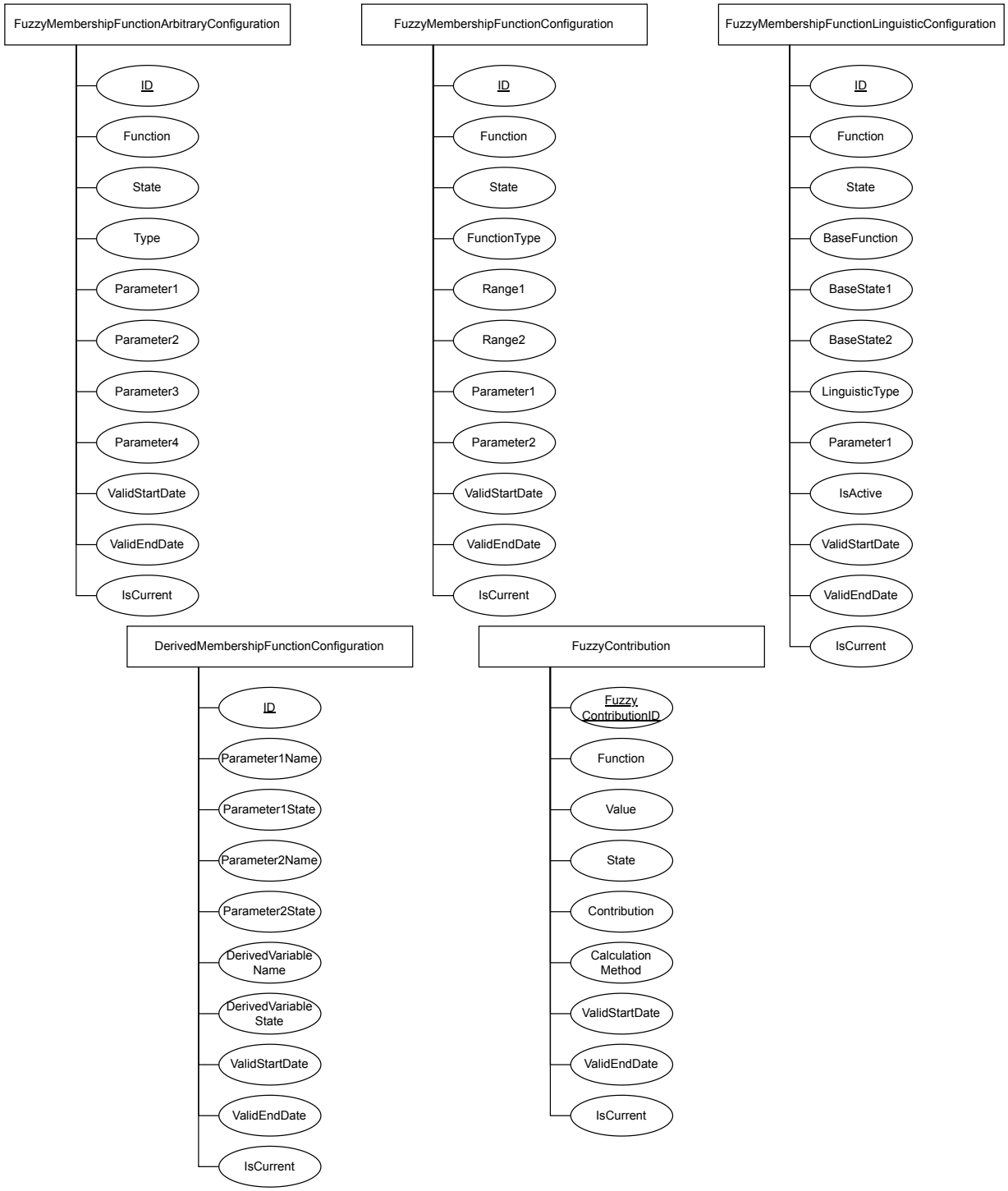


Рисунок 1.5 – Сущности, предложенные Динешем Асанка и Амалем Перера

Предложенные в работе [11] сущности базы данных представлены на рисунке 1.5. Авторы предлагают использовать отдельные таблицы для хранения информации о разных видах функций принадлежности, таких как обычная, формируемая на основе данных, дифференциальная и лингвистическая. При этом данная система за счет полей ValidStartDate и ValidEndDate позволяет хранить историю изменений. Обычная функция принадлежности может иметь до

четырех параметров, поскольку среди наиболее распространенных ее видов (треугольная, П-функция, проч.) максимальное количество аргументов (четыре) принимает трапециевидная. Для увеличения эффективности за счет однократного вычисления значения функции принадлежности для конкретной переменной предложена сущность FuzzyContribution, которая связывает лингвистическую переменную и функцию принадлежности.

Предложенная схема имеет следующий недостаток: хранение информации о функциях принадлежности в разных таблицах не позволяет ссылаться в одном поле переменной на необходимую функцию. Следовательно, возникает необходимость объединения различных параметров функций в одну таблицу.

Авторы работы [12] предлагают создавать отдельные таблицы для каждой возможной функции принадлежности, в которых содержатся только необходимые для вычисления параметры. При этом переменная, относительно которой происходит вычисление функции принадлежности (например, лингвистической) хранится в таблице FuzzyValue (нечеткая переменная), как и тип используемой функции. Предлагаемая схема представлена на рисунке 1.6.

Данная схема имеет несколько недостатков:

- числовое значение нечеткой переменной является ее идентификатором, из-за этого отсутствует возможность разделять переменные каким-либо образом или создавать разные переменные с одинаковым значением;
- отсутствие лингвистических переменных не позволяет учитывать возможность анализа принадлежности значения нескольким нечетким множествам;
- сложность поиска данных: таблица, к которой должно производиться обращение, зависит от описанного в поле типа;
- значение функции принадлежности должно вычисляться каждый раз при обращении к данным.

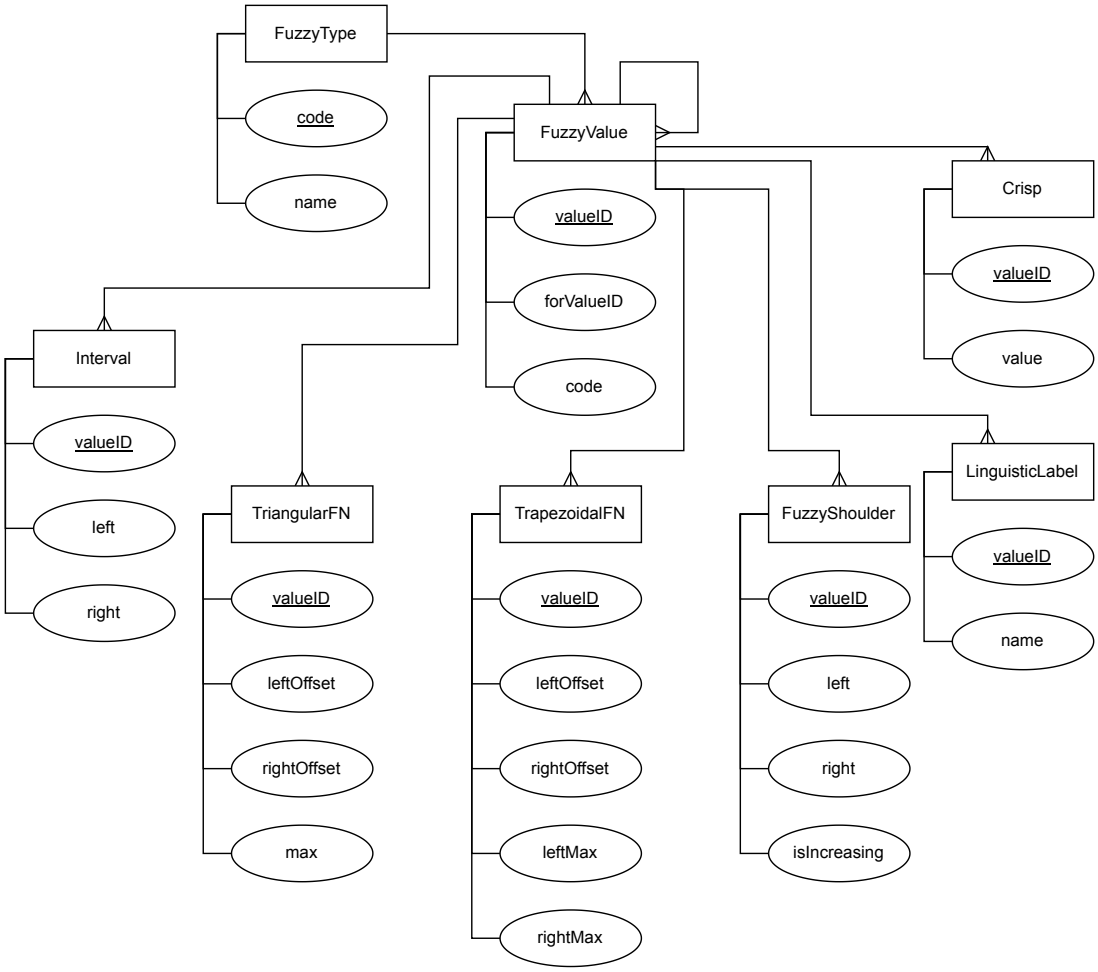


Рисунок 1.6 – Модель базы данных, предложенная Срджаном Шкрибич и Милошем Ракович

1.3 Базы данных и системы управления базами данных

Для хранения больших объемов информации о нечетких экспертных системах используются базы данных. Для управления базами данных существуют системы управления базами данных (СУБД) – совокупность программных, языковых и прочих средств, предназначенных для создания, управления, контролирования, администрирования и совместного использования базы данных разными пользователями [13].

1.3.1 Классификация баз данных

Все системы управления базами данных по модели хранения делятся на 3 типа, представленных ниже.

1. Дореляционные [14]:
    1) Основанные на иерархической модели – инвертированной древообразной структуры, в которой все узлы связываются друг с другом указателями.

Преимущества подобного хранения данных заключаются в простоте понимания структуры, целостности, независимости и безопасности данных. Недостатками являются ограничения в отношениях между сущностями (невозможно создать отношения типа многие ко многим), структурная зависимость (данные также хранятся в виде дерева, при серьезных изменениях существует вероятность потери возможности навигации по данным) и сложность разработки прикладного ПО.

2) Основанные на сетевой модели – графовая структура.
Данная модель позволяет назначать неограниченное количество связей между узлами графа и не требует переподчинения дочерних узлов при удалении узла, однако большое количество связей повышает сложности схемы и усложняет обеспечение целостности данных и разработку ПО.

2. Реляционные. Данные хранятся в структурированном виде в таблицах, которые могут быть связаны с другими таблицами через внешние ключи [15]. Такой подход требует тщательного анализа и выделения сущностей и связей между ними. Главными достоинствами реляционной модели являются целостность данных и соблюдение принципов ACID (атомарность, надежность, изолированность и долговечность) [16], однако в процессе нормализации с целью устранения избыточности данных появляется много таблиц, соединение которых для получения информации требует больших временных затрат [14]. Для работы с реляционными базами данных используется язык структурированных запросов SQL.
3. Постреляционные (к таким СУБД относятся нереляционные) [14]–[16]. Позволяют хранить неструктурированные данные, не имеют общего формата. К нереляционным относятся хранилища типа «ключ-значение» и базы данных, основанные на документ-ориентированной, графовой или столбцовой моделях. Основные преимущества данных СУБД заключаются в отсутствии необходимости предварительного структурирования данных, низких затратах на сборку в единое целое и простоте масштабирования, однако за это приходится платить ограниченностью синтаксических конструкций, разными языками для взаимодействия с каждой отдельной БД и возможностью потерей данных из-за ошибок.

1.3.2 Выбор базы данных

В результате сравнения моделей хранения данных принято решение об использовании реляционной БД по следующим причинам:

- гарантия целостности данных;
- сохранность данных благодаря соблюдению принципов ACID;
- универсальный язык запросов SQL.

1.4 Выводы

В данном разделе приведен аппарат нечеткой логики, описан механизм работы нечетких экспертных систем. Проведен анализ существующих решений проблемы хранения нечетких значений в базе данных. Приведена классификация баз данных по модели хранения, обоснован выбор реляционной БД.

2 Конструкторская часть

2.1 Постановка задачи

Решаемая задача формулируется следующим образом: необходимо создать программное обеспечение, позволяющее работать с нечеткими экспертными системами типа Мамдани и Сугено. Обычный пользователь может выбирать конкретную систему и получать результат при конкретных входных данных. Эксперты в зависимости от своей специализации дополнительно обладают возможностью добавлять, редактировать и удалять лингвистические переменные, функции принадлежности, правила вывода некоторых экспертных систем. Администраторы могут добавлять, изменять и удалять экспертные системы, лингвистические переменные, функции принадлежности и правила, изменять права доступа пользователей и экспертов.

В общем виде задача разрабатываемого ПО представлена на рисунке 2.1 и заключается в получении результата работы нечеткой экспертной системы с определенными параметрами на основе входных данных.

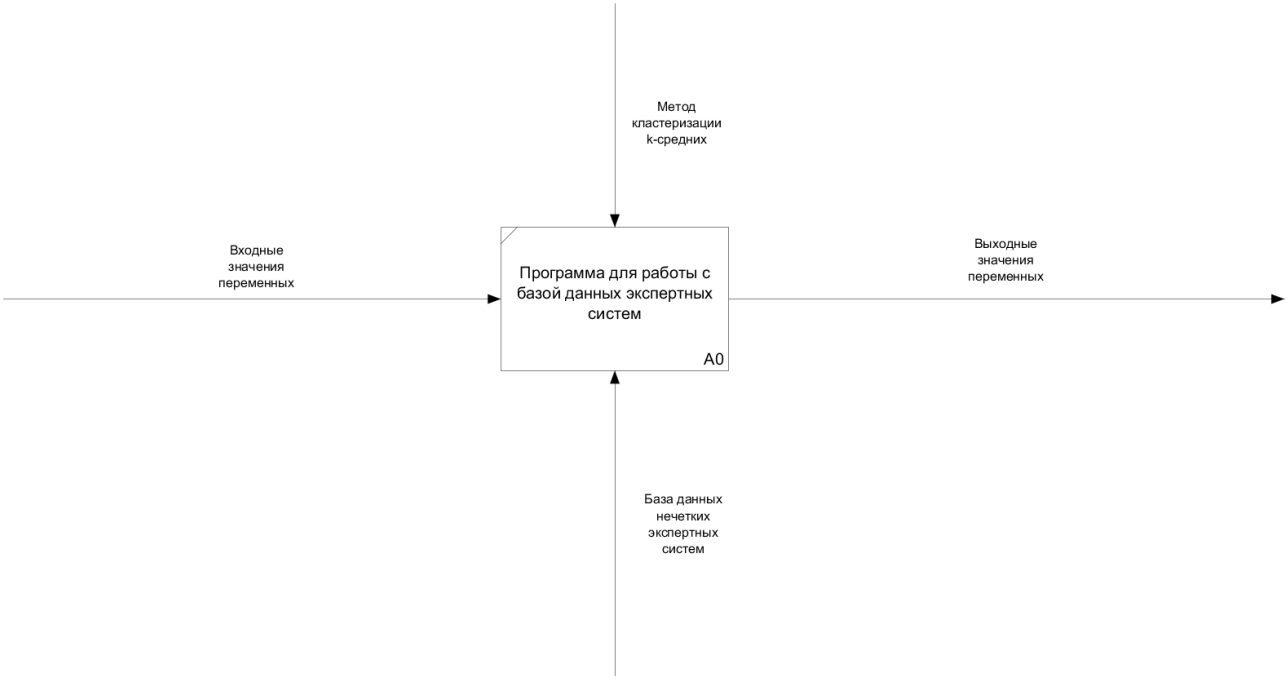


Рисунок 2.1 – Постановка задачи

Диаграмма прецедентов для различных пользователей представлена на рисунке 2.2.

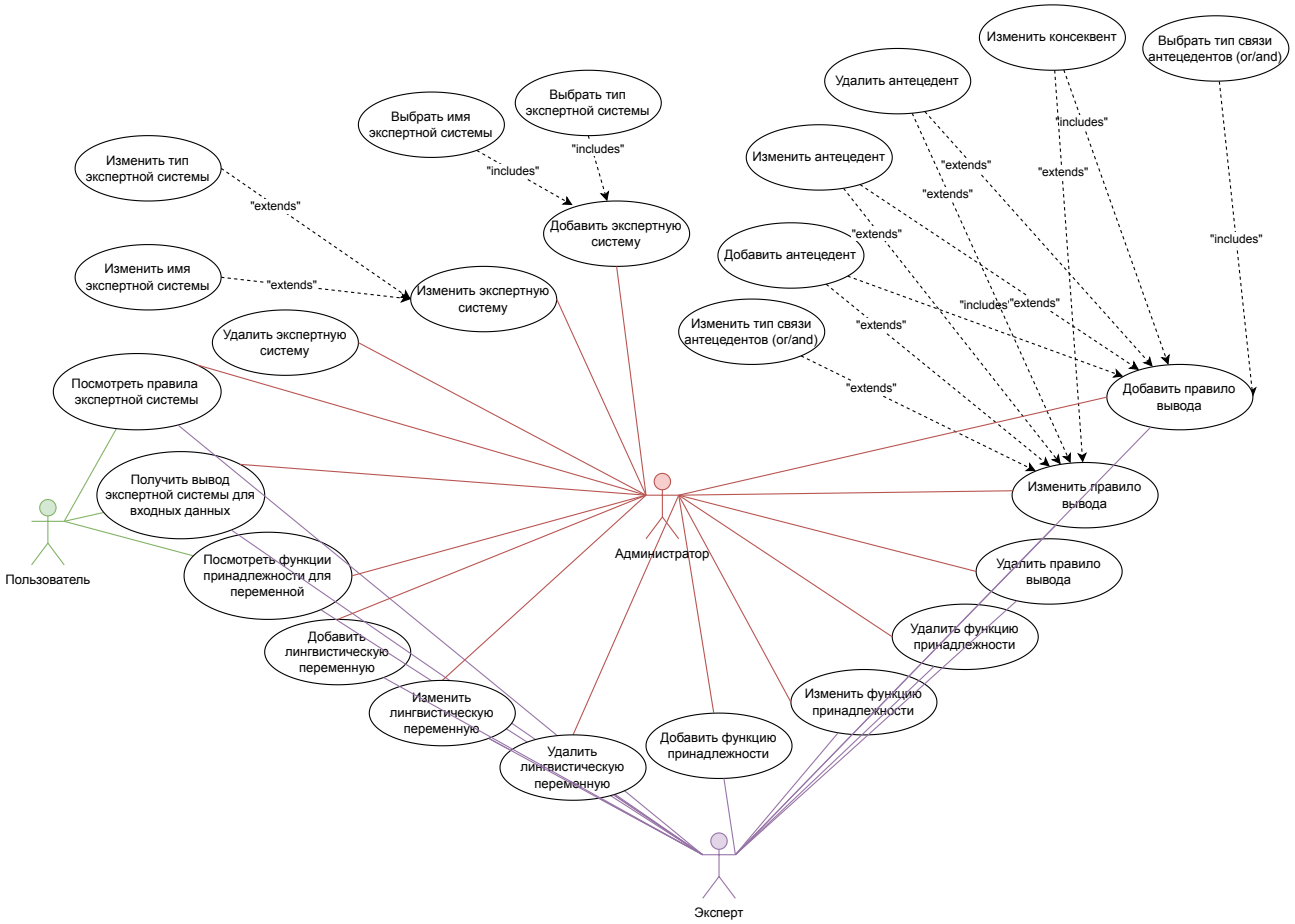


Рисунок 2.2 – Диаграмма вариантов использования разрабатываемого продукта

2.2 Представление в базе данных

Для реализации базы данных, хранящей информацию о нечетких экспертных системах, необходимо выделить сущности. Диаграмма «сущность–связь» разрабатываемой базы данных представлена на рисунке 2.3.

Таблица system отвечает за хранение следующей информации о нечеткой экспертной системе:

– s_id – уникальный идентификатор системы в базе данных;
– s_name – название нечеткой экспертной системы;
– s_type – тип нечеткой экспертной системы (Мамдани или Сугено);
– specialization – специализация нечеткой экспертной систем (физика, химия, информатика) для разделения прав доступа экспертов.

Таблица variable хранит информацию о переменных, использующихся в экспертных системах, и имеет следующие поля:

– v_id – уникальный идентификатор переменной в базе данных;
– v_name – название переменной;

– min_value – минимальное значение, принимаемое переменной;
– max_value – максимальное значение, принимаемое переменной;
– v_value – текущее значение переменной;
– s_id – внешний ключ для таблицы System, определяет, в какой системе используется переменная.

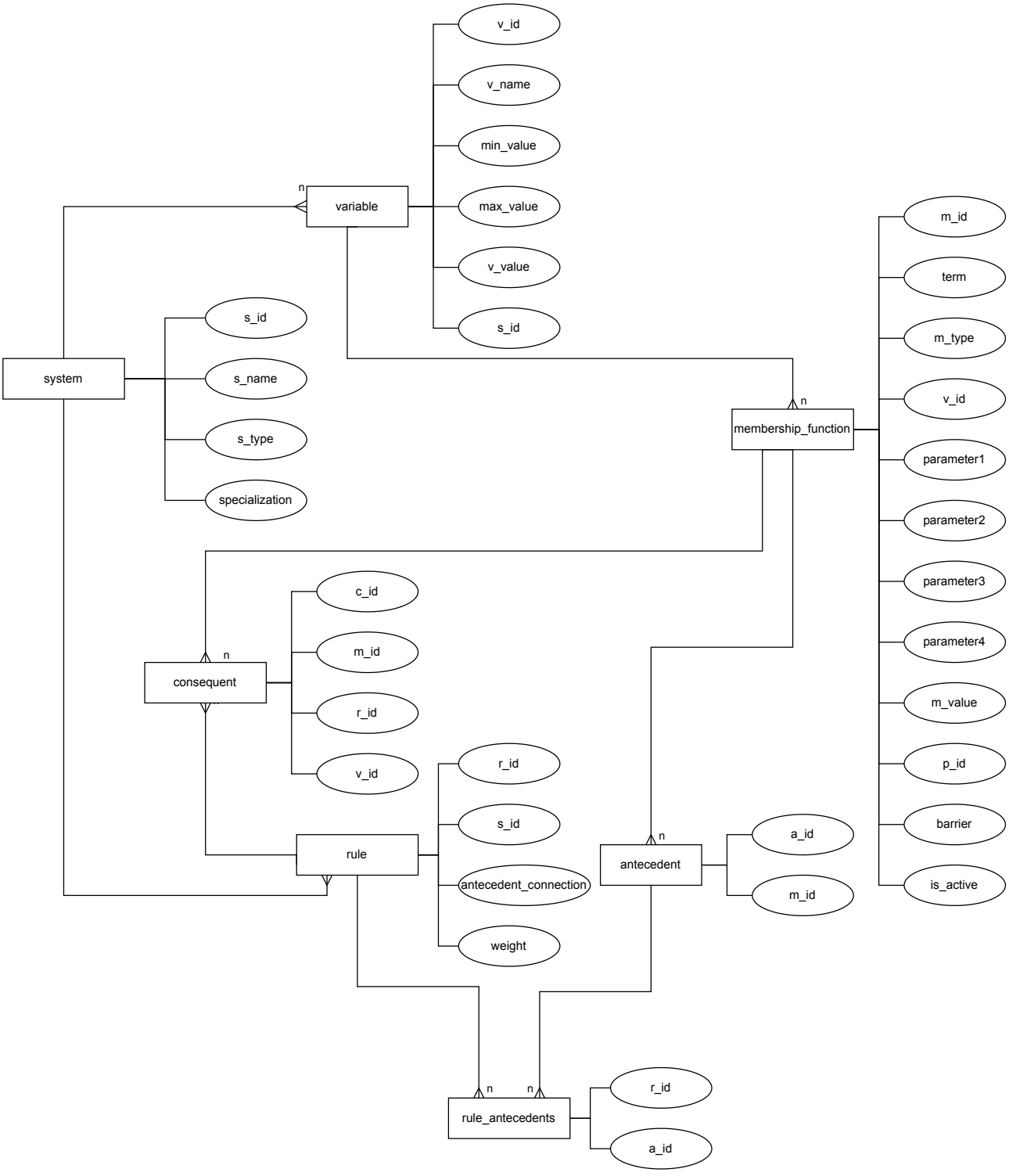


Рисунок 2.3 – Диаграмма сущность-связь

Таблица membership_function хранит информацию о функциях принадлежности нечетких множеств. Структура таблицы описана ниже.

- m_id – уникальный идентификатор функции принадлежности в базе данных;
- term – лингвистический терм, связанный с функцией принадлежности;
- m_type – вид функции принадлежности: треугольная, трапециевидная, лингвистическая, S-функция, П-функция, линейная или числовая (последние две используются для выражения консеквентов правил нечетких экспертных систем типа Сугено);
- v_id – связанная с функцией принадлежности переменная;
- parameter1, ..., parameter4 – параметры функции принадлежности;
- m_value – текущая степень принадлежности переменной множеству;
- p_id – внешний ключ для таблицы membership_function, используется для лингвистических функций, определяет, к какому множеству применяется лингвистический барьер;
- barrier – лингвистический барьер;
- is_active – флаг, определяющий, используется ли функция для правил вывода в данный момент.

Для хранения правил вывода используются таблицы rule, consequent, antecedent и rule_antecedents. У каждого правила может быть несколько антецедентов, при этом один и тот же антецедент может входить в несколько правил.

В таблице rule хранится следующая информация о правилах вывода:

- r_id – уникальный идентификатор правила в базе данных;
- s_id – внешний ключ для таблицы system, описывает, к какой системе относится правило вывода;
- antecedent_connection – вид соединения антецедентов («и» или «или»);
- weight – вес правила.

Таблица antecedent хранит информацию об антецедентах правил и имеет представленную ниже структуру.

- a_id – уникальный идентификатор антецедента в системе;
- m_id – внешний ключ для таблицы membership_function, которая задает условие «если x есть Y», где x – переменная, Y – нечеткое множество.

Таблица rule_antecendents связывает антецеденты с правилами и состоит только из внешних ссылок на соответствующие таблицы.

Таблица consequent описывает консеквенты правил и имеет следующие поля:

- c_id – уникальный идентификатор консеквента в базе данных;
- m_id – связанная с консеквентом функция принадлежности (для систем типа Мамдани описывает выражение «x есть Y», где x – переменная, Y – нечеткое множество; для систем типа Сугено – отдельное слагаемое для линейной функции принадлежности выходной переменной);
- r_id – внешний ключ для таблицы rule, описывает, частью какого правила является консеквент;
- v_id – внешний ключ для таблицы variable, для правил типа Сугено описывает выходную переменную.

2.3 Разработка алгоритмов

Из-за специфики баз данных необходимо определить хранимые процедуры и триггеры для полноценной работы экспертных систем. Для этого понадобятся следующие функции:

1. Вычисление значений функций принадлежности каждого из типов (треугольной, трапециевидной, s-функции, П-функции, лингвистической) при обновлении значения связанной с ними лингвистической переменной (триггер).
2. Вычисление результата работы нечеткой экспертной системы (функции):
    1) типа Сугено:
        а) вычисление антецедентов;
        б) вычисление результата;
    2) типа Мамдани:
        а) вычисление степеней принадлежности выходных переменных нечетким множествам;
        б) вычисление центров нечетких множеств при некотором значении функции принадлежности;
        в) дефаззификация методом центра тяжести для дискретных множеств.

Функции вычисления значений функций принадлежности высчитывают значение формулам 1.2–1.9. Схемы алгоритма получения результата нечеткой экспертной системы приведены на рисунках 2.4 и 2.5.

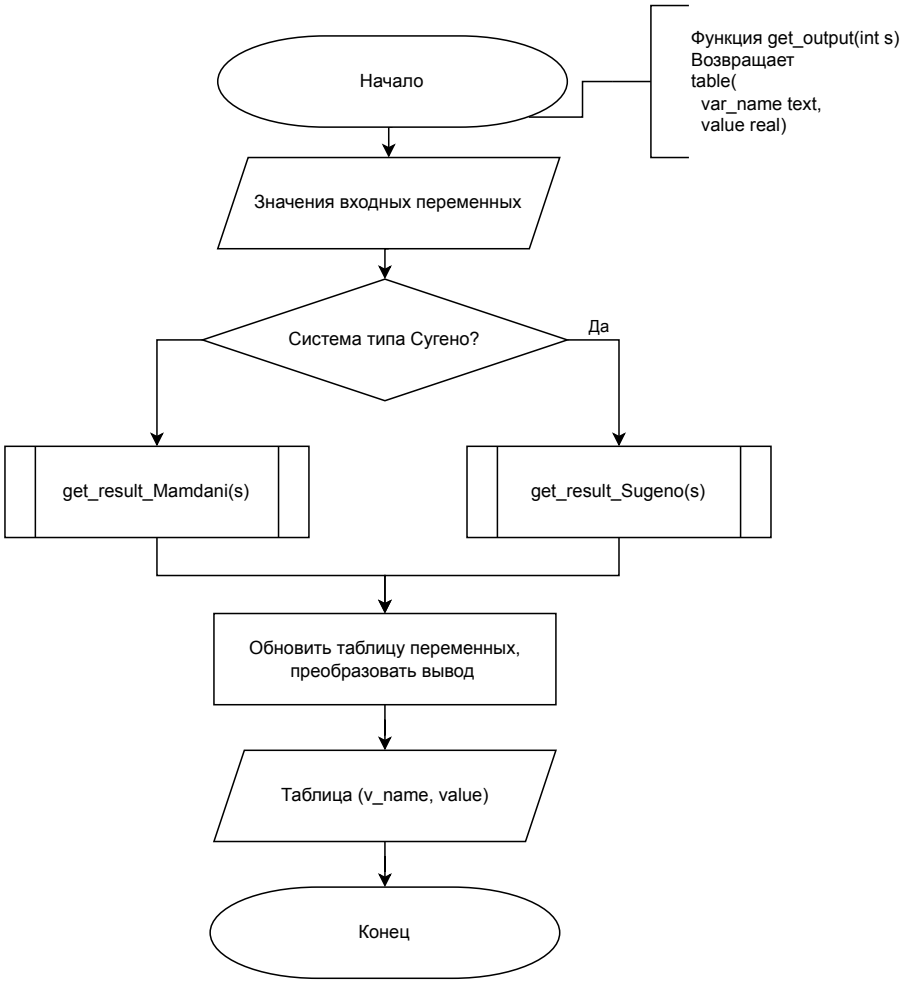


Рисунок 2.4 – Алгоритм получения результата работы нечеткой экспертной системы в базе данных

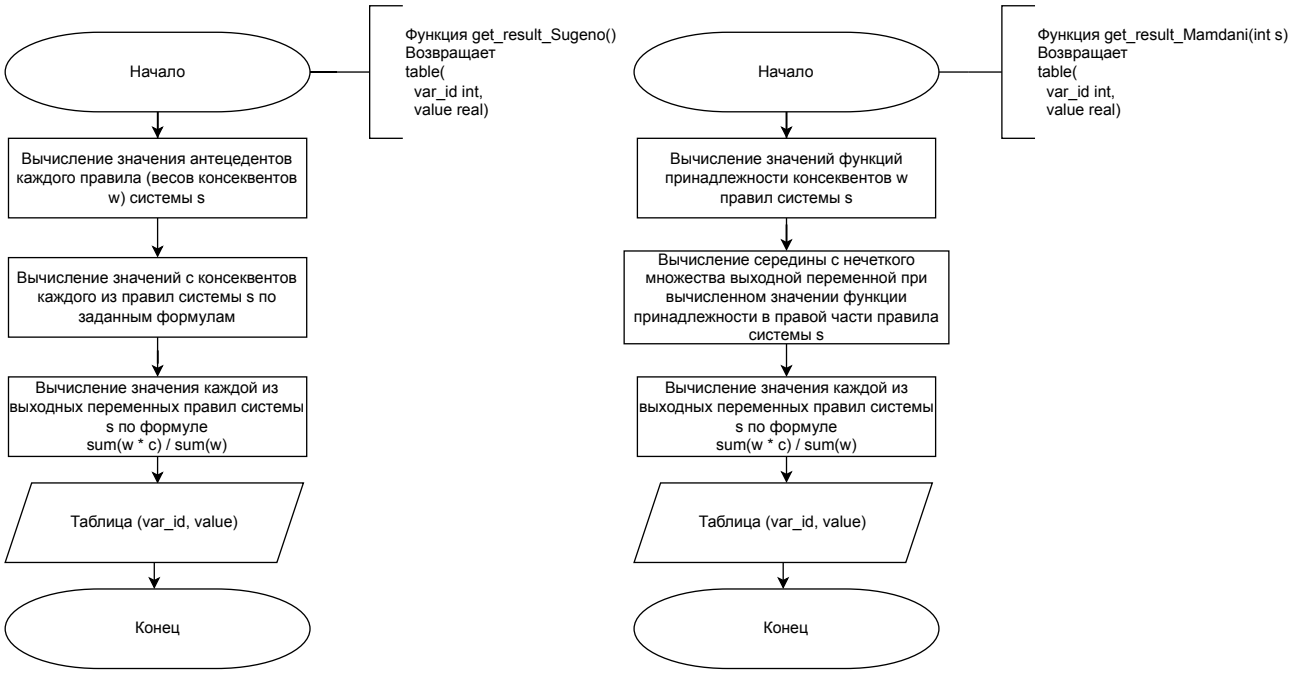


Рисунок 2.5 – Схема функций get_result_Sugeno и get_result_Mamdani

Реализация объектов базы данных приведена в приложении А.

2.4 Выводы

В данном разделе описана постановка задачи, представлены диаграммы вариантов использование и «сущность–связь» разрабатываемой базы данных, приведен алгоритм получения вывода нечеткой экспертной системы для баз данных.

3 Технологическая часть

3.1 Выбор языка программирования и среды разработки

Программное обеспечение разрабатывается с использованием операционной системы Ubuntu 20.04 LTS [17].

Для создания базы данных нечетких экспертных систем принято решение об использовании СУБД PostgreSQL [18] по следующим причинам:

- максимальное соответствие стандарту SQL;
- открытый исходный код;
- поддержка БД неограниченного размера;
- наличие большого количества типов данных, в том числе uuid;
- кроссплатформенность;
- возможность создания пользователей на уровне БД.

Для генерации нечетких экспертных систем из исходных данных используется Matlab [19] в связи с рядом преимуществ:

- наличие пакета Fuzzy Logic Toolbox для работы с нечеткими экспертными системами типа Мамдани и Сугено:
  - генерация нечетких систем на основе примеров входных и выходных данных;
  - задание типа генерируемой экспертной системы и вида функций принадлежности;
  - изменение формы функций принадлежности;
- бесплатная лицензия для студентов;
- подключение к базе данных.

Приложения для работы с БД написано на языке Java [20] в соответствии с приведенными ниже соображениями:

- кроссплатформенность;
- строгая типизация данных;
- автоматический сборщик мусора;
- компиляция в Just-in-time;
- большое количество расширений и библиотек.

Для создания графического интерфейса используется фреймворк JavaFX поскольку:

- является кроссплатформенным;
- имеет утилиту Scene Builder для автоматической генерации FXML файлов

разметки;

– поддерживает большое количество управляющих элементов.

Для редактирования кода используется среда разработки IntelliJ IDEA [21] по нескольким причинам:

– встроенная утилита Scene Builder для работы с фреймфорком JavaFX;

– глубокий анализ кода, поддержка автодополнений;

– наличие инструментов для работы с базами данных.

3.2 Генерация экспертных систем

Для генерации экспертных систем на основе входных данных была написана программа в среде Matlab с интерфейсом в виде командной строки. При запуске запрашиваются необходимые для генерации экспертной системы данные (количество выходных переменных, кластеров для каждой из них, тип создаваемой системы и путь к файлу с данными) а так же логин и пароль для подключения к БД. В программе происходит добавление данных в сами таблицы, что разрешено только для пользователей с ролью администратор. Таким образом гарантируется, что никто более не сможет создавать новые экспертные системы. Код реализованной программы приведен в листингах 1–3.

Листинг 1 – Программа для генерации экспертных систем на основе примеров входных и выходных данных (часть 1)

```
function expert_sys_generation()
    n = input('Введите количество выходных переменных системы: ');
    mustBeInteger(n)
    if (n <= 0)
    fprintfКоличество(" входных переменных должно быть больше нуля\n");
    return
    end
    if (n <= 0)
    fprintfКоличество(" входных переменных должно быть больше нуля\n");
    return
    end
    clustNums = input('Введите количество кластеров для каждой переменной: ');
    mustBeInteger(clustNums)
    sType = inputВведите(" тип создаваемой системы (mamdani или sugeno): ", 's');
    if (~strcmp(sType, 'mamdani') && ~strcmp(sType, 'sugeno'))
    fprintfНекорректный(" тип системы\n");
    return
    end
    path = inputВведите(" путь к файлу с данными для генерации нечеткой экспертной системы: ", 's');

    login = inputВведите(" логин для подключения к базе данных: ", 's');
    password = inputВведите(" пароль для подключения к базе данных: ", 's');

    generateAndSaveFis(n, clustNums, sType, path, login, password)
end

function generateAndSaveFis(n, clustNums, sType, path, username, password)
    A = readmatrix(path);
    databasename = 'FuzzyDb';
    dbConnection = database(databasename,username,password);

    ops = genfisOptions('FCMClustering', 'FISType', sType);
    ops.NumClusters = clustNums;
    fis = genfis(A(:, 1:end - n), A(:, end - n + 1:end), ops);

    if (strcmp(sType, 'mamdani'))
        typeName = "Mamdani";
    else
        typeName = "Sugeno";
    end
```

Листинг 2 – Программа для генерации экспертных систем на основе примеров входных и выходных данных (часть 2)

```
    sId = saveSystem(dbConnection, typeName);
    rIds = saveRules(dbConnection, fis, sId);
    vIds = saveAntecedents(dbConnection, fis, rIds, sId);
    if (strcmp(sType, 'mamdani'))
        saveConsequentsMamdani(dbConnection, fis, rIds, sId);
    else
        saveConsequentsSugeno(dbConnection, fis, rIds, vIds, sId);
    end
    close(dbConnection);
end
function sId = saveSystem(dbConnection, typeName)
    tablename = 'system';
    colnames = {'s_id' 's_name' 's_type' 'specialization'};
    sId = getUid();
    sName = strcat('System_', string(char(randi([33 126],1,15))));
    insertData = table(sId, sName, typeName, "physics", 'VariableNames', colnames);
    sqlwrite(dbConnection, tablename, insertData);
end

function saveConsequentsMamdani(dbConnection, fis, rIds, sId)
    for i = 1:length(fis.Outputs)
        var = fis.Outputs(i);
        vId = getUid();
        query = "insert into variable values" + "(" + vId + ", '" + ...
        var.Name + "', " + var.Range(1) + ", " + var.Range(2) + ...
        ", null, " + sId + ");";
        execute(dbConnection, query);
        mIds = saveVariableMfs(dbConnection, var, vId);
        for j = 1:length(fis.Rules)
            ind = fis.Rules(j).Consequent(i);
            cId = getUid();
            query = "insert into consequent values" + "(" + cId + ", '" + ...
            mIds(ind) + "', " + rIds(j) + ");";
            execute(dbConnection, query);
        end
    end
end

function saveConsequentsSugeno(dbConnection, fis, rIds, vIds, sId)
    tablename = 'consequent';
    colnames = {'c_id' 'm_id' 'r_id' 'v_id'};
    for i = 1:length(fis.Outputs)
        var = fis.Outputs(i);
        vId = getUid();
        query = "insert into variable values" + "(" + vId + ", '" + ...
        var.Name + "', " + var.Range(1) + ", " + var.Range(2) + ...
        ", null, " + sId + ");";
        execute(dbConnection, query);
        mIds = saveConsequentSugenoMf(dbConnection, var, vIds);
        for j = 1:length(fis.Rules)
            ind = fis.Rules(j).Consequent(i);
            for k = 1:length(var.MembershipFunctions(1).Parameters)
                cId = getUid();
                insertData = table(cId, mIds(ind, k), rIds(j), vId, ...
                'VariableNames', colnames);
                sqlwrite(dbConnection, tablename, insertData);
            end
        end
    end
end

function mIds = saveConsequentSugenoMf(dbConnection, var, vIds)
    mIds = zeros(length(var.MembershipFunctions), ...
    length(var.MembershipFunctions(1).Parameters) + 1, 'int32');
    for j = 1:length(var.MembershipFunctions)
        mf = var.MembershipFunctions(j);
        for k = 1:length(mf.Parameters)
            mIds(j, k) = getUid();
            if (k < length(mf.Parameters))
                query = "insert into membership_function values" + ...
                "(" + mIds(j, k) + ", '" + mf.Name + ...
                "', 'linear', " + vIds(k) + ", " + ...
                mf.Parameters(k) + ");";
            else
                query = "insert into membership_function values" + ...
                "(" + mIds(j, k) + ", '" + mf.Name + ...
                "', 'crisp', null, " + mf.Parameters(k) + ");";
                end
            execute(dbConnection, query);
        end
    end
end
```

Листинг 3 – Программа для генерации экспертных систем на основе примеров входных и выходных данных (часть 3)

```
function mIds = saveVariableMfs(dbConnection, var, vId)
    mIds = zeros(length(var.MembershipFunctions), 'int32');
    for j = 1:length(var.MembershipFunctions)
        mIds(j) = getUid();
        mf = var.MembershipFunctions(j);
        m = mf.Parameters(2);
        s = mf.Parameters(1) * 3;
        query = "insert into membership_function values" + ...
        "(" + mIds(j) + ", '" + mf.Name + "', 'gauss', " + ...
        vId + ", " + m + ", " + ...
        s + ");";
        execute(dbConnection, query);
    end
end

function vIds = saveAntecedents(dbConnection, fis, rIds, sId)
    anttablename = 'antecedent';
    antcolnames = {'a_id' 'm_id'};
    ratablename = 'rule_antecedents';
    racolnames = {'r_id' 'a_id'};
    vIds = zeros(length(fis.Inputs), 'int32');
    for i = 1:length(fis.Inputs)
        var = fis.Inputs(i);
        vIds(i) = getUid();
        query = "insert into variable values" + "(" + vIds(i) + ", '" + ...
        var.Name + "', " + var.Range(1) + ", " + var.Range(2) + ...
        ", null, " + sId + ");";
        execute(dbConnection, query);
        mIds = saveVariableMfs(dbConnection, var, vIds(i));
        for j = 1:length(fis.Rules)
            ind = fis.Rules(j).Antecedent(i);
            if (~antecedentExists(dbConnection, mIds(ind)))
                aId = getUid();
                insertData = table(aId, mIds(ind), 'VariableNames', ...
                antcolnames);
                sqlwrite(dbConnection, anttablename, insertData);
            else
                aId = getaId(dbConnection, mIds(ind));
            end
        insertData = table(rIds(j), aId, 'VariableNames', racolnames);
        sqlwrite(dbConnection, ratablename, insertData);
        end
    end
end

function id = getUid()
    uid = javaMethod('toString', java.util.UUID.randomUUID);
    id = javaMethod('hashCode', uid);
    if (id < 0)
        id = id * -1;
    end
end

function aId = getaId(dbConnection, m_id)
    query = "SELECT a_id " + ...
    "FROM postgres.public.antecedent " + ...
    "WHERE m_id = " + m_id;

    data = fetch(dbConnection,query);
    aId = data(1, 1).a_id;
end

function exists = antecedentExists(dbConnection, m_id)
    query = "SELECT * " + ...
    "FROM postgres.public.antecedent " + ...
    "WHERE m_id = " + m_id;
    data = fetch(dbConnection,query);
    exists = height(data) ~= 0;
end

function rIds = saveRules(dbConnection, fis, sId)
    rIds = zeros(length(fis.Rules), 'int32');
    tablename = 'rule';
    colnames = {'r_id' 's_id' 'antecedent_connection' 'weight'};
    for i = 1:length(fis.Rules)
        rIds(i) = getUid();
        if (fis.Rules(i).Connection == 1)
            conn = "and";
        else
            conn = "or";
        end
        insertData = table(rIds(i), sId, conn, fis.Rules(i).Weight, 'VariableNames', colnames);
        sqlwrite(dbConnection, tablename, insertData);
    end
end
```

3.3 Описание интерфейса

Интерфейс программного обеспечения для работы с базой данных экспертных систем разделен на 3 области: слева находится список нечетких систем, в центре располагается информация о правилах логического вывода или нечетких переменных выбранной системы в зависимости от текущей вкладки, справа отображается информация о правиле вывода или переменной.

Внешний вид интерфейса при запуске приложения представлен на рисунке 3.1.

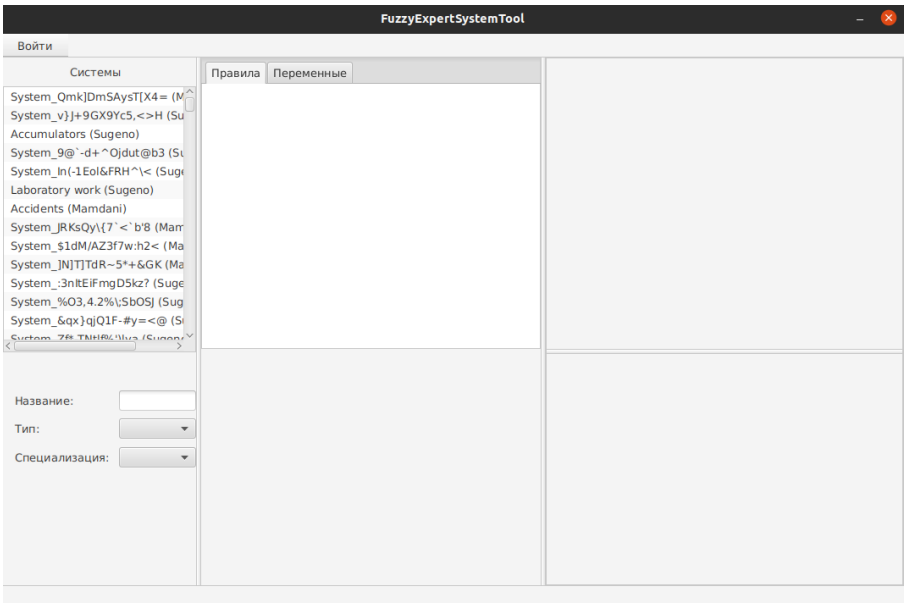


Рисунок 3.1 – Начальное состояние приложения

При нажатии на имя экспертной системы снизу отображается информация о ней (имя, тип, специализация), в центре – список определенных в ней правил логического вывода. Под списком правил находится таблица для получения результата нечеткого логического вывода текущей экспертной системы при хранящихся в базе значениях переменных. При выборе правила посредством нажатия левой кнопки мыши по соответствующей строке списка в правом верхнем углу отображается информация об антецедентах, в правом нижнем углу – о консеквентах соответствующего правила вывода. Для отображения информации об антецеденте или консеквенте также требуется нажать на соответствующую строку левой кнопкой мыши. Пример интерфейса, отображающего информацию об антецеденте и консеквенте правила нечеткой экспертной системы представлен на 3.2.

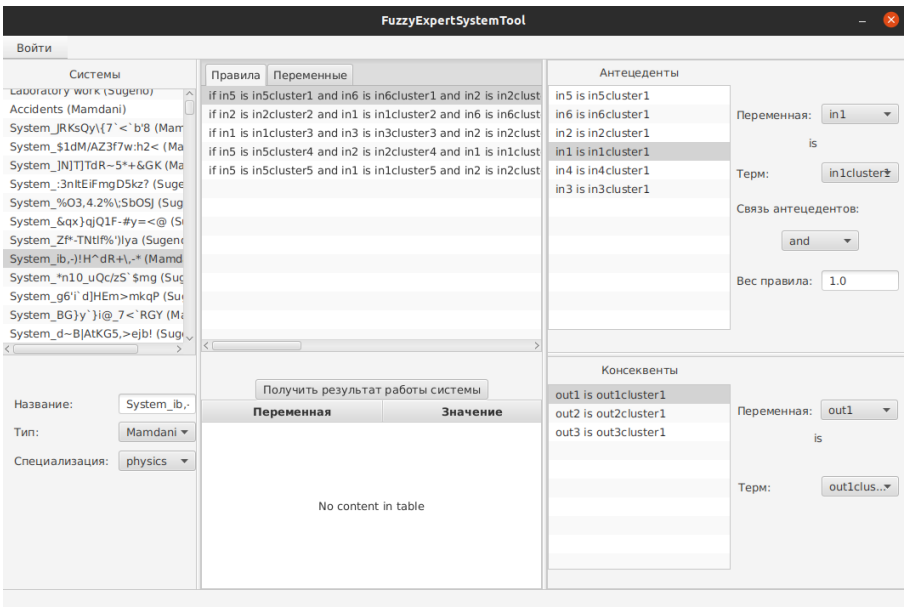


Рисунок 3.2 – Информация об антецедентах и консеквентах правила нечеткого логического вывода

Для получения информации о нечетких переменных необходимо нажать на соответствующую вкладку в центральной части интерфейса. При выборе нечеткой переменной снизу отображается информация о ней (название, текущее, минимальное и максимальное значения), справа появляется список функций принадлежности. При выборе функции предоставляется информация о ней и строится ее график в нижнем правом углу. Обычный пользователь может изменять текущее значение переменной для получения результата работы нечеткой экспертной системы. Пример изображения информации о переменной и ее функции принадлежности представлен на рисунке 3.3.

Для получения результата логического вывода системы необходимо изменить значения переменных на соответствующей вкладке при необходимости и нажать кнопку «получить результат работы системы» на вкладке правил. Пример получения результата логического вывода представлен на рисунке 3.4.

Для подключения к базе данных с существующими учетными данными используется кнопка «Войти» в левом верхнем углу. При нажатии появляется всплывающее окно для ввода логина и пароля. В случае успешной аутентификации и авторизации пользователь получает сообщение об успехе, в противном случае происходит сброс текущей учетной записи до записи по умолчанию (обычного пользователя). Пример окна для ввода логина и пароля представлен на рисунке 3.5.

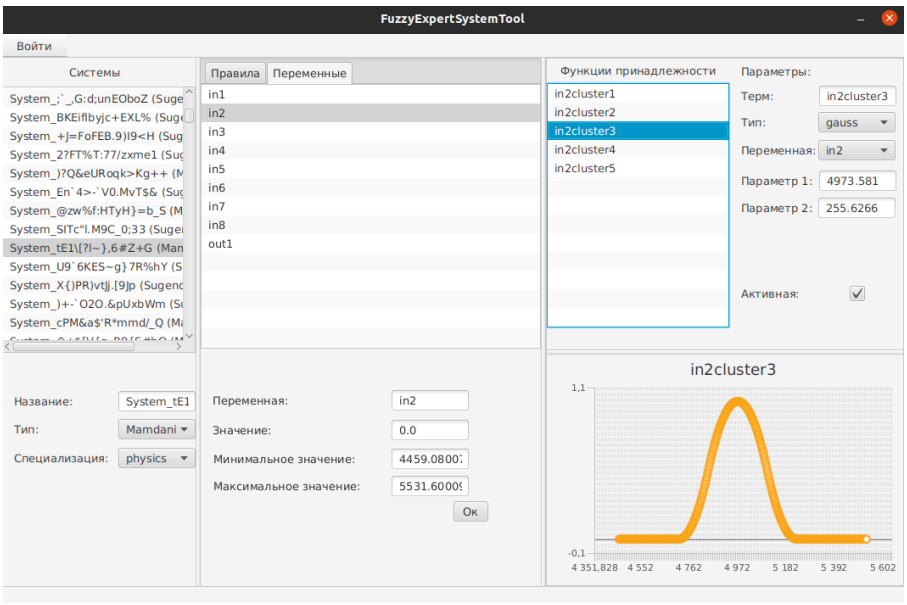


Рисунок 3.3 – Информация о переменной системы и ее функции принадлежности

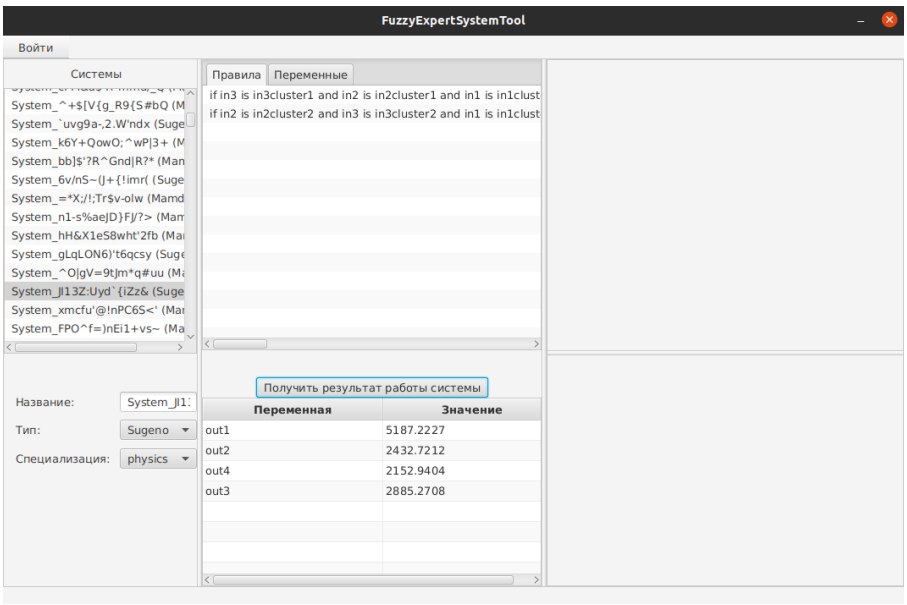


Рисунок 3.4 – Получение результата работы системы

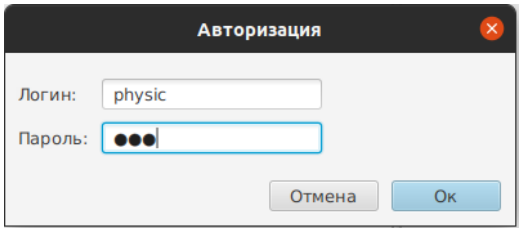


Рисунок 3.5 – Окно ввода данных для аутентификации и авторизации

При успешной авторизации пользователя с ролью администратора появляются кнопки добавления, изменения и удаления систем, правил, антецеден-

тов, консеквентов, нечетких переменных и их функций принадлежности. При авторизации пользователя с ролью эксперта появляются кнопки добавления, изменения и удаления всех компонентов нечетких экспертных систем, специализация которых совпадает со специализацией эксперта. Эксперт не может создавать или изменять экспертные системы. Примеры интерфейсов для администратора и эксперта представлены на рисунках 3.6–3.7 соответственно.

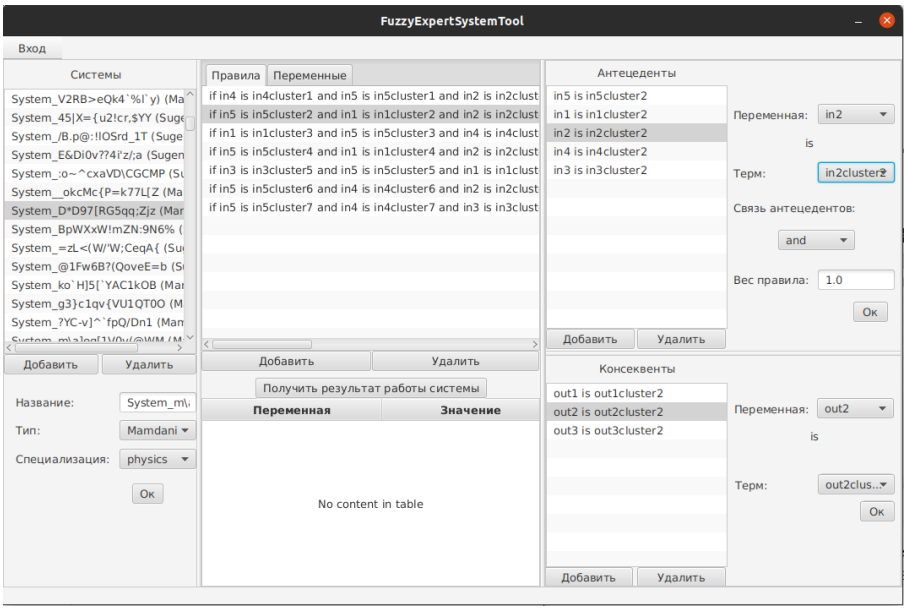


Рисунок 3.6 – Пример интерфейса для пользователя с ролью администратора

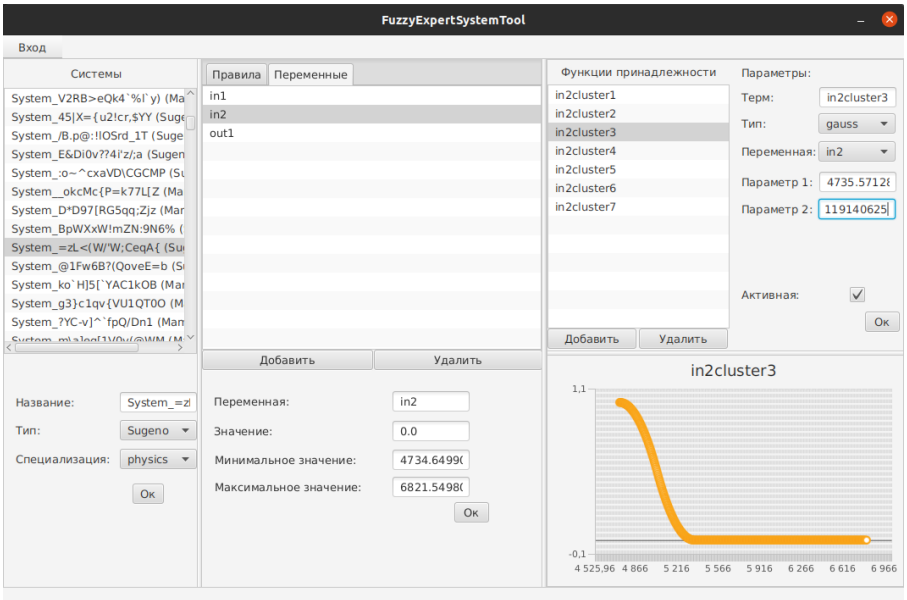


Рисунок 3.7 – Пример интерфейса для пользователя с ролью эксперта в области физики

3.4 Выводы

В данном разделе обоснован выбор средств разработки приложения, приведена программа для генерации нечетких экспертных систем, описан интерфейс приложения для работы с БД.

4 Исследовательская часть

4.1 Технические характеристики

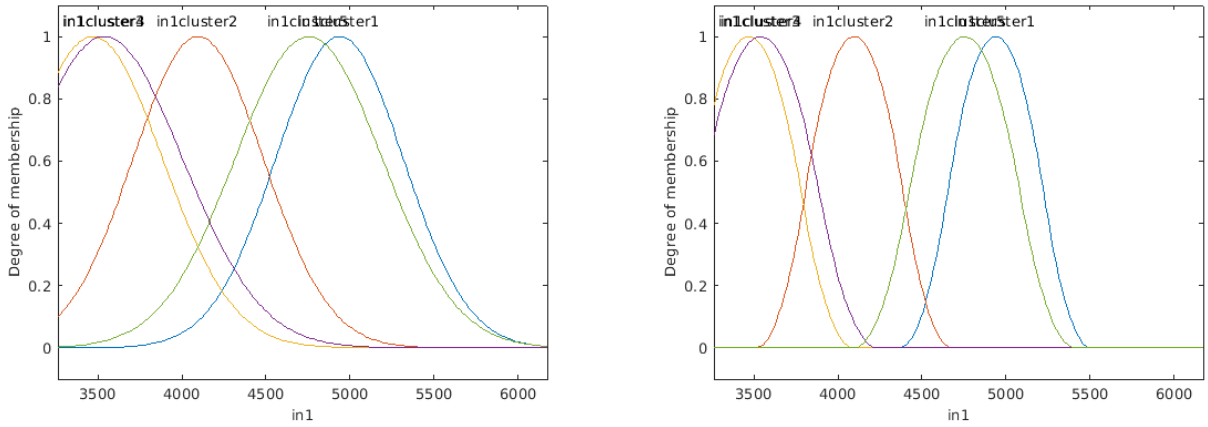
Технические характеристики устройства, на котором выполнялось исследование:

- операционная система: Ubuntu 22.04, 64-bit;
- оперативная память: 8 Гб;
- процессор: AMD Ryzen5 4500U [22].

4.2 Исследование генерируемых систем

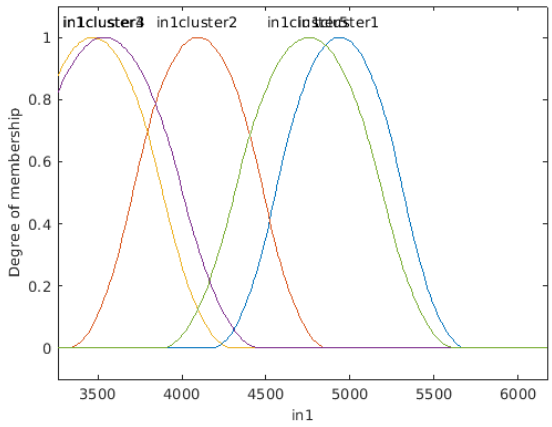
При разработке программы для генерации экспертных систем и добавления их в базу данных было обнаружено, что гауссова функция распределения и П-функция, используемая на уровне БД, имеют существенные отличия, влияющие на результат работы нечеткой экспертной системы.

На рисунке 4.1 а) приведены сгенерированные программно гауссовы функции принадлежности для одной из входных переменных системы, 4.2 в) – адаптированные под хранение в базе данных. Для сглаживания различий принято решение сохранять в базу гауссовы функции в виде П-функций, вычисляемых по формуле 1.5, с параметрами $\beta = \mu$, $\gamma = 3 \cdot \sigma$, где μ – математическое ожидание функции плотности нормального распределения, σ – среднеквадратическое отклонение.

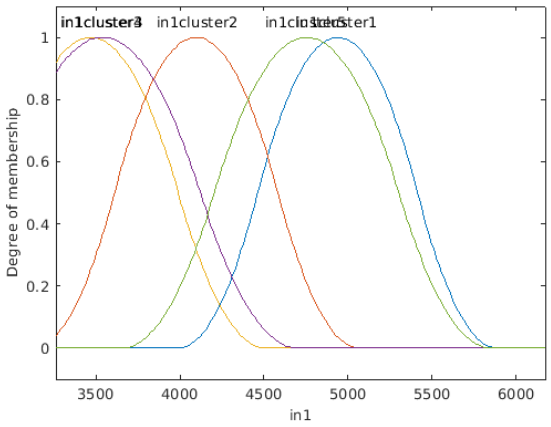


а) б)

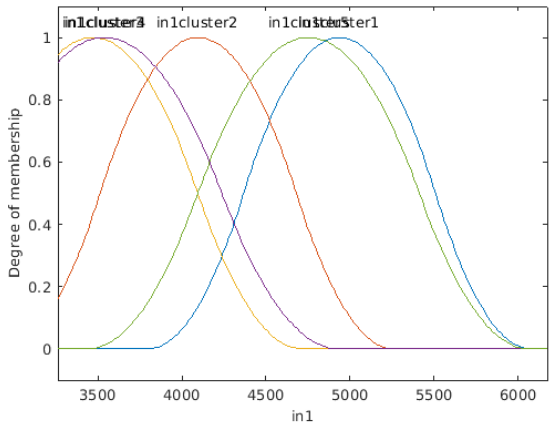
Рисунок 4.1 – Графики функций принадлежности переменной: а) сгенерированные с помощью matlab, б) П-функции с параметрами $\gamma = 1.5 \cdot \sigma$



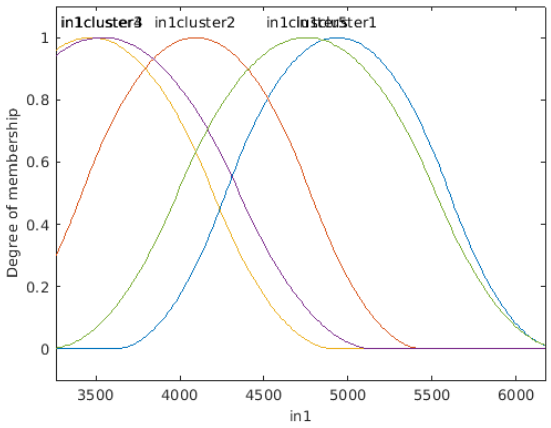
а)



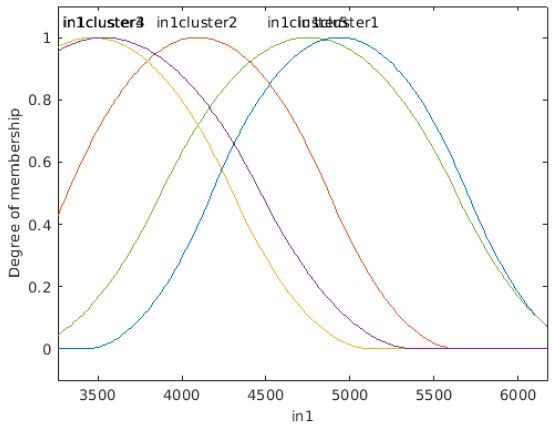
б)



в)



г)



д)

Рисунок 4.2 – Графики функций принадлежности переменной: а) П-функции с параметрами $\gamma = 2 \cdot \sigma$, б) П-функции с параметрами $\gamma = 2.5 \cdot \sigma$, в) П-функции с параметрами $\gamma = 3 \cdot \sigma$, г) П-функции с параметрами $\gamma = 3.5 \cdot \sigma$, д) П-функции с параметрами $\gamma = 4 \cdot \sigma$

При меньшем коэффициенте для вычисления γ и количестве кластеров, меньшем четырех, возможно появление больших зон, в которых все функции

принадлежности для переменной равны 0 (рисунок 4.1 б), рисунок 4.2 а) – б)), что не позволяет получить корректный результат работы системы.

Большее значение коэффициента создает существенные отличия в интервале значений, в которых значение функции принадлежности больше 0, поскольку, согласно правилу трех сигм для нормального распределения, 99.7% значений лежат в пределах трех стандартных отклонений от математического ожидания. Примеры П-функций принадлежности при $\gamma = 3.5 \cdot \sigma$ и $\gamma = 4 \cdot \sigma$ представлен на рисунке 4.2 г)–д).

В результате подобных изменений также меняются значения, получаемые при нечетком выводе. Из-за неполного соответствия интервалов, в которых функции принадлежности принимают значения, большие 0, возможны ситуации при которых невозможно получить результат логического вывода, если все правила системы конъюнктивные и хотя бы 1 антецедент в каждом из них принимает значение, равное 0, следовательно, ни одно из правил невозможно активизировать. В связи с этим в дальнейшем возникает необходимость добавления поддержки чистых гауссовых функций, а не их аппроксимации в виде П-функций, на уровне базы данных.

В таблице 4.1 приведено сравнение результатов работы системы с одной выходной переменной и 5 кластерами при разных коэффициентах c для П-функции ($\gamma = c \cdot \sigma$) с исходными данными и значением, вычисленным с использованием средств Matlab и исходных гауссовых функций на основе набора данных, приведенного в листинге 4. Значение, равное 0, говорит о том, что ни одно из правил системы не было активизировано и получить результат не удалось. Использованные наборы значений:

1. in1 = 3356.09, in2 = 6509.38, in3 = 5587.47, in4 = 4106.31, in5 = 5512.47
2. in1 = 4915.10, in2 = 6460.23, in3 = 6069.10, in4 = 4309.56, in5 = 5699.36

Листинг 4 – Пример входных и выходных данных для генерации нечеткой экспертной системы (последний столбец - значения выходной переменной)

```
1 6173.82    6051.42    5164.89    2395.29    6173.63    4070.99
2 3262.04    6290.51    5795.75    2546.39    6170.57    4826.16
3 4021.02    6654.06    6350.57    2402.60    4601.78    5653.51
4 3440.28    6655.82    4615.67    3789.04    4547.42    4230.85
5 4369.20    6419.66    4095.09    4157.41    4917.27    5623.29
6 4915.10    6460.23    6069.10    4309.56    5699.36    6924.39
7 3400.89    6695.89    7003.25    2476.91    6074.97    4165.13
8 4942.51    6759.04    4182.98    4479.01    5744.79    5589.45
```

Таблица 4.1 – Результаты работы системы

	$c = 1.5$	$c = 2$	$c = 2.5$	$c = 3$	$c = 3.5$	$c = 4$	Гаусс	Ожидание
1	0	0	0	4256.6	4579.4	4622.8	4921.3	4365.8
2	5571.7	0	7629.6	7802	4792.9	8196.3	6361.3	6924.4

4.3 Исследование времени работы запросов

Получение результата работы системы при большом количестве правил может быть ресурсоемким, поэтому для оптимизации времени получения результата работы системы предлагается кэшировать полученные данные с помощью in-memory базы данных Redis на 5 минут, если пользователь не изменяет значения входных переменных. На рисунке 4.3 приведена зависимость времени получения результата запроса для системы типа Мамдани с 5 входными, 4 выходными переменными и 7 правилами нечеткого вывода (соответственно, для каждой переменной создается 7 функций принадлежности) от номера итерации. Значение времени выполнения запроса для каждой итерации вычисляется путем усреднения времени выполнения 50 запросов. Для графика с использованием кэширования в Redis раз в 2 итерации происходит сброс кэширования для оценки затрат на загрузку данных при истечении TTL/изменении значений переменных.

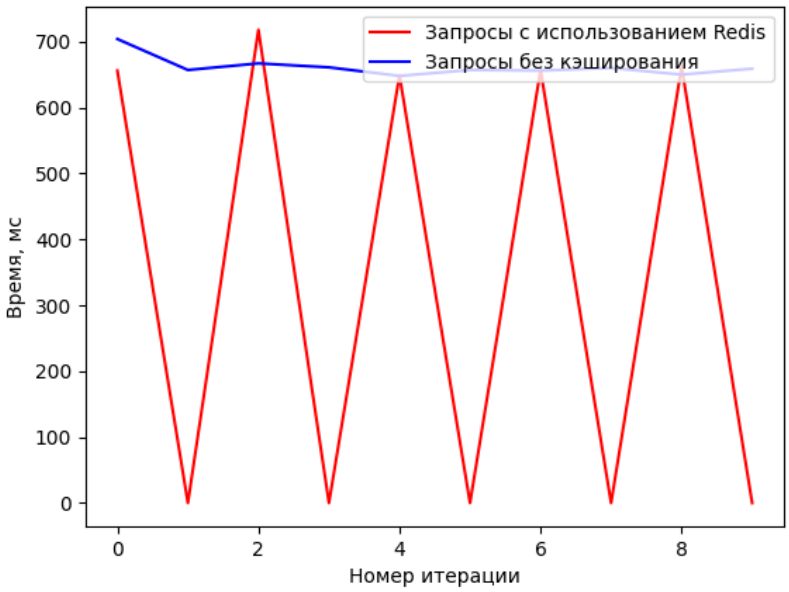


Рисунок 4.3 – Исследование времени выполнения запросов с кэшированием и без

На графике видно, что при использовании кэширования при отсутствии изменений значений переменных/функций принадлежности время выполнения запроса стремится к 0 миллисекундам, т.е. скорость увеличивается более чем в 600 раз. При необходимости обновления данных в кэше время выполнения запроса превышает время выполнения запроса без кэширования не более чем в 1,1 раз. Таким образом, использование встроенной базы данных для хранения результатов запросов оправдано, поскольку при хранении данных о 1000 экспертных системах время выполнения запроса для получения результата работы одной из них составляет около 0,6 секунд при небольшом количестве правил и функций принадлежности. При увеличении количества хранимых экспертных систем в 2 раза время выполнения запроса увеличивается в среднем в 1,15 раз (рисунок 4.4), кэширование результатов позволяет снизить нагрузку на базу данных и выполнение запроса.

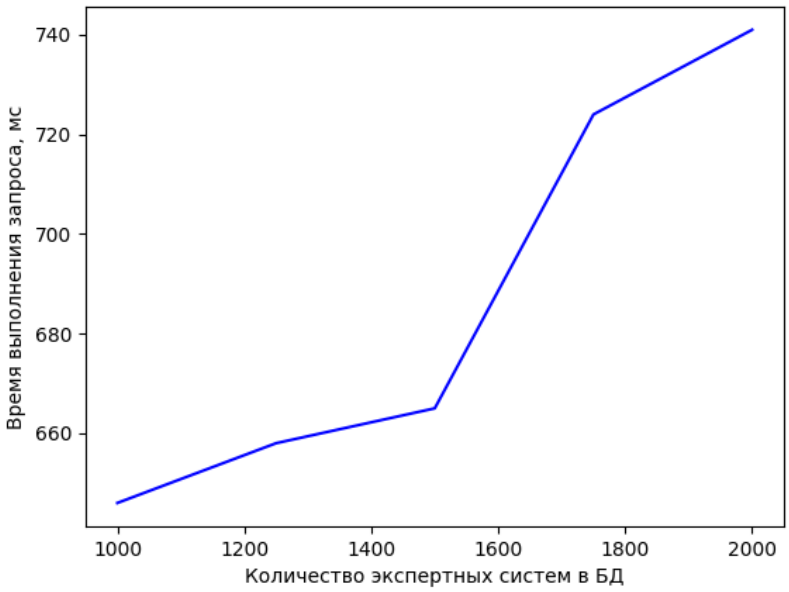


Рисунок 4.4 – Зависимость времени обработки запроса от количества экспертных систем в БД

4.4 Выводы

В данном разделе проведено исследование корректности хранимых данных и получаемых результатов, выявлено, что П-функция принадлежности не совпадает с гауссовой функцией, что ведет к изменению получаемых результатов работы нечеткой экспертной системы.

Исследовано время выполнения запросов получения результата работы нечеткой экспертной системы с кэшированием во встроенной базе данных Redis и без. Выявлено, что кэширование результатов дает выигрыш более чем в 600 раз при повторном обращении, при первичном сохранении запрос выполняется не более чем в 1,1 раз дольше запроса без кэширования.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы были изучены и проанализированы существующие решения проблемы хранения данных о нечетких экспертных системах, выделены их недостатки. По результатам проведенного анализа представлена формулировка решаемой задачи, выделена ролевая модель. Описаны сущности, используемые для хранения информации о нечетких экспертных системах типа Мамдани и Сугено. Описаны алгоритмы, позволяющие получить результат работы экспертной системы, приведена реализация объектов базы данных.

Описан интерфейс программного обеспечения, позволяющего работать с базой данных нечетких экспертных систем. ПО состоит из программы Matlab для добавления в базу данных информации, сгенерированной на основе файла с примерами входных и выходных данных системы, и настольного приложения, позволяющего просматривать и изменять хранимую информацию, подключаться к базе.

Проведено исследование полученных результатов работы системы, по результатам которого выявлено, что П-функция, используемая на уровне базы данных, не позволяет полностью аппроксимировать гауссову функцию принадлежности, что приводит к возможности получения некорректного результата работы сгенерированной автоматически системы. В связи с этим одним из направлений дальнейшего развития является добавление поддержки функций нормального распределения на уровне базы данных.

Также исследована скорость выполнения запросов на получение результата работы экспертной системы с кэшированием и без. Выявлено, что использование кэширования дает выигрыш по времени более чем в 600 раз при повторном запросе с неизменными правилами, функциями принадлежности и переменными. При необходимости обновления данных в кэше прогрыш составляет не более 1,1 раза.

Таким образом, поставленные задачи выполнены, цель работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Полещук О.М. Повышение эффективности оценки параметров технических систем на основе учета неопределенности разных типов // Лесной вестник. 2018. Т. 22, № 5. С. 121–128.

2. Короченцев Д.А., Зеленский А.А. Разработка адаптивного алгоритма идентификации технических каналов утечки информации // Colloquium-journal. 2019. № 13(37). С. 33–37.

3. С.Н. Басманов, А.А. Басманова. Обзор эволюции экспертных систем в медицине с точки зрения соответствия основным признакам // Перспективы развития информационных технологий. 2014. № 21. С. 126–130.

4. Демидова Г.Л., Лукичев Д.В. Регуляторы на основе нечеткой логики в системах управления техническими объектами. СПб: Университет ИТМО, 2017. Т. 81.

5. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование. 4 изд. ООО “И.Д. Вильямс”, 2007. Т. 1152.

6. Алгоритм настройки системы нечеткого логического вывода типа Мамдани / М.С. Голосовский, А.В. Богомолов, Д.С. Теребов [и др.] // Вестник ЮУрГУ. Серия “Математика. Механика. Физика”. 2018. Т. 10, № 3. С. 19–29.

7. Солдатова О.П. Многофункциональный имитатор нейронных сетей // Программные продукты и системы. 2012. № 3. С. 27–31.

8. В.Е. Сорокин. Хранение и эффективная обработка нечетких данных в СУБД PostgreSQL // Программные продукты и системы. 2017. № 4(30). С. 609–618.

9. Medina Juan Miguel, Blanco Ignacio J., Pons Olga. A fuzzy database engine for mongoDB // International Journal of Intelligent Systems. 2022. С. 1–34.

10. Е.Е. Бизянов, А.А. Гутник. Хранение нечетких чисел в реляционных базах данных информационных систем управления // Автоматизированные технологии и производства. 2016. № 3. С. 11–15.

11. Asanka D., Perera A. S. Defining Fuzzy Membership Functions for Fuzzy Data Warehouses // 4th International Conference for Convergence in Technology (I2CT). 2018.

12. Škrbić S., Racković M. Fuzzy Databases. Faculty of Science, University of Novi Sad, 2013. Vol. 105.

13. А.С. Ершов, А.С. Тортика. Обзор и сравнительный анализ современных систем управления базами данный // Вестник Саратовского государственного технического университета. 2020. С. 79–82.

14. Д.Л. Осипов. Технологии проектирования баз данных. М.: ДМК Пресс, 2019. Т. 498.

15. А.К. Манохин. Обзор популярных систем управления базами данных // Инновационное развитие техники и технологий в промышленности (ИНТЕКС-2021)». 2021. С. 234–237.

16. А.Н. Степовик, Н.В. Ефанова. Анализ реляционных и нереляционных баз данных // Цифровизация экономики: направления, методы, инструменты Сборник материалов I всероссийской студенческой научно-практической конференции. 2019. С. 414–416.

17. Ubuntu 22.04 LTS (Jammy Jellyfish) [Электронный ресурс]. Режим доступа: https://releases.ubuntu.com/22.04/. Дата обращения: 30.05.2022.

18. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. Режим доступа: https://www.postgresql.org/. Дата обращения: 31.05.2022.

19. MATLAB - MathWorks - MATLAB & Simulink [Электронный ресурс]. Режим доступа: https://www.mathworks.com/products/matlab.html. Дата обращения: 30.05.2022.

20. Java Software | Oracle [Электронный ресурс]. Режим доступа: https://www.oracle.com/java/. Дата обращения: 31.05.2022.

21. IntelliJ IDEA: функциональная и эргономичная IDE для разработки на Java от JetBrains [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/ru-ru/idea/. Дата обращения: 31.05.2022.

22. AMD Ryzen 5 4500U | AMD [Электронный ресурс]. Режим доступа: https://www.amd.com/ru/products/apu/amd-ryzen-5-4500u. Дата обращения: 30.05.2022.

ПРИЛОЖЕНИЕ А

Листинг 5 – Создание таблиц (часть 1)

```
create schema public;

create domain system_type as text
not null check (
    value = 'Sugeno' or value = 'Mamdani'
);

create domain specialization_type as text
not null check
(
    value = 'physics' or value = 'chemistry' or value = 'informatics'
);

create table system (
    s_id int primary key,
    s_name text unique,
    s_type system_type not null,
    specialization specialization_type
);

create table variable (
    v_id int primary key,
    v_name text not null,
    min_value real not null,
    max_value real not null,
    v_value real null check(v_value between min_value and max_value),
    s_id int not null references system(s_id) on delete cascade
);

create domain m_function_type as text
not null check
(
    value = 'trapezoidal' or value = 'triangle' or value = 'shoulder'
    or value = 'linguistic' or value = 'gauss' or value = 'crisp'
    or value = 'linear'
);

create domain linguistic_barrier as text
check
(
    value = 'Very' or value = 'More or less' or value = 'Plus' or value = 'Not'
    or value = 'Not very' or value = null
);

create table membership_function
(
    m_id int primary key,
    term text check (term is not null or m_type = 'linear' or m_type = 'crisp'),
    m_type m_function_type,
    v_id int references variable(v_id) on delete cascade check((m_type = 'linear' and v_id is not
     null) or (m_type != 'linear')),
    parameter1 real check (m_type != 'linguistic' or null),
    parameter2 real check (((m_type = 'trapezoidal' or m_type = 'triangle' or m_type = 'shoulder'
    )
    and parameter2 > parameter1) or m_type = 'gauss' or null),
    parameter3 real check (((m_type = 'trapezoidal' or m_type = 'triangle')
    and parameter3 > parameter2) or m_type = 'shoulder' or null),
    parameter4 real check (m_type = 'trapezoidal' and parameter4 > parameter3 or null),
    m_value real check (m_value between 0 and 1 or m_type = 'linear' or m_type = 'crisp'),
    p_id int references membership_function(m_id) on delete cascade,
    barrier linguistic_barrier check ((m_type = 'lingustic' and barrier != null) or null),
    is_active boolean default true
);

create domain ant_connection_type as text
check
(
    value = 'or' or value = 'and'
);

create table rule (
    r_id int primary key,
    s_id int references system(s_id) on delete cascade,
    antecedent_connection ant_connection_type,
    weight real not null default 1
);

create table antecedent (
    a_id int primary key,
    m_id int references membership_function(m_id) on delete cascade
);
```

Листинг 6 – Создание таблиц (часть 2)

```
create table consequent (
    c_id int primary key,
    m_id int references membership_function(m_id) on delete cascade,
    r_id int references rule(r_id) on delete cascade,
    v_id int references variable(v_id) on delete cascade
);

create table rule_antecedents (
    r_id int references rule(r_id) on delete cascade,
    a_id int references antecedent(a_id) on delete cascade,
    primary key (r_id, a_id)
);
```

Листинг 7 – Реализация вычисления значений функций принадлежности (часть 1)

```
create or replace function trapezoidal(value real, a real, b real, c real, d real)
returns real
as $$
declare
    res real;
begin
    if (value between a and b)
    then
        res := (value - a) / (b - a);
    else
        if (value between b and c)
        then
            res := 1;
        else
            if (value between c and d)
            then
                res := (d - value) / (d - c);
            else
                res := 0;
            end if;
        end if;
    end if;
    return res;
end;
$$ language plpgsql;

create or replace function triangle(value real, a real, b real, c real)
returns real
as $$
declare
    res real;
begin
    if (value between a and b)
    then
        res := (value - a) / (b - a);
    else
        if (value between b and c)
        then
            res := (c - value) / (c - b);
        else
            res := 0;
        end if;
    end if;
    return res;
end;
$$ language plpgsql;

create or replace function shoulder(value real, a real, b real, g real)
returns real
as $$
declare
    res real;
begin
    if (g < a)
    then
        select 1 - shoulder from shoulder(value, g, b, a) into res;
    else
        if (value <= a)
        then
            res := 0;
        else
            if (value <= b)
            then
                res := 2 * ((value - a) / (g - a)) ^ 2;
            else
                if (value <= g)
                then
                    res := 1 - 2 * ((value - g) / (g - a)) ^ 2;
```

Листинг 8 – Реализация вычисления значений функций принадлежности (часть 2)

```
                else
                    res := 1;
                end if;
            end if;
        end if;
    end if;
    return res;
end
$$ language plpgsql;

create or replace function gauss(value real, g real, b real)
returns real
as $$
declare
    res real;
begin
    if (value <= g)
    then
        select * from shoulder(value, g - b, (g - b / 2)::real, g) into res;
    else
        select * from shoulder(value, g, (g + b / 2)::real, g + b) into res;
        res := 1 - res;
    end if;
    return res;
end
$$ language plpgsql;

create or replace function linguistic(v real, barrier linguistic_barrier, pid int)
returns real
as $$
declare
    res real;
begin
    select case m_type
        when 'trapezoidal' then (select * from trapezoidal(v, parameter1, parameter2, parameter3,
     parameter4))
        when 'triangle' then (select * from triangle(v, parameter1, parameter2, parameter3))
        when 'shoulder' then (select * from shoulder(v, parameter1, parameter2, parameter3))
        when 'gauss' then (select * from gauss(v, parameter1, parameter2))
        when 'crisp' then parameter1
        when 'linguistic' then (select mf2.m_value from membership_function mf2 where mf2.m_id =
    pid)
        end
    from membership_function mf1
    where m_id = pid
    into res;

    select case barrier
        when 'Very'
            then res ^ 2
        when 'More or less'
            then res ^ 0.5
        when 'Plus'
            then res ^ 1.25
        when 'Not'
            then 1 - res
        when 'Not very'
            then 1 - res ^ 2
        else 0
    end
    into res;
    return res;
end;
$$ language plpgsql;

create or replace function update_membership()
returns trigger
as $$
begin
    update membership_function
    set m_value = case m_type
        when 'trapezoidal' then (select * from trapezoidal(new.v_value, parameter1, parameter2,
    parameter3, parameter4))
        when 'triangle' then (select * from triangle(new.v_value, parameter1, parameter2,
    parameter3))
        when 'shoulder' then (select * from shoulder(new.v_value, parameter1, parameter2,
    parameter3))
        when 'gauss' then (select * from gauss(new.v_value, parameter1, parameter2))
        when 'linguistic' then (select * from linguistic(new.v_value, barrier, p_id))
        when 'linear' then parameter1 * new.v_value
    end
    where v_id = new.v_id;
    return new;
end
$$ language plpgsql;
```

Листинг 9 – Реализация вычисления значений функций принадлежности (часть 3)

```
create trigger update_funcs
after update on variable
for row execute procedure update_membership();

create or replace function membership_func_check()
returns trigger
as $$
begin
    if (new.m_type = 'crisp')
    then
        new.m_value := new.parameter1;
    end if;
    return new;
end
$$ language plpgsql;

create trigger insert_mf
before insert on membership_function
for row execute procedure membership_func_check();
```

Листинг 10 – Реализация получения результата работы нечетких экспертных систем (часть 1)

```
create or replace function count_antecedents(sys_id int)
returns table (
    r_id int,
    ant_value real
)
as $$
begin
    return query (select distinct r.r_id as "rule_id", case
            when exists(select *
                    from rule_antecedents ra1
                        join antecedent a1 on ra1.a_id  = a1.a_id
                        join membership_function mf1 on mf1.m_id = a1.m_id
                    where ra1.r_id = r.r_id and not mf1.is_active)
                    then 0.0
            when r.antecedent_connection = 'or' then (select max(mf1.m_value) * r.weight
                from rule_antecedents ra1
                    join antecedent a1 on ra1.a_id  = a1.a_id
                    join membership_function mf1 on mf1.m_id = a1.m_id
                where ra1.r_id = r.r_id)
            when r.antecedent_connection = 'and' then (select min(mf1.m_value) * r.weight
                from rule_antecedents ra1
                    join antecedent a1 on ra1.a_id  = a1.a_id
                    join membership_function mf1 on mf1.m_id = a1.m_id
                where ra1.r_id = r.r_id)
            end as "value"
        from system s
            join rule r on s.s_id = r.s_id
            join rule_antecedents ra on r.r_id = ra.r_id
            join antecedent a on ra.a_id  = a.a_id
            join membership_function mf on mf.m_id = a.m_id
        where s.s_id = sys_id);
    end;
$$ language plpgsql;

create or replace function get_output_Sugeno(s_id int)
returns table (
    r_id int,
    consequent int,
    value real,
    weight real
)
as $$
begin
    return query
    (select q.r_id, q.consequent, sum(q.m_value) as "value", q.weight::real as "weight"
    from
        (select distinct a.r_id, c.v_id as "consequent", mf.m_value, a.ant_value as "weight"
        from (select * from count_antecedents(s_id)) a
        join consequent c on a.r_id = c.r_id
        join membership_function mf on c.m_id = mf.m_id
        where a.ant_value != 0 and mf.is_active) q
    group by q.r_id, q.consequent, q.weight);
end
$$ language plpgsql;

create or replace function get_result_Sugeno(s_id int)
returns table (
    v_id int,
```

Листинг 11 – Реализация получения результата работы нечетких экспертных систем (часть 2)

```
    value real
)
as $$
begin
    return query
    (select q.consequent, sum(q.weight * q.value) / sum(q.weight) as "value"
    from get_output_Sugeno(s_id) q
    group by q.consequent);
end
$$ language plpgsql;

create or replace function get_fuzzy_Mamdani_output(s_id int)
returns table (
    v_id int,
    value real,
    cent real
)
as $$
begin
    return query
    (
    with intervals as (
        select q.r_id, q.v_id, q.m_id, q.value,
        memb_func_get_start(q.value, q.m_id) as "start",
        memb_func_get_end(q.value, q.m_id) as "end"
        from
            (select c.r_id, mf.v_id, mf.m_id, case r.antecedent_connection
                when 'or' then case
                    when max(ant_value) > max(mf.m_value) or mf.m_value is null
                    then max(ant_value)
                    else max(mf.m_value)
                    end
                else case
                    when min(ant_value) < min(mf.m_value) or mf.m_value is null
                    then min(ant_value)
                    else min(mf.m_value)
                    end
                end as "value"
            from count_antecedents(s_id) a
            join "rule" r  on a.r_id = r.r_id
            join consequent c on c.r_id = r.r_id
            join membership_function mf on c.m_id = mf.m_id
            where mf.is_active
            group by c.r_id, mf.v_id, mf.m_id, r.antecedent_connection) q)
    select ql.v_id, ql.value, ((ql.start + ql.end) / 2)::real as "cent"
    from
        (select q.v_id, q.value, case
        when (select max("end")
            from intervals i
            where i.value > q.value and i.end < q.end) > q.start
        then (select max("end")
            from intervals i
            where i.value > q.value and i.end < q.end)
        else q.start
        end as "start",
        case
        when (select min("start")
            from intervals i
            where i.value > q.value and i.start > q.start) < q.end
        then (select min("start")
            from intervals i
            where i.value > q.value and i.start > q.start)
        else q.end
        end as "end"
    from intervals q) ql);
end;
$$ language plpgsql;

create or replace function get_result_Mamdani(s_id int)
returns table (
    v_id int,
    value real
)
as $$
begin
    return query
    (select q.v_id, case sum(q.value)
                    when 0 then 0
                    else sum(q.value * q.cent) / sum(q.value)
                    end as "value"
        from get_fuzzy_Mamdani_output(s_id) q
        group by q.v_id);
end
$$ language plpgsql;
```

Листинг 12 – Реализация получения результата работы нечетких экспертных систем (часть 3)

```
create or replace function get_output(sys_id int)
returns table(
    var_name text,
    value real
)
security definer
as $$
declare
    t text;
begin
    drop table if exists tmp;
    create temp table tmp
    (
        v_id int,
        value real
    );
    select s_type
    from system s
    where s.s_id = sys_id
    into t;
    if (t = 'Sugeno') then
        insert into tmp select * from get_result_Sugeno(sys_id);
    else
        insert into tmp select * from get_result_Mamdani(sys_id);-- into tmp;
    end if;
    return query (select v_name, t.value
        from tmp t join variable v on t.v_id = v.v_id);
end;
$$ language plpgsql;
```

Листинг 13 – Функции, используемые для дефаззификации методом центра тяжести для дискретных множеств (часть 1)

```
create or replace function trapezoidal_get_start(degree real, a real, b real, c real, d real)
returns real
as $$
declare
    res real;
begin
    if (abs(degree) < 1e-5)
    then
        res := a;
    else
        if (abs(degree - 1) < 1e-5)
        then
            res := b;
        else
            res := degree * (b - a) + a;
        end if;
    end if;
    return res;
end
$$ language plpgsql;

create or replace function trapezoidal_get_end(degree real, a real, b real, c real, d real)
returns real
as $$
declare
    res real;
begin
    if (abs(degree) < 1e-5)
    then
        res := d;
    else
        if (abs(degree - 1) < 1e-5)
        then
            res := c;
        else
            res := d - degree * (d - c);
        end if;
    end if;
    return res;
end
$$ language plpgsql;

create or replace function triangle_get_start(degree real, a real, b real, c real)
returns real
as $$
declare
    res real;
begin
```

Листинг 14 – Функции, используемые для дефаззификации методом центра тяжести для дискретных множеств (часть 2)

```
    if (abs(degree) < 1e-5)
    then
        res := a;
    else
        res := degree * (b - a) + a;
    end if;
    return res;
end
$$ language plpgsql;

create or replace function triangle_get_end(degree real, a real, b real, c real)
returns real
as $$
declare
    res real;
begin
    if (abs(degree) < 1e-5)
    then
        res := c;
    else
        res := c - degree * (c - b);
    end if;
    return res;
end
$$ language plpgsql;

create or replace function shoulder_get_start(degree real, a real, b real, g real, var int)
returns real
as $$
declare
    res real;
    tmp_res real;
begin
    select min_value from variable where v_id = var into res;
    if (a < g)
        then begin
            if (abs(degree) < 1e-5)
                then tmp_res := a;
            else
                if (abs(degree - 1) < 1e-5)
                    then tmp_res := g;
                else
                    if (degree <= 0.5)
                        then tmp_res := sqrt(degree / 2) * (g - a) + a;
                    else
                        tmp_res := g - (g - a) * sqrt((1 - degree) / 2);
                    end if;
                end if;
            end if;
            if (tmp_res > res)
                then res := tmp_res;
            end if;
        end;
    end if;
    return res;
end;
$$ language plpgsql;

create or replace function shoulder_get_end(degree real, a real, b real, g real, var int)
returns real
as $$
declare
    res real;
begin
    if (a > g)
    then
        select shoulder_get_start((1 - degree)::real, g, b, a, var) into res;
    else
        select max_value from variable where v_id = var into res;
    end if;
    return res;
end;
$$ language plpgsql;

create or replace function gauss_get_start(degree real, g real, b real, var int)
returns real
as $$
begin
    return (select * from shoulder_get_start(degree::real, g - b, (g - b/2)::real, g, var));
end;
$$ language plpgsql;

create or replace function gauss_get_end(degree real, g real, b real, var int)
returns real
as $$
```

Листинг 15 – Функции, используемые для дефаззификации методом центра тяжести для дискретных множеств (часть 3)

```
begin
    return (select * from shoulder_get_end((1 - degree)::real, g + b, (g + b/2)::real, g, var));
end
$$ language plpgsql;
create or replace function linguistic_get_start(degree real, barrier linguistic_barrier, pid int)
returns real
as $$
declare
    res real;
begin
    select case barrier
        when 'Very' then degree ^ 0.5
        when 'More or less' then degree ^ 2
        when 'Plus' then degree ^ (4/5)
        when 'Not' then 1 - degree -- x = 1 - y^2
        when 'Not very' then (1 - degree) ^ 0.5
        end
    into res;
    return (
        select case m_type
            when 'trapezoidal' then (select * from trapezoidal_get_start(res, parameter1,
    parameter2, parameter3, parameter4))
            when 'triangle' then (select * from triangle_get_start(res, parameter1, parameter2,
    parameter3))
            when 'shoulder' then (select * from shoulder_get_start(res, parameter1, parameter2,
    parameter3, v_id))
            when 'gauss' then (select * from gauss_get_start(res, parameter1, parameter2, v_id))
            when 'linguistic' then (select * from linguistic_get_start(res, barrier, p_id))
            end
        from membership_function
        where m_id = pid);
end
$$ language plpgsql;

create or replace function linguistic_get_end(degree real, barrier linguistic_barrier, pid int)
returns real
as $$
declare
    res real;
begin
    select case barrier
        when 'Very' then degree ^ 0.5
        when 'More or less' then degree ^ 2
        when 'Plus' then degree ^ (4/5)
        when 'Not' then 1 - degree -- x = 1 - y^2
        when 'Not very' then (1 - degree) ^ 0.5
        end
    into res;
    return (
        select case m_type
            when 'trapezoidal' then (select * from trapezoidal_get_end(res, parameter1,
    parameter2, parameter3, parameter4))
            when 'triangle' then (select * from triangle_get_end(res, parameter1, parameter2,
    parameter3))
            when 'shoulder' then (select * from shoulder_get_end(res, parameter1, parameter2,
    parameter3, v_id))
            when 'gauss' then (select * from gauss_get_end(res, parameter1, parameter2, v_id))
            when 'linguistic' then (select * from linguistic_get_end(res, barrier, p_id))
            end
        from membership_function
        where m_id = pid);
end
$$ language plpgsql;

create or replace function memb_func_get_start(degree real, mf_id int)
returns real
as $$
begin
    return (
    select case m_type
        when 'trapezoidal' then (select * from trapezoidal_get_start(degree, parameter1,
    parameter2, parameter3, parameter4))
        when 'triangle' then (select * from triangle_get_start(degree, parameter1, parameter2,
    parameter3))
        when 'shoulder' then (select * from shoulder_get_start(degree, parameter1, parameter2,
    parameter3, v_id))
        when 'gauss' then (select * from gauss_get_start(degree, parameter1, parameter2, v_id))
        when 'linguistic' then (select * from linguistic_get_start(degree, barrier, p_id))
        end
    from membership_function mf
    where m_id = mf_id);
end
$$ language plpgsql;
```

Листинг 16 – Функции, используемые для дефаззификации методом центра тяжести для дискретных множеств (часть 4)

```
create or replace function memb_func_get_end(degree real, mf_id int)
returns real
as $$
begin
    return (
    select case m_type
        when 'trapezoidal' then (select * from trapezoidal_get_end(degree, parameter1, parameter2
    , parameter3, parameter4))
        when 'triangle' then (select * from triangle_get_end(degree, parameter1, parameter2,
    parameter3))
        when 'shoulder' then (select * from shoulder_get_end(degree, parameter1, parameter2,
    parameter3, v_id))
        when 'gauss' then (select * from gauss_get_end(degree, parameter1, parameter2, v_id))
        when 'linguistic' then (select * from linguistic_get_end(degree, barrier, p_id))
        end
    from membership_function mf
    where m_id = mf_id);
end
$$ language plpgsql;
```

Листинг 17 – Создание ролей на уровне базы данных (часть 1)

```
create role administrator superuser;
grant administrator to admin;
revoke all on all tables in schema public from public;
revoke all on database postgres from public;
revoke all on schema public from public;
grant connect on database postgres to user_role;

grant usage on schema pg_catalog to user_role;

create role user_role;
revoke all on schema public from user_role;
set search_path to public;

grant usage on schema public to user_role;
revoke all on all tables in schema public from user_role;
grant select on all tables in schema public to user_role;
revoke all on all functions in schema public from user_role;

grant execute on function get_output(int) to user_role;
create or replace procedure set_variable_value(var_id int, value real)
security definer
as $$
begin
    update variable set v_value = value where v_id = var_id;
end
$$ language plpgsql;

grant execute on procedure set_variable_value(int, real) to user_role;

create or replace function get_roles()
returns table(
    usr_role name)
as $$
begin
    return query
        select rolname from pg_roles where pg_has_role((select current_user), oid, 'member');
end;
$$ language plpgsql;

grant select on table pg_roles to user_role;

grant execute on function get_roles() to user_role;

create role expert inherit;

grant select on all tables in schema pg_catalog to admin;

------------------PHYSICS EXPERTS VIEWS-------------------------

create view physics_expert_systems as
    select *
    from system
    where specialization = 'physics'
with cascaded check option;

create view physics_expert_variables as
    select *
    from variable
    where s_id in (
    select s_id from physics_expert_systems)
with cascaded check option;
```

Листинг 18 – Создание ролей на уровне базы данных (часть 2)

```
create view physics_expert_membership_functions as
    select *
    from membership_function
    where v_id in (select v_id from physics_expert_variables)
with cascaded check option;
create view physics_expert_rules as
    select *
    from rule
    where s_id in (select s_id from physics_expert_systems)
with cascaded check option;

create view physics_expert_antecedents as
    select * from antecedent a
    where m_id in (select m_id from physics_expert_membership_functions)
with cascaded check option;

create view physics_expert_consequents as
    select * from consequent c
    where r_id in (select r_id from physics_expert_rules)
    and m_id in (select m_id from physics_expert_membership_functions)
    and (v_id is null or v_id in (select v_id from physics_expert_variables))
with cascaded check option;

create view phycics_expert_rule_antecedents as
    select * from rule_antecedents ra
    where r_id in (select r_id from physics_expert_rules)
    and a_id in (select a_id from physics_expert_antecedents)
with cascaded check option;

grant user_role to expert;
create role physics inherit;
create role chemistry inherit;
create role informatics inherit;

grant expert to physics;
grant expert to chemistry;
grant expert to informatics;

grant all on physics_expert_variables to physics;
grant all on phycics_expert_rule_antecedents to physics;
grant all on physics_expert_antecedents to physics;
grant all on physics_expert_consequents to physics;
grant all on physics_expert_membership_functions to physics;
grant all on physics_expert_rules to physics;
grant all on physics_expert_systems to physics;

------------------CHEMISTRY EXPERTS VIEWS-----------------------
create view chemistry_expert_systems as
    select *
    from system
    where specialization = 'chemistry'
with cascaded check option;

create view chemistry_expert_variables as
    select *
    from variable
    where s_id in (
    select s_id from chemistry_expert_systems)
with cascaded check option;

create view chemistry_expert_membership_functions as
    select *
    from membership_function
    where v_id in (select v_id from chemistry_expert_variables)
with cascaded check option;

create view chemistry_expert_rules as
    select *
    from rule
    where s_id in (select s_id from chemistry_expert_systems)
with cascaded check option;

create view chemistry_expert_antecedents as
    select * from antecedent a
    where m_id in (select m_id from chemistry_expert_membership_functions)
with cascaded check option;

create view chemistry_expert_consequents as
    select * from consequent c
    where r_id in (select r_id from chemistry_expert_rules)
    and m_id in (select m_id from chemistry_expert_membership_functions)
    and (v_id is null or v_id in (select v_id from chemistry_expert_variables))
with cascaded check option;

create view chemistry_expert_rule_antecedents as
    select * from rule_antecedents ra
    where r_id in (select r_id from chemistry_expert_rules)
    and a_id in (select a_id from chemistry_expert_antecedents)
```

Листинг 19 – Создание ролей на уровне базы данных (часть 3)

```
with cascaded check option;

grant all on chemistry_expert_variables to chemistry;
grant all on chemistry_expert_antecedents to chemistry;
grant all on chemistry_expert_rule_antecedents to chemistry;
grant all on chemistry_expert_consequents to chemistry;
grant all on chemistry_expert_membership_functions to chemistry;
grant all on chemistry_expert_rules to chemistry;
grant all on chemistry_expert_systems to chemistry;

------------------INFORMATICS EXPERTS VIEWS----------------------
create view informatics_expert_systems as
    select *
    from system
    where specialization = 'informatics'
with cascaded check option;

create view informatics_expert_variables as
    select *
    from variable
    where s_id in (
    select s_id from informatics_expert_systems)
with cascaded check option;

create view informatics_expert_membership_functions as
    select *
    from membership_function
    where v_id in (select v_id from informatics_expert_variables)
with cascaded check option;

create view informatics_expert_rules as
    select *
    from rule
    where s_id in (select s_id from informatics_expert_systems)
with cascaded check option;

create view informatics_expert_antecedents as
    select * from antecedent a
    where m_id in (select m_id from informatics_expert_membership_functions)
with cascaded check option;

create view informatics_expert_consequents as
    select * from consequent c
    where r_id in (select r_id from informatics_expert_rules)
    and m_id in (select m_id from informatics_expert_membership_functions)
    and (v_id is null or v_id in (select v_id from informatics_expert_variables))
with cascaded check option;

create view informatics_expert_rule_antecedents as
    select * from rule_antecedents ra
    where r_id in (select r_id from informatics_expert_rules)
    and a_id in (select a_id from informatics_expert_antecedents)
with cascaded check option;

grant all on informatics_expert_variables to informatics;
grant all on informatics_expert_antecedents to informatics;
grant all on informatics_expert_rule_antecedents to informatics;
grant all on informatics_expert_consequents to informatics;
grant all on informatics_expert_membership_functions to informatics;
grant all on informatics_expert_rules to informatics;
grant all on informatics_expert_systems to informatics;
```