Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Капустинський Роман Ігорович

«Алгоритми нечіткого управління системою доступу до бази даних підприємства / Algorithms for fuzzy control of the enterprise database access system»

Студент групи КІм – 21
Капустинський Роман Ігорович
_____

Науковий керівник
к.т.н, доц. Л.О. Дубчак
_____

Тернопіль – 2020

РЕЗЮМЕ

Кваліфікаційна робота на тему “Алгоритми нечіткого управління системою доступу до бази даних підприємства” зі спеціальності 123 «Комп’ютерна інженерія» освітнього ступеня «магістр» написана обсягом 99 сторінок і містить 44 ілюстрації, 2 таблиці, 4 додатки та 52 джерела за переліком посилань.

Метою роботи є розроблення алгоритмів управління системою доступу до бази даних підприємства на основі нечіткої логіки та проектування системи контролю доступу.

Методи досліджень. Для розв’язання поставлених задач у кваліфікаційній роботі використано: методи: аналізу(для проведення огляду сучасних нечітких систем); синтезу (для розробки алгоритму управління доступом); порівняння (для виявлення відмінностей між розробленим алгоритмом та вже існуючим).

Результати дослідження: алгоритми адміністрування та контролю доступу до бази даних підприємства на основі нечіткої логіки, система контролю доступу.

Результати можуть бути використані на приватних підприємствах та організаціях, які потребують захисту персональних даних у базах даних.

Орієнтовні напрямки розвитку досліджень: збільшення ролі комп’ютеризації у процесі адміністрування; оптимізації діяльності підприємства на основі застосування інформаційних технологій; створення і впровадження інформаційних систем нового покоління, на основі нечіткої логіки; ефективний контроль доступу до персональних даних; захист інформації та матеріальних цінностей підприємства.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, БАЗА ДАНИХ, БАЗА ЗНАНЬ, НЕЧІТКА ЛОГІКА, НЕЧІТКИЙ ВИСНОВОК МАМДАНІ, ЗАХИСТ ІНФОРМАЦІЇ, ІНФОРМАЦІЙНА СИСТЕМА.

RESUME

Qualification work on "Algorithms for fuzzy control of the enterprise database access system" in the specialty 123 "Computer Engineering" educational degree "Master" is written in 99 pages and contains 44 illustrations, 2 tables, 4 appendices and 52 sources from the list of references.

The purpose of the work is to develop algorithms for managing the system of access to the database of the enterprise on the basis of fuzzy logic and designing an access control system.

Research methods. To solve the tasks in the qualification work we used: methods of: analysis (for a review of modern fuzzy systems); synthesis (to develop an access control algorithm); comparison (to identify differences between the developed algorithm and the existing one).

Research results: algorithms for administration and control of access to the enterprise database based on fuzzy logic, access control system.

The results of work can be used in private enterprises and organizations that need personal data protection in databases.

Guidelines for research development: increasing the role of computerization in the administration process; optimization of enterprise activity on the basis of application of information technologies; creation and implementation of new generation information systems, based on fuzzy logic; effective control of access to personal data; protection of information and material values of the enterprise.

KEYWORDS: ALGORITHM, DATABASE, KNOWLEDGE BASE, FUZZY LOGIC, FUZZY CONCLUSION OF MAMDANI, INFORMATION PROTECTION, INFORMATION SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

КМР – Кваліфікаційна магістерська робота

SQL (англ. Structured Query Language) – Структурована мова запитів

NoSQL (англ. non-SQL) – Бази даних, які не використовують мову запитів

RDBMS (англ. Relational Database Management System) – Реляційна система управління базами даних

IoT (англ. Internet of Things) – Інтернет речей

IT (англ. Information Technology) – Інформаційні технології

ВСТУП

Актуальність теми. У наш час люди використовують Інтернет для багатьох речей, наприклад покупок, перегляду відео, прослуховування музики, спілкування, навчання та роботи. Не секрет, що швидке розширення Всесвітньої павутини протягом останніх кількох десятиліть принесло нам дивовижні речі і з кожним роком у нас з'являється все більше можливостей [1].

Невпинно збільшується кількість персональних даних які люди надають третій стороні. Дані стають все більш цінними. Крім того, навички та можливості для отримання різних типів персональних даних розвиваються надзвичайно швидко. Несанкціонована, необережна або необізнана обробка персональних даних може завдати великої шкоди людям і компаніям. Саме тому, на сьогоднішній день, захист інформації вже набув широкого значення та закріплений на законодавчому рівні. Метою захисту персональних даних є не лише їх збереження у таємниці, а й захист основних прав і свобод осіб, пов'язаних із ними. Захищаючи ці дані, можна гарантувати, що права та свободи людей не порушуються. Недотримання правил захисту персональних даних може призвести до різних ситуацій, починаючи від викрадення коштів з банківського рахунку людини до спричинення ситуації, що загрожує її життю [2].

Захист інформації повинен мати системний характер, тобто включати у себе різні засоби захисту (апаратні, програмні, фізичні, організаційні). Вони повинні застосовуватися одночасно та бути під єдиним контролем. Існує велика кількість засобів захисту інформації: засоби ідентифікації та аутентифікації користувачів; засоби шифрування інформації, брандмауери; віртуальні приватні мережі; інструменти фільтрації вмісту; інструменти перевірки цілісності вмісту диска; засоби антивірусного захисту; системи виявлення вразливості мережі та аналізатори мережевих атак.

Щодня хакери здійснюють атаки, призначені для викрадення конфіденційних даних, і сервери баз даних організацій часто є основною ціллю цих атак. Причина, через яку бази даних так часто стають цілями досить проста - вони є основою будь-

якої організації, у них зберігають записи клієнтів та інші конфіденційні ділові дані. Організації недостатньо добре захищають ці життєво важливі дані. Найпоширенішою загрозою для проникнення у базу даних є надання надмірного доступу працівникам. Коли працівникам надаються привілеї бази даних за замовчуванням, що перевищують вимоги їхніх робочих функцій, цими привілеями можна зловживати. Крім того, деякі компанії не оновлюють привілеї доступу для працівників, які змінюють посаду в організації або її покидають [3]. Контроль доступу до баз даних - це спосіб надання доступу до конфіденційних даних компанії лише тим людям (користувачам баз даних), яким дозволено отримати доступ до таких даних та обмежити доступ неавторизованих осіб [4]. Тому створення системи контролю доступу до бази даних підприємства є доцільним.

Актуальність розробки цієї системи полягає у наступному:

– збільшення ролі комп'ютеризації в процесі адміністрування;

– оптимізація діяльності підприємства на основі застосування інформаційних технологій;

– ефективна та безпечна організація доступу до конфіденційних даних підприємства, які містяться у базі даних;

– створення та впровадження інформаційних систем нового покоління, заснованих на нечіткій логіці;

Мета і завдання дослідження. Розробити алгоритм контролю доступу до бази даних підприємства з використанням нечіткої логіки. Для досягнення поставленої мети потрібно виконати наступні задачі:

– провести огляд сучасних нечітких системи та алгоритмів контролю доступу;

– провести порівняльну характеристику існуючих нечітких систем;

– дослідити нечітку логіку та обрати відповідний алгоритм нечіткого виводу по базі знань;

– розробити алгоритм контролю доступу;

– побудувати функції належності для вхідних змінних нечіткої системи;

– сформувати базу продуктивних правил нечіткої системи;

– дослідити роботу реалізованої нечіткої системи.

Об’єкт дослідження – захист даних у інформаційних системах.

Предмет дослідження – алгоритми контролю доступу.

Методи досліджень. При проведенні огляду сучасних нечітких систем було використано метод аналізу. Для розробки алгоритму управління доступом було використано метод синтезу. Метод порівняння було використано для виявлення відмінностей між розробленим алгоритмом та вже існуючими.[5]

Наукова новизна одержаних результатів**.** Полягає у запропонованому алгоритмі контролю доступу до бази даних підприємства, який удосконалює уже відомі алгоритми шляхом виділення їхніх найкращих методів та їх об'єднання у одну систему для досягнення кращих результатів.

Розроблена модель нечіткого контролера контролю доступу до бази даних підприємства може бути використана для фізичної побудови системи і використання її на підприємствах.

Публікації та апробація. У матеріалах науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі» опубліковані результати наукового дослідження КМР (Тернопіль, 2020 р.) [6, 7]. У додатках наведено світлокопію матеріалів конференції.

У першому розділі магістерської роботи проведено аналіз сучасних нечітких систем управління базою даних. Зокрема було досліджено сучасні бази даних та їх особливості, проаналізовано застосування нечіткої логіки в системах управління базами даних, розглянуто вже існуючі нечіткі системи.

У другому розділі описано існуючий алгоритм нечіткої системи, роботу розробленого алгоритму та його порівняння із існуючим. Також було описано поняття бази знань та створено 35 правил для роботи нечіткої системи.

У третьому розділі створено модель системи контролю доступу до бази даних підприємства у середовищі MATLAB із використанням бібліотеки Fuzzy Logic Toolbox. Також у даному розділі описано проектування нечіткого контролера у середовищі Simulink та проведено перевірку правильності роботи розробленої системи. У додатку А наведено програмний код на мові MATLAB отриманий у результаті створення моделі системи.

1 СУЧАСНІ НЕЧІТКІ СИСТЕМИ УПРАВЛІННЯ БАЗОЮ ДАНИХ

1.1 Сучасні бази даних та їх особливості

Обсяг даних, які обробляються сьогодні, вищий, ніж будь-коли раніше, що робить підтримування баз даних більш складним. Проблеми починаються при виборі типу бази даних, яку ви збираєтеся використовувати у вашій програмі [8].

Раніше відповідь була б дійсно зрозумілою і простою: реляційна база даних вирішить дану проблему. Сьогодні, хоча реляційна база даних все-таки, можливо, вирішить вашу проблему, існують інші варіанти, які могли б краще підійти саме для даного типу використання інформації та полегшити наше життя. Необхідно знати варіанти та, звичайно, дійсно розуміти потреби конкретного випадку використання.[9]

Насправді популярність нереляційних баз даних зростає, більш ніж вдвічі збільшуючись за останні п’ять років. Однак лише один представник даного типу баз даних – MongoDB входить до першої п’ятірки в цілому (реляційні та нереляційні БД у поєднанні).

Залежно від типу, структури, моделі даних, сховища даних та призначеного випадку використання ваших даних, різні системи, ймовірно, будуть краще відповідати вашим потребам. Потрібна схема чи механізм запитів, ваші вимоги послідовності чи затримки або навіть швидкість транзакції (включаючи в режимі реального часу) також можуть вплинути на вибір. Наприклад, вбудована база даних для системи з локально збереженими даними про динамічну конфігурацію матиме зовсім інші вимоги, ніж операційна реляційна база даних, призначена для відстеження бронювання готельних номерів.

Отже, з чого починається вибір бази даних? Для початку роботи ми розглянемо як нереляційні (так звані NoSQL), так і реляційні бази даних (SQL). Це дозволить провести огляд обох екосистем та дізнатися слабка та сильні сторони кожної з них.

1.1.1 Реляційні бази даних

Реляційні бази даних та пов'язані з ними системи управління є більш відомими та зрозумілими, ніж їхні поплічники NoSQL. Реляційні бази даних з'явилися в 70-х роках для зберігання даних за схемою, яка дозволяє відображати дані у вигляді таблиць із рядками та стовпцями. Реляційну базу даних слід розглядати як сукупність таблиць, кожна із яких має схему, яка представляє фіксовані атрибути та типи даних, які матимуть елементи в таблиці. Всі реляційні БД надають функціональні можливості для читання, створення, оновлення та видалення даних, як правило, за допомогою операторів структурованої мови запитів (SQL) [10].

У таблицях реляційної бази даних пов'язані з ними ключі, які використовуються для ідентифікації конкретних стовпців або рядків таблиці та полегшення швидшого доступу до певної таблиці, рядка чи стовпця.

Цілісність даних викликає особливу стурбованість у реляційних базах даних, і RDBMS використовує ряд обмежень, щоб гарантувати надійність та точність даних, що містяться у таблицях.

Хоча існує багато реляційних баз даних, з часом саме ця п’ятірка стала найпопулярнішою:

1. База даних Oracle (зазвичай її називають Oracle RDBMS або просто як Oracle) – це багатомодельна система управління базами даних, яка виробляється та продається корпорацією Oracle.

2. MySQL – це система управління реляційними базами даних з відкритим кодом (RDBMS), заснована на структурованій мові запитів (SQL). MySQL працює практично на всіх платформах, включаючи UNIX та Windows.

3. Microsoft SQL Server – це RDBMS, який підтримує широкий спектр програм для обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середовищах.

4. PostgreSQL – часто є об'єктно-реляційною системою управління базами даних з акцентом на розширюваність та відповідність стандартам.

5. DB2 – це RDBMS, призначений для ефективного зберігання, аналізу та отримання даних.

Розглянемо переваги реляційних баз даних [11]:

–реляційні бази даних є добре задокументованими та старішими технологіями, а RDBMS продаються та підтримуються низкою створених корпорацій;

–стандарти SQL є чітко визначеними та загальноприйнятими;

–великий пул кваліфікованих розробників має досвід роботи з SQL;

–всі RDBMS задовольняють вимогам атомності, консистенції, ізоляції та довговічності.

Недоліками реляційних баз даних є:

–RDBMS не працюють добре з неструктурованими або напівструктурованими даними через обмеження схеми та типу і це робить їх непридатними для великих аналітичних навантажень або навантажень IoT подій;

–таблиці у вашій реляційній базі даних не обов'язково відображатимуть один до одного об'єкт або клас, що представляють однакові дані;

–при переміщенні однієї RDBMS на іншу, схеми та типи, як правило, повинні бути однаковими між вихідними та цільовими таблицями для міграції на роботу (обмеження схеми).

1.1.2 Нереляційні бази даних

Бази даних NoSQL стали популярною альтернативою реляційним базам даних, оскільки веб-додатки ставали все складнішими. Нереляційні бази даних можуть приймати найрізноманітніші форми. Однак критична відмінність NoSQL від реляційних баз даних полягає в тому, що схеми RDBMS жорстко визначають, як всі дані, вставлені в базу даних, повинні бути введені та складені, тоді як бази даних NoSQL можуть бути схематично-агностичними, дозволяючи зберігати та маніпулювати неструктурованими та напівструктурованими даними [12].

Нереляційні бази даних є різних типів. Зауважимо, що деякі типи можуть належати до декількох категорій. Наприклад, Couchbase – це і база даних документів, і сховище ключових значень.

Сховища ключових значень, такі як Redis та Amazon DynamoDB, є надзвичайно простими системами управління базами даних, які зберігають лише

пари ключових значень і забезпечують основну функціональність для отримання значення, пов'язаного з відомим ключем. Простота сховищ ключових значень робить ці системи управління базами даних особливо пристосованими до вбудованих баз даних, де збережені дані не є особливо складними, а швидкість має першорядне значення.

Широкі сховища стовпців, такі як Scylla та Cassandra та HBase – це схематично-агностичні системи, які дозволяють користувачам зберігати дані в сімействах стовпців або таблицях, окремий рядок яких можна вважати записом - багатовимірним ключовим значенням магазин. Ці рішення розроблені з метою досить масштабування для управління петабайтами даних на багатьох тисячах товарних серверів у масивній розподіленій системі. Хоча технічно без схем, широкі стовпці сховищ, як Scylla та Cassandra, використовують варіант SQL під назвою CQL для визначення даних та маніпулювання, що робить їх простими для тих, хто вже знайомий з RDBMS.

Сховища документів, включаючи MongoDB та Couchbase, - це схеми без схем, які зберігають дані у вигляді документів JSON. Сховища документів схожі на сховища ключових значень або широкі стовпці, але назва документа - це ключ, а вміст документа – як би там не було. У сховищі документів окремі записи не потребують рівномірної структури, можуть містити багато різних типів значень і можуть бути вкладеними. Ця гнучкість робить їх особливо придатними для управління напівструктурованими даними в розподілених системах.

Бази даних графіків, такі як Neo4J та Datastax Enterprise Graph, представляють дані як мережу пов'язаних вузлів або об'єктів, щоб полегшити візуалізацію даних та аналітику графіків. Вузол або об’єкт у базі даних графіків містять дані у вільній формі, які пов'язані між собою зв'язками та згруповані відповідно до міток. Програмне забезпечення для управління базами даних (СУБД), орієнтоване на графіки, розроблене з акцентом на ілюструванні зв’язків між точками даних. Як результат, бази даних графіків зазвичай використовуються, коли аналіз взаємозв'язків між неоднорідними точками даних є кінцевою метою системи, наприклад, у запобіганні шахрайствам, вдосконаленій діяльності підприємства або оригінальному графіку друзів Facebook.

Пошукові системи, такі як Elasticsearch, Splunk і Solr, зберігають дані, використовуючи JSON – документи без схем. Вони схожі на сховища документів, але з більшим акцентом роблять доступ до неструктурованих або напівструктурованих даних легко доступними за допомогою текстового пошуку з рядками різної складності.

Переваги нереляційних баз даних[13]:

–моделі даних без схем є більш гнучкими та легшими в управлінні;

–бази даних NoSQL, як правило, є більш горизонтально масштабованими та відмовними;

–дані можна легко розподілити по різних вузлах, щоб поліпшити доступність та/або переносимість розділів, можна вибрати, що дані про деякі вузли будуть з часом узгоджені.

Недоліки нереляційних баз даних:

–бази даних NoSQL, як правило, менш широко прийняті, ніж рішення RDBMS, тому часто потрібна спеціальна експертиза;

–існує діапазон форматів і обмежень, характерних для кожного типу бази даних.

1.1.3 Як правильно вибрати базу даних

Якщо відповідність ACID (атомність, міцність, послідовність та довговічність) є першочерговим завданням, то варто надати перевагу використання RDBMS [14]. Якщо використовується широко розповсюджена система і можна погодитися з можливою консистенцією на деяких вузлах / перегородках, ви можете розглянути базу зберігання широких стовпців, таких як Scylla та Cassandra. Якщо вхідні дані особливо неоднорідні і їх важко інкапсулювати відповідно до схеми нормалізації, варто використовувати СУБД NoSQL. Якщо головна мета – масштабувати вертикально, потрібно використовувати RDBMS. І навпаки, якщо масштабувати горизонтально, може бути кращою СУБД NoSQL.

1.1.4 Огляд найпопулярніших сучасних систем баз даних

У всіх випадках перераховані нижче бази даних підтримують як формати даних SQL, так і NoSQL. Однак нереляційні бази даних, такі як PostgreSQL і MongoDB, як правило, краще працюють з форматами NoSQL. Реляційні бази даних, такі як Oracle, Microsoft SQL Server та MySQL, працюють краще з чисто форматами SQL.

Найпопулярніші бази даних, які використовують компанії у 2020 році:

1. Oracle;
2. MySQL;
3. Microsoft SQL Server;
4. PostgreSQL;
5. MongoDB.

Oracle надає якісні рішення для баз даних з 1970-х років. Остання версія бази даних Oracle була розроблена для інтеграції з хмарними системами, і вона дозволяє управляти масивними базами даних з мільярдами записів. Більше того, Oracle дозволяє організовувати дані за допомогою «сіткової рамки», і він використовує динамічне маскування даних для додаткового рівня безпеки. Традиційно Oracle пропонує рішення RDMBS, але тепер ви також можете знайти рішення SQL і NoSQL.

Можна виділити дві переваги використання Oracle:

1. Найсучасніша технологія. Oracle відомий тим, що знаходиться на передовій технології баз даних. Вони мають давню репутацію за те, щоб забезпечити якість – разом з найновішими характеристиками та інноваціями - до своєї клієнтської бази.

2. Широкий спектр рішень. Oracle пропонує широкий набір інструментів та рішень, які можуть вирішити практично будь-які інформаційні проблеми, з якими ви стикаєтесь.

Основні недоліки використання Oracle:

1. Дороге рішення. Oracle, як правило, є дорогим рішенням, яке менші організації, що не є підприємствами, можуть не дозволити собі.

2. Можливо, знадобиться оновлення системи. Поточних специфікацій системи може бути недостатньо для впровадження Oracle. Багатьом компаніям доводиться модернізувати обладнання, перш ніж використовувати рішення Oracle.

MySQL - це безкоштовне рішення з відкритим кодом з RDBMS, яким володіє та керує Oracle. Незважаючи на те, що це безкоштовно, MySQL має перевагу від частої безпеки та оновлень функцій. Комерційні та підприємства можуть перейти на платну версію MySQL, щоб отримати додаткові функції та підтримку користувачів. Хоча MySQL не підтримував NoSQL в минулому, починаючи з версії 8, він надає підтримку NoSQL, щоб конкурувати з іншими рішеннями, такими як PostgreSQL.

Безкоштовна версія MySQL орієнтована на швидкість та надійність замість усіх дзвіночків, що зустрічаються у багатьох сучасних проектах баз даних. Можна виділити наступні ознаки MySQL:

–використовує лише прості команди SQL;

–не підтримує XML;

–дозволяє використовувати лише два типи символів: CHAR та VARCHAR;

–не дозволяє додаткові резервні копії;

–не вистачає інших функцій, пропонованих його конкурентами.

Тим не менш, MySQL надає вам можливість використовувати безліч різних типів двигунів зберігання даних, що забезпечує гнучкість для роботи з найрізноманітнішими типами даних. Крім того, зручний інтерфейс та пакетні команди пропонують можливість ефективно керувати, редагувати та обробляти велику кількість інформації.

До переваг використання MySQL належать [16]:

–це безкоштовно, як рішення з відкритим кодом RDBMS, MySQL вільний у використанні будь-яким способом;

–відмінно підходить для будь-якої організації розміру: MySQL – це відмінне рішення для підприємств на рівні підприємств та для малих стартап-компаній;

–доступні різні інтерфейси користувача;

–високо сумісний з іншими системами, MySQL має репутацію сумісності з багатьма іншими системами баз даних.

До недоліків використання MySQL можна віднести:

–відсутні функції, поширені в інших RDBMS. Оскільки MySQL надає пріоритет швидкості та спритності над функціями, ви можете виявити, що в ньому відсутні деякі стандартні функції, знайдені в інших рішеннях, наприклад, можливість створювати додаткові резервні копії, наприклад;

–проблеми з якісною підтримкою, окільки це безкоштовне рішення то не буде команди, яка має в своєму розпорядженні команду підтримки, яка допоможе вам вирішити проблеми, але при цьому, у MySQL є активне волонтерське співтовариство, корисні форуми та багато документації, яка може бути корисною для відповіді на більшість питань.

MySQL є особливо цінним рішенням RDBMS для підприємств, які потребують рішення з можливостями на рівні підприємства, але працюють в умовах жорстких бюджетних обмежень. Як безкоштовний, але надзвичайно потужний і надійний сучасний RDBMS, ви не можете помилитися з MySQL.

Сервер Microsoft SQL - це двигун бази даних, сумісний як з локальними, так і з хмарними серверами. Крім того, невдовзі після випуску Microsoft SQL Server 2016 компанія Microsoft оприлюднила версію Linux для двигуна бази даних на додаток до версії Windows.

Microsoft SQL Server має різні версії, з яких можна вибрати. У цій статті детально розглядаються відмінності в кожній, хоча в більшості розробників віддають перевагу останній версію (2016). Цікавим доповненням у випуску 2016 року є тимчасова підтримка даних. Це дозволяє відслідковувати зміни інформації в базі даних з часом.

Тимчасові дані тісно пов'язані з періодом часу та використовуються для обробки даних, що змінюються за часом. Він надає можливість переглядати тенденції даних, типи змін даних та загальну еволюцію даних у вашій базі даних. З часової таблиці, значення кожного запису в будь-який момент часу можна визначити, а не просто поточне значення кожного запису.

Також Microsoft SQL Server на даний момент підтримує динамічне маскування даних, що підвищує безпеку, маскуючи конфіденційну інформацію від непривілейованих користувачів.

Ця функція допомагає запобігти несанкціонованому доступу до конфіденційних даних, дозволяючи клієнтам визначити, яку частину конфіденційних даних розкривати з мінімальним впливом на рівень додатків.

До переваг використання Microsoft SQL Server належать:

–мобільність. цей механізм баз даних дозволяє отримувати доступ до графіки та візуальної панелі інформаційної панелі через мобільні пристрої;

–інтеграція з продуктами Microsoft. Компанії, які значною мірою покладаються на продукти Microsoft, будуть насолоджуватися тим, як SQL Server легко інтегрується з цими додатками;

–швидкість. Microsoft SQL Server створив репутацію швидкості та стабільності.

Недоліки використання Microsoft SQL Server:

–дороговизна. Враховуючи, що доступно безліч безкоштовних механізмів баз даних, вартість Microsoft SQL Server висока – це понад 14 000 доларів США за одну ліцензію на рівні підприємства на ядро;

–потрібно багато ресурсів. Цей важкий RDBMS може вимагати придбання кращого обладнання.

PostgreSQL – це безкоштовний механізм бази даних з відкритим кодом і необмеженими можливостями масштабування. Якщо ви подивитесь на розділ вище щодо NoSQL або "нереляційних систем баз даних", PostgreSQL – це відмінний варіант, який підтримує як реляційні, так і нереляційні формати.

Як надійний СУБД, який існує з початку 1990-х, PostgreSQL налічую велику базу відданих користувачів і останні два роки поспіль виграв нагороду «База даних року» (як результат, він є найбільш розвинутим механізм бази даних).

Цікавою особливістю PostgreSQL є її історія роботи як зі структурованими (SQL), так і з неструктурованими (NoSQL) даними. PostgreSQL є дуже розширюваним, завдяки роботі з керованим каталогом. Він не просто зберігає інформацію для ідентифікації таблиць і стовпців; це дозволяє визначати типи даних, типи індексу та функціональні мови. Він також сумісний з більшістю операційних систем, включаючи платформи Linux, і добре інтегрується з даними з

широкого спектру баз даних. Нарешті, PostgreSQL працює з виділеними серверами та хмарними серверами.

Незважаючи на те, що це неприбуткова, безкоштовна система баз даних, велика мережа відданих послідовників та волонтерів надає безкоштовну підтримку користувачам та регулярно оновлює систему. Наприклад, нещодавнє оновлення (PostgreSQL 9.5) збільшило розмір обсягів даних та кількість одночасних користувачів, які підтримує система.

Ось деякі з найважливіших переваг PostgreSQL:

–Більше можливостей. PostgreSQL має набагато більше функцій, ніж інші СУБД. Ці додаткові функції включають наслідування таблиць, багатий набір типів даних (включаючи вбудовану підтримку JSON), можливість визначення стовпця як "масиву" типів стовпців та інші функції.

–Високо сумісний з ACID. PostgreSQL належить до СУБД, найбільш сумісних з ACID, тому цей механізм баз даних може стати відмінним вибором, якщо ви стурбовані цілісністю даних.

–Масова масштабованість. PostgreSQL може працювати з масивними таблицями баз даних.

Ось кілька можливих недоліків використання PostgreSQL[17]:

–Відсутність документації. PostgreSQL не має кращої документації порівняно з альтернативними двигунами бази даних. Тому, якщо у вас виникнуть труднощі, вам може знадобитися звернутися за допомогою до приватної служби підтримки PostgreSQL або спробувати свою удачу на форумах підтримки спільноти.

–Проблеми зі швидкістю під час операцій лише для читання. PostgreSQL надає перевагу операціям читання-запису для даних, які потребують перевірки, але уповільнення може статися під час роботи з операціями лише для читання.

Оскільки PostgreSQL є абсолютно безкоштовним і масштабованим, це відмінне рішення для компаній будь-якого розміру, тим, хто має бюджетні обмеження, не потрібно турбуватися про витрати. Крім того, якщо ваш випадок використання може скористатися СУБД із вбудованою підтримкою JSON, PostgreSQL, ймовірно, для вас.

MongoDB – це вільний СУБД із відкритим кодом, створений спеціально для додатків, що використовують неструктуровані дані. Оскільки більшість СУБД були побудовані для структурованих даних (навіть якщо додатки дозволяють їм обробляти нереляційні дані зараз), MongoDB часто виграє там, де інші СУБД виходять з ладу. MongoDB працює і зі структурованими даними, але оскільки цей механізм бази даних не був розроблений для реляційних даних, можливе уповільнення продуктивності.

MongoDB з'єднує нереляційні бази даних із програмами, використовуючи широкий спектр драйверів (на основі мови програмування програми). Останні версії MongoDB включають в себе підключаються двигуни для зберігання даних. Також доступні оновлені функції пошуку тексту, а також функції часткової індексації, які можуть допомогти в продуктивності.

Переваги використання MongoDB:

–підтримка NoSQL. ця СУБД була спеціально створена для JSON та NoSQL;

–висока гнучкість. Оскільки MongoDB зберігає та керує будь-якою інформацією, розробники стикаються з меншими обмеженнями при включенні даних у базу даних MongoDB;

–чудово підходить для програм, включаючи веб-додатки. MongoDB став популярною СУБД для веб-додатків.

Недоліки використання MongoDB:

–немає запитів SQL. MongoDB не приймає SQL запити;

–складність налаштування. MongoDB вимагає часу та більше досвіду для правильної настройки, ніж інші рішення;

–відсутність безпеки. Стандартні налаштування MongoDB, як правило, не дуже безпечні, необхідні додаткові кроки для захисту цієї бази даних.

1.2 Застосування нечіткої логіки в системах управління базами даних

Класичні мови запитів дають лише точний або нульовий результат і не

повідомляти про приблизно подібні результати. У багатьох випадках користувач виконує сценарій із серією запитів з різними значеннями параметрів для отримання потрібної інформації. Такий підхід позбавлений когнітивної підтримки створення висновку із баз даних.

У минулому уже були спроби розширити можливості класичних мов запитів шляхом інтеграції їх з нечіткою логічною системою в нечітких запитах. Таке об’єднання двох методів дозволяють користувачеві переглядати інформацію в базах даних, яка надає пізнавальну підтримку для зменшення перевантаження інформацією дозволяючи користувачеві використовувати лінгвістичні змінні для формування запиту на видобуток даних. Існує концепція дворівневої бази даних. Схеми адміністратора призначені для кращого управління даними у корпоративному обчислювальному середовищі та для роботи із чутливою до контексту інформацією. Адміністратор бази даних несе відповідальність за збереження чіткості БД. База даних Crisp являється базою над якою будується нечітка база даних.

Поняття теорії нечітких множин, введене Лотфі А. Заде у 1965 р. являє собою узагальнення класичної теорії множин. Елемент у нечіткому наборі являє собою ступінь його належності в нечіткій множині. Нехай U - всесвіт дискурсу, x - це загальний елемент U такого що:

$$U = \{x, x\ U\} \tag{1.1}$$

Тоді характеризується нечітка множина A в U за функцією членства μA(x), яка пов'язує x реальне число в інтервалі [0,1], де μA (x) являє собою ступінь приналежності x в A. Більше значення μA (x) до одиниці, вищий ступінь належності x у множині A. У випадку функції членства класичної теорії множин містить будь-яке з двох значень, тобто {0, 1} [18].

Коли A - нечітка множина, а x - відповідний об'єкт, твердження "X є членом A" не обов'язково є істиною або хибою, як вимагається двійковою логікою, це може бути правдою лише для певного ступеню. Ступінь, до якої x насправді є членом A – дійсне число в інтервалі [0, 1].

$$F = \{(x, \mu A(x)) \mid x X\}, \mu A(x) \quad (1.2)$$

Теоретично, якщо X - це сукупність предметів, позначених загалом через x, то нечіткий набір F у X - це набір упорядкованих пар, функція 1.2 називається функцією членства (або ступенем членство) x у F. Діапазон функції членства - це підмножина негативних дійсних чисел, супремусом яких є кінець (рисунок 1.1).

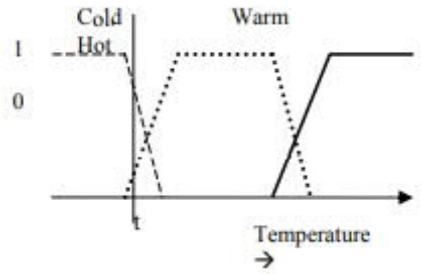

Рисунок 1.1 – Графік членства функції належності визначення температури

Кожна вхідна змінна у нечіткій системі має належність функції, яка може бути побудована на графіку членства, як на рисунку 1.1. Стани холодного, теплого та гарячого представлені значеннями функції приналежності до температури. Вертикальна лінія на зображенні показує, що температура t дорівнює "Не гаряче" (оскільки гаряча лінія знаходиться на нулі). Холодна лінія описує цю температуру як "дуже холодна", а тепла лінія як "злегка тепла".

Нечітка логіка імітує логіку мозку людиною. Це забезпечує простий спосіб дійти певного висновку на основі неясної, неточної, неоднозначної або відсутньої вхідної інформації. Нечітка логіка не тільки забезпечує змістовне і потужне подання вимірювання невизначеностей, але також передбачає змістовне подання розпливчастого поняття, яке виражене природною мовою. Нечітка логіка дозволяє нам використовувати лінгвістичні змінні, такі як гаряче, холодне, молодий, добрий тощо. Це також дозволяє нам використовувати прикметники у змінних, щоб додатково відфільтрувати елементи, такі як дуже гарячі, менш холодні, занадто

молодий і т.д.. Це призвело до переходу від чіткої реляційної моделі баз даних до сфери нечітких моделей бази даних. Можна виділити два підходи до реалізації:

–дозволити нечіткі запити в базах даних, що зберігають чіткі дані.

–додавання нечіткої інформації до баз даних.

Найпростіший спосіб реалізувати пошук інформації, яка є грубо неточна є використання моделі бази даних що зберігає чіткі записи, але дозволяє використання нечітких запитів з використання інтерактивної програми графічного інтерфейсу користувача. Класичні мови запитів розроблені для отримання даних, які повністю задовільняють чіткому обмеженню напр. "SELECT * FROM table WHERE age < 25 AND salary > 70000". Цей вид запиту отримає всі кортежі в базі даних, які задовольняють задане обмеження, але якщо база даних не містить таких кортежів, тоді немає можливості витягти дані, які дещо схожі на бажаний вихід. Традиційні мови запитів не в змозі отримати дані які схожі на ті, які бажає користувачем. Тому було введено поняття нечіткої логіки в мові запитів. Поєднання нечіткої мови запитів дозволяє користувачеві витягувати бажану інформацію, навіть якщо користувач не надає точні найменування змінних у запиті. Нечіткий запит буде виглядають як "SELECT * FROM TABLE, WHERE young age AND high salary". Нечіткість, відповідна віку, показана на рисунку 1.2.

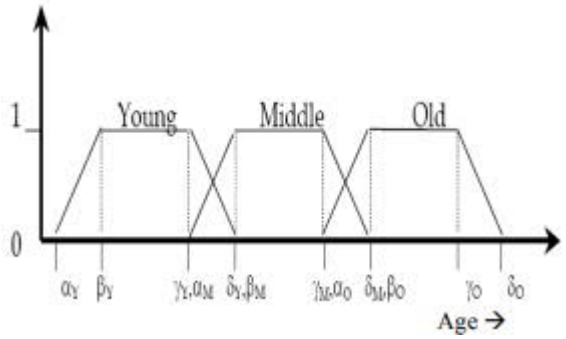

Рисунок 1.2 – Функції належності змінної віку у нечітких запитах до бази даних

Значення членства, відповідне віку, позначається через змінні альфа, бета та гами. Цей запит витягує ті кортежі, де віковий атрибут менший а атрибут зарплати високий. Нечіткий запит знімає обмеження, накладені на вибір умови, виконуючи нечіткий алгоритм. Дані, видобуті з використанням нечіткого запиту мають

найбільшу схожість з інформацією, яку бажає бачити користувач порівняно з пустим результатом.

1.3 Аналіз існуючих нечітких систем

Методи штучного інтелекту, такі як нечіткі та нейро-нечіткі системи, в даний час використовуються для вирішення проблем комп'ютерної інженерії в режимі реального часу. Використання нечіткої логіки дозволяє природним чином виразити поняття, що використовуються експертами та користувачами.

Існує експертна система функціональної діагностики, яка базується на знаннях у формі нейронної мережі. Поточні значення діагностичних параметрів вимірюються датчиками.

Гібридна експертна діагностична система з базою даних нейрофізіологічної мережі підтримує рішення в ситуації, коли алгоритм діагностики невідомий і формується з вихідних даних у вигляді виробничих правил. Датчики використовуються для автоматизації процесу накопичення знань в експертній системі.

Нечітка логіка часто використовується для створення нейро-нечітких моделей. За допомогою нечітких систем ви можете аналізувати програмне забезпечення [19] та обладнання [20], а також досліджувати мережевий трафік [21].

У роботі [22] пропонується використовувати експертну діагностичну систему для аналізу технічного стану комп'ютерної системи. Математичний апарат, що дозволяє контролювати прогнозовану оцінку стану діагностичного об'єкта (апаратного, програмного забезпечення або персоналу), є нечіткою логікою.

На етапі підготовки діагностичного експерименту пропонується описати діагностичні особливості комп’ютерної системи за допомогою лінгвістичних змінних, щоб знання та досвід експерта могли бути використані у звичній формі. Ця стаття рекомендує використовувати експертну діагностичну систему для аналізу технічного стану комп’ютерної системи.

Дослідження [22] представляє метод, який зменшує проблеми при діагностиці апаратних несправностей. Пропонується інтелектуальна експертна система, яка використовує технологію, засновану на правилах, для діагностики апаратних несправностей. Користувачеві або комп'ютерному майстру не потрібно тестувати окремі комп'ютерні пристрої окремо.

Все, що вам потрібно зробити, це ввести апаратне забезпечення комп’ютера та симптоми в систему, а потім діагностувати апаратне забезпечення.

Розробка діагностичної діагностики обчислювальних пристроїв з використанням методу, заснованого на правилах, базується на методології розробки експертної системи, яка складається з восьми етапів: дослідження та аналіз, концептуалізація, оцінка проблем, набуття та аналіз знань, проектування та впровадження, тестування, управління документацією.

Система показує можливі причини та пропонує рішення. Правила запропонованої експертної системи вказані у формі інструкцій "якщо-тоді". Категорії цієї системи: аудіо, жорсткий диск, клавіатура, миша, блок живлення, процесор, завантаження, послідовний ATA, USB-пристрій, принтер, материнська плата, процесор, оперативна пам'ять, периферія, BIOS, відеомонітор та адаптер, привід DVD та запис на DVD / CD.

Мета побудови цієї простої експертної системи - допомогти користувачам комп’ютерів у діагностиці проблем із апаратним обладнанням. Крім того, ця програма допоможе користувачам усунути деякі важливі апаратні проблеми або виконати більш комплексний сеанс усунення несправностей, перш ніж звертатися до служби підтримки або професіоналів.

Однак у цій роботі розроблена нечітка система може споживати великі ресурси ПК, оскільки їй доводиться постійно аналізувати стан обладнання в режимі реального часу.

Перспектива розвитку нейро-нечіткої системи полягає у розробці експертної системи, що є складним завданням, оскільки існує експертна система, яка є системою високого рівня, яка має справу зі знаннями, з якими важко боротися . У [23] описано комп'ютерне проектування інтелектуальних систем із використанням сучасної технології штучного інтелекту.

Нечітка логіка, відома як "нечітке управління", відома системним інженерам як зручний інструмент для програмування та моніторингу програм управління процесами. Аналогічно звичайним інструментам управління процесом, нечіткі логічні системи можуть бути використані для опису контурів управління та участі у розрахунку ефектів управління відповідно до завдання для одного або декількох вимірювань [24].

Правила нечіткої логіки допускають наступне:

–застосування існуючого досвіду управління;

–використовуйте гнучкі правила, якщо неможливо точно моделювати систему традиційними засобами.

Якість управління покращується завдяки:

–саморегулювання системи управління;

–прогнозована зміна початкового впливу (попереджувальна функція) внаслідок подій, які неможливо врахувати за допомогою звичайних методів управління.

Кількість додатків, заснованих на цих методах управління, постійно збільшується для безперервних процесів, для додатків пакетної обробки, а також для автоматизованих систем.

Нечітка логіка була описана і сформульована як метод програмування через її використання в цій галузі. Це дозволяє систематизувати емпіричні знання та застосовувати їх для управління процесами у разі виникнення труднощів із застосуванням класичних методів управління [25].

Теорія нечіткої логіки дозволяє описати набори методів управління, які легко застосувати до реальної системи, та врахувати досвід операторів та технологів для динамічного управління процесами.

Таким чином ви можете описати нечітку логіку окремих частин виробничого процесу, напр. В. Ініціалізація та встановлення параметрів.

У разі технічних проблем зазвичай використовується механізм нечіткого виведення Мамдані, який може бути реалізований в середовищі Matlab. Метод Мамдані був одним із перших, який був побудований з використанням теорії нечітких множин. Його запропонував Ібрагім Мамдані в 1975 році.

Метод Мамдані - це система нечіткого висновку та система, яка використовує теорію нечітких множин для відображення входів та виходів [26].

Метод складається з наступних етапів:

– формування бази правил;

– фазифікація;

– агрегування підумов;

– активізація підвисновків;

– акумулювання висновків;

– дефазифікація.

Кожен етап виконується по черзі, і кожен наступний етап отримує вхідні значення, отримані в результаті попереднього. Метод Мамдані використовує певну базу правил для вхідних значень.

Для прикладу, правила можуть бути представлені у такому вигляді - якщо

$$x \in Y_1, \text{ тоді } z \in N_1,$$

де x – вхідне значення,

z – вихідне значення,

Y і N – нечіткі множини.

Також фіксується захоплення вхідними змінними. Вхідні дані є регулярною базою та найкращим масивом входів $A = \{a_1, ..., a_n\}$. Цей масив містить значення всіх входів. Для кожної з умов існує значення $b_i = \mu(a_i)$. Таким чином, існує багато значень $b_i (i = 1...k)$.

На фазі агрегуючих умов визначається ступінь істинності умов для кожного з правил системи нечітких умовиводів [27]. Тобто для кожного стану існує мінімальне значення істинності всієї вашої підсвідомості. Формально це виглядає так:

$$c_j = \min\{b_i\}, \tag{1.3}$$

де $j = 1...q$;

i – число з безліччю номерів підумов, в яких бере участь j-а вхідна змінна.

На фазі активації висновків відбувається перехід від умов до підзаключень. Для кожного підтипу існує ступінь істинності $d_i = c_i * F_i$, де *(i = 1..q).*

Потім кожен і-тий підвихід порівнюється з набором D_i з новою функцією членства. Його значення визначається принаймні d_i та значенням функції членства терміна з підряду. Цей метод відомий як мінімальна активація і формально записується наступним чином (формула 1.4):

$$\mu_i^{'} = \min\{d_i, \mu_i(x)\}, \tag{1.4}$$

де $\mu_i^{'}(x)$ – «активована» функція належності;

$\mu_i(x)$ – функція належності терму;

d_i – ступінь істинності i-го підвиводу.

Тому метою цього етапу є отримання суми "активованих" нечітких наборів D_і для кожного з підвиводів у базі правил *(i = 1…q).*

На етапі накопичення результатам надається нечіткий набір (або їх комбінація) для кожної з вихідних змінних. Цей крок виконується наступним чином: Порівнюється і-та вихідна змінна для об'єднання множин $E_i = \mathrm{U} D_j$. Де j - кількість підз'єднань, у яких бере участь і-та вихідна змінна *(i = 1..s).*

Об'єднання двох нечітких множин - це третій нечіткий набір з наступною функцією належності (формула 1.5):

$$\mu_i^{'}(x) = max\{\mu_1(x), \mu_2(x)\}, \tag{1.5}$$

де $\mu_1(x), \mu_2(x)$ – функції приналежності поєднуваних множин [24].

Метою дефазифікації є отримання кількісного значення для кожної з оригінальних змінних мови. На практиці це відбувається наступним чином: Ми розглядаємо і-ту початкову змінну та відповідний набір $E_i(1..s)$.

Далі, використання методу дефазифікації є кінцевим кількісним значенням вихідної змінної. У цій реалізації алгоритму використовується метод центроїд, в якому значення і-ї вихідної змінної обчислюється за формулою 1.6:

$$y_i = \frac{\int_{\min}^{\max} x * \mu_i(x) dx}{\int_{\min}^{\max} \mu_i(x) dx}, \quad (1.6)$$

де $\mu_i(x)$ – функція приналежності відповідної нечіткої множини E_i;

min і max – кордони універсуму нечітких змінних;

y_i – результат дефазифікації.

1.4 Висновки до розділу

У першому розділі було проведено аналітичний огляд існуючих видів баз даних, аналіз застосування нечіткої логіки в системах управління базами даних. Також було проаналізовано існуючі нечіткі системи.

У даній магістерській роботі поставлена задача розробки алгоритму контролю доступу до бази даних підприємства. Дерево рішень процесу розробки зображене на рисунку 1.3.

Виходячи з дерева рішень зображеного на рисунку 1.3 можна побачити наступні етапи розробки алгоритму контролю доступу до бази даних підприємства:

1. Проведення аналізу алгоритму контролю доступу з використанням нечіткої логіки та системою оцінки ризику.
2. Розробка власного алгоритму контролю доступу до бази даних підприємства.
3. Розробка правил нечіткої бази знань.
4. Розробка моделі системи у середовищі MATLAB
5. Розробка нечіткого контролера у середовищі Simulink.
6. Тестування та верифікація працездатності системи.

Рисунок 1.3 – Дерево рішень процесу розробки алгоритму

У наступному розділі магістерської роботи буде проведено аналіз існуючого алгоритму контролю доступу, буде здійснено розробку власного алгоритму та нечіткої бази знань.

2 АЛГОРИТМ НЕЧІТКОГО УПРАВЛІННЯ СИСТЕМОЮ ДОСТУПУ ДО БАЗИ ДАНИХ

2.1 Алгоритм контролю доступу з використанням нечіткої логіки та системи оцінки ризику

Ризик безпеки, пов'язаний із запитом доступу, є складовим елементом моделі на основі ризику. Ця модель проводить аналіз ризику для оцінки значення ризику, пов'язаного із запитом на доступ. Потім оцінене значення ризику порівнюється з політикою ризику для визначення рішення про доступ. Модель на основі ризику вирішує декілька питань, пов'язаних із гнучкістю доступу до системних ресурсів [28].

Адаптивну модель управління доступом на основі ризику показано на рисунку 2.1. Ця модель збирає реальну та контекстну інформацію, що стосується запиту на доступ, для визначення рішення про надання доступу. Пропонована модель налічує чотири входи:

1. Контекст користувача
2. Чутливість до ресурсів.
3. Серйозність дій.
4. Історія ризику.

Ці дані використовуються модулем оцінки ризику, який відповідає за оцінку загального значення ризику, пов'язаного із запитом на доступ. Далі слід порівняти оцінене значення з політикою ризику для прийняття рішень про доступ. Існує тільки два рішення - наданням або заборона доступу. Щоб включити можливості виявлення ненормальності, запропоновано розумні контракти для відстеження та моніторингу діяльності користувачів протягом сеансів доступу, щоб запобігти зловмисним атакам та розкриттю конфіденційної інформації.

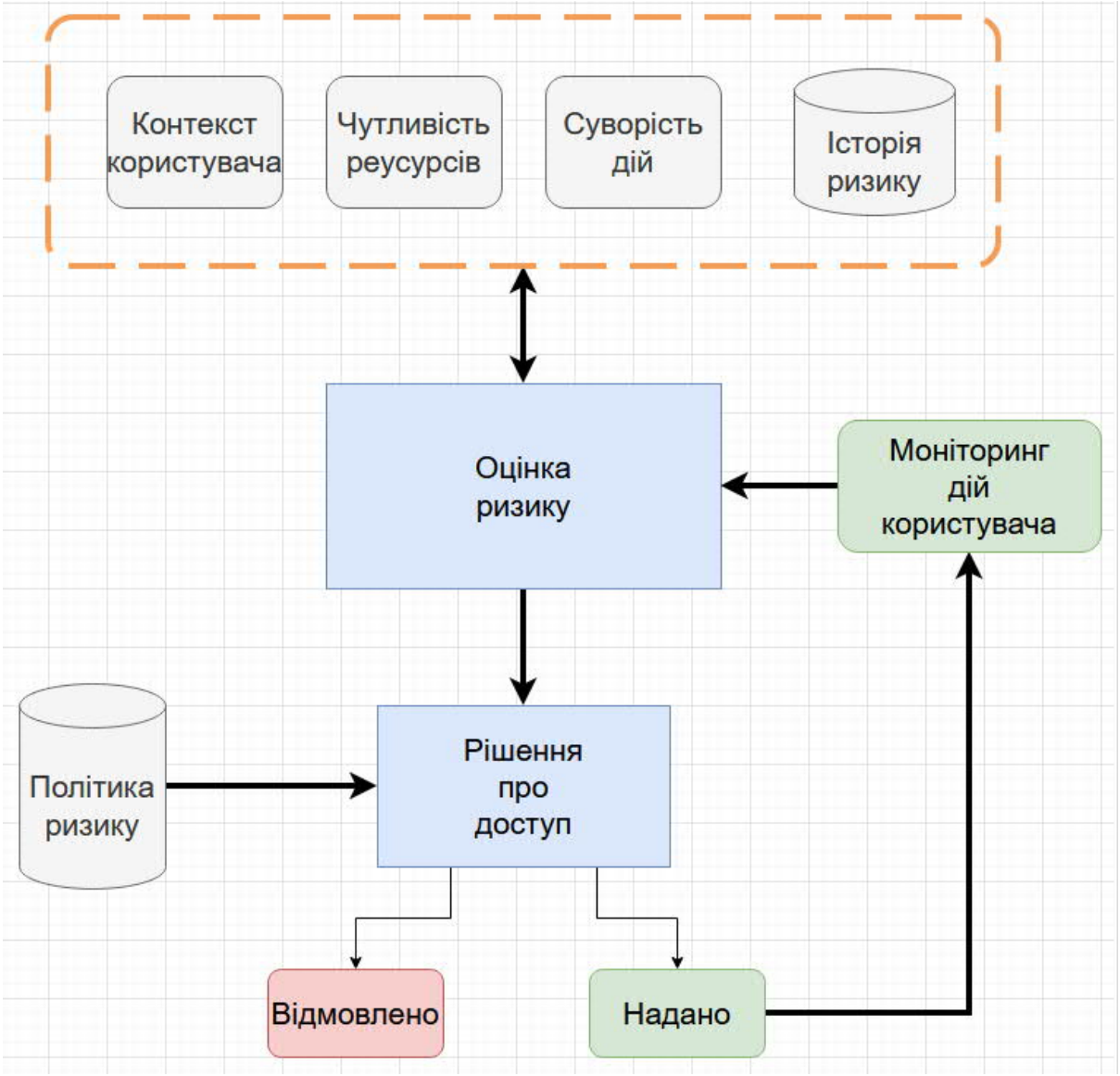

Рисунок 2.1 – Адаптивна модель управління доступом на основі ризику

Запропонована модель використовує функції реального часу, пов'язані з користувачем, для представлення того, що називається контекстом користувача. Ці функції описують екологічні атрибути, пов'язані з користувачем / агентом під час подання запиту на доступ. Значення ризику безпеки відображається в різних контекстах користувачів. Місце і час - найпоширеніші контексти користувачів.

Чутливість до ресурсів описує рівень важливості даних, на які може бути атаковано неналежним чином. Визначення рівнів чутливості різних типів даних - це повністю суб'єктивна операція, яка залежить лише від власника даних, який вирішить, яка цінніша за інших. Щоб гарантувати ефективну класифікацію чутливості, для категоризації даних слід найняти фахівців із безпеки. Різні дані

мають різний рівень чутливості, тому даним призначається метрика чутливості для диференціювання даних різних типів в базі даних.

Для кожного запиту доступу користувач, що запитує, визначає дію, яку він хоче виконати на певному ресурсі. Суворість дій використовується для опису впливу дій на системні ресурси. Експерти з безпеки можуть класифікувати різні дії та призначити показник суворості для кожної дії. Отже, метрика ризику буде пов’язана з кожною дією на певному ресурсі. Крім того, історія ризику користувача описує попередні значення ризику користувача щодо різних дій, що виконуються користувачем. Він відображає моделі поведінки попередніх користувачів, щоб визнати добрих та зловмисних користувачів.

Однією з основних частин моделі на основі ризику є модуль оцінки ризику. Цей модуль приймає вхідні фактори ризику для вимірювання значення ризику стосовно кожного запиту доступу. Можлива мета - побудувати ефективний метод оцінки ризику, який використовує інформацію в режимі реального часу, щоб дати точне значення ризику для контролю операцій доступу в системі. Оціночне значення ризику порівнюється з політикою ризику для визначення рішення про доступ. Політика ризику будується для визначення меж доступу та ситуацій, коли доступ може бути наданий чи заборонений. Він визначає порогове значення таким, що якщо значення ризику запиту доступу нижче значення порогового ризику, доступ буде наданий, інакше в доступі буде відмовлено.

Потік процесу запропонованої моделі на основі ризику показаний на рисунку 2.2. Він починається, коли користувач надсилає запит доступу до менеджера управління доступом. Користувач, який запитує, повинен вказати ресурс або дані, до яких потрібно отримати доступ, і дії, які слід виконати. Менеджер управління доступом збирає контекстну інформацію, пов’язану з запитуючим користувачем, створюючи запит доступу, такі як місцеположення та час, із рівнем чутливості ресурсу, до якого можна отримати доступ, серйозністю дії, що виконується, як зазначено в запиті доступу, і попередні записи історії ризиків запитуючого користувача.

Модуль оцінки ризику використовує зібрану інформацію для вимірювання значення ризику, пов'язаного із запитуючим користувачем. Далі йде порівняння

вимірюваної величини ризику з політикою ризику для визначення рішення про доступ. Якщо значення ризику менше порогового значення ризику, визначеного в політиках щодо ризику, доступ буде наданий, інакше в доступі буде відмовлено.

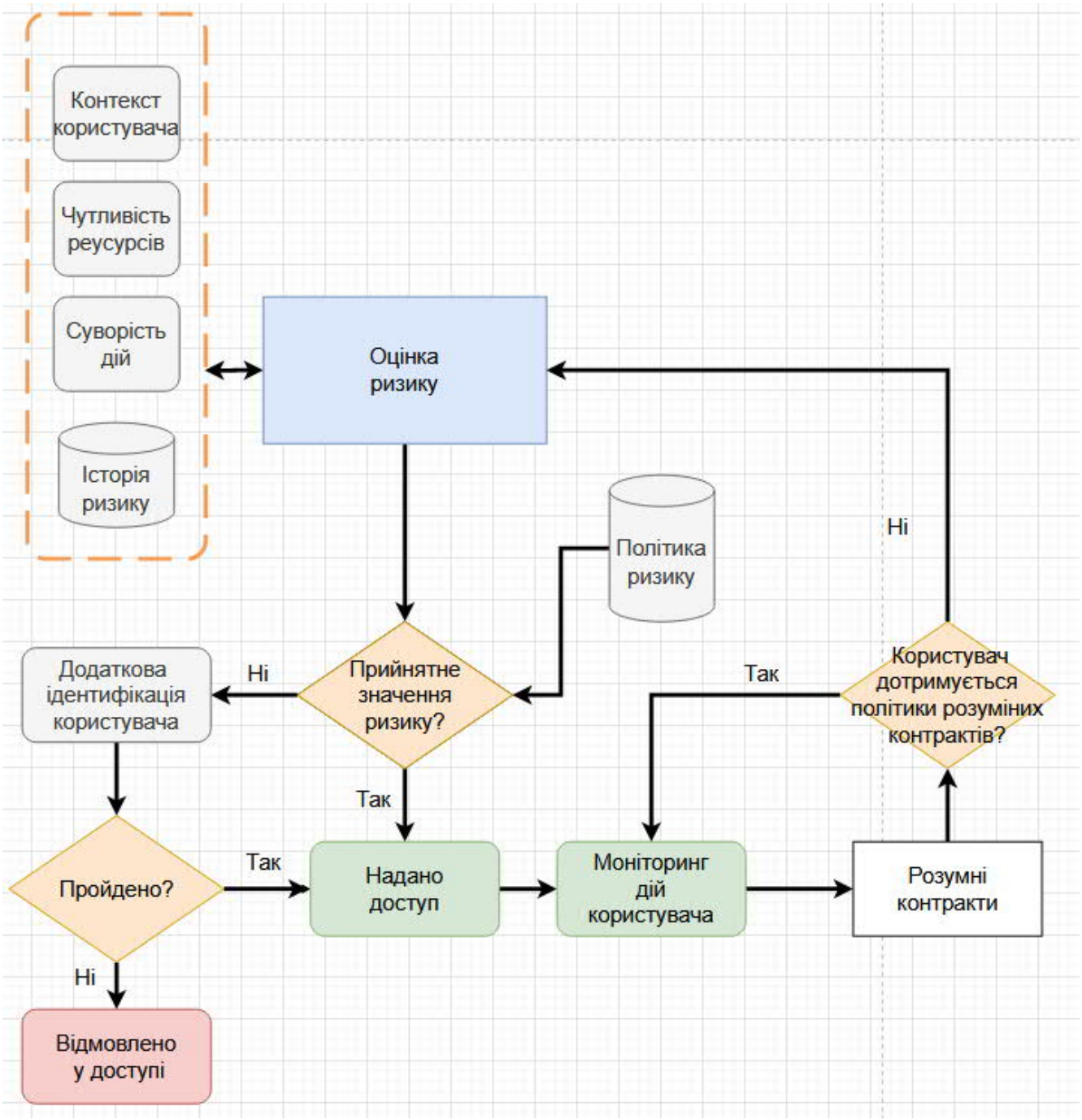

Рисунок 2.2 – Процес отримання доступу до системи

На цьому етапі у нас є два сценарії. Перший сценарій - надання доступу. Якщо доступ надається, смарт-контракти будуть використовуватися для відстеження та моніторингу діяльності користувачів під час сеансу доступу для виявлення шкідливих дій та переконання, що користувач дотримується умов

договору. Якщо смарт-контракт не виявить зловмисних дій, він буде відслідковувати та контролювати поведінку користувачів протягом усього сеансу доступу. Хоча якщо буде виявлено порушення, система видасть попередження та припинить сеанс.

Другий сценарій - це заборона доступу. Якщо в доступі заборонено, щоб знизити неправдиво-позитивну швидкість системи, користувачеві буде запропоновано надати додаткове підтвердження ідентифікації. Якщо система отримає правильні облікові дані, доступ буде наданий, а сеанс буде відстежено, інакше в доступі буде відмовлено.

Класичні підходи до контролю доступу не забезпечують спосіб виявлення шкідливих дій та захисту системних ресурсів після надання доступу. Тому запропонована модель покращує гнучкість системи та додає можливості виявлення аномалій, використовуючи смарт-контракти для відстеження та моніторингу діяльності користувача під час сеансів доступу. Модуль оцінки ризику адаптує дозвіл користувача адаптивно залежно від його поведінки під час сеансів доступу, таким чином, якщо виявлено ненормальну дію, привілеї користувача будуть зменшені або сеанс доступу буде припинено.

Розумні контракти настільки потужні через свою гнучкість. Вони можуть безпечно зашифровувати та зберігати дані, обмежувати доступ до даних лише бажаним сторонам, а потім запрограмувати їх на використання даних у межах самостійного виконання логічного робочого процесу операцій між сторонами. Розумні контракти перетворюють бізнес-процес на обчислювальний процес для підвищення операційної ефективності. Реалізація розумного контракту здійснюється за допомогою побудови програмного коду, який працює на блокчейні. У запропонованій моделі для кожного наданого користувача буде створений розумний контракт. Тому модуль моніторингу буде порівнювати поведінку користувача під час сеансу доступу з умовами договору для виявлення ненормальних дій протягом сеансів доступу.

Користувач, який запитує, визначає дані, до яких потрібно отримати доступ, і дії, які слід виконати в запиті доступу. Отже, якщо доступ буде наданий, буде створений розумний контракт, реалізуючи умови, які гарантують, що користувач

матиме можливість лише доступу до даних та дій, зазначених у запиті на доступ. Ресурси / дані, до яких користувач звертається, відстежуються, щоб перевірити, чи користувач отримує доступ до ресурсів, дозволених умовами смарт-контракту. Аналогічно, дії, що виконуються користувачем під час сеансу доступу, відстежуються для виявлення будь-яких порушень умов смарт-контракту. Якщо виявлено порушення, система видасть попереджувальне повідомлення і сеанс доступу буде припинено. Процес застосування інтелектуальних контрактів для моніторингу діяльності користувачів під час сеансів доступу показаний на рис. 3.

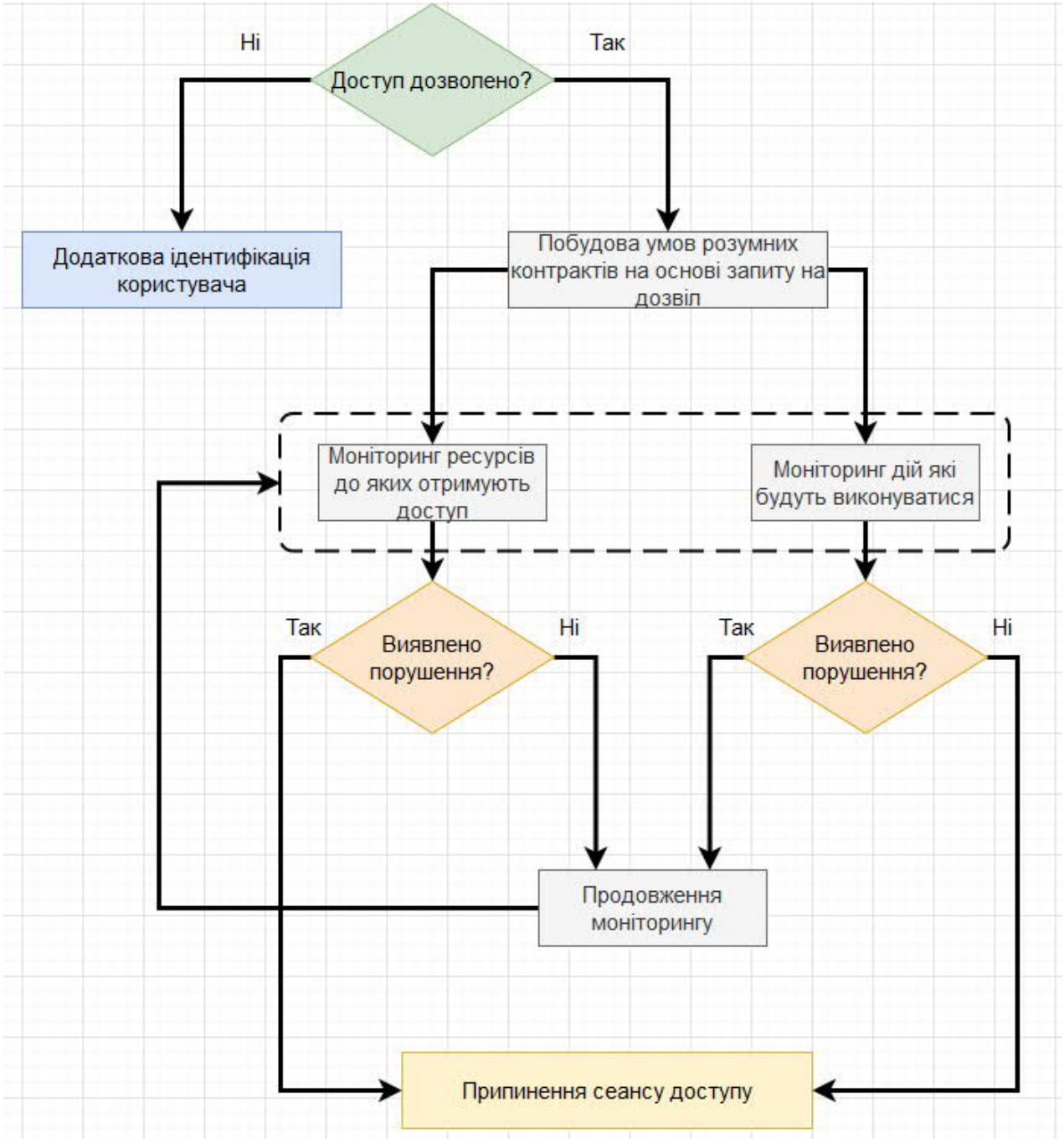

Рисунок 2.3 – Робота смарт-контрактів

Дана модель забезпечує необхідну гнучкість для системи доступу до баз даних. Це забезпечує ефективне рішення для багатьох несподіваних обставин, які потребують порушення політики, включаючи захист в режимі реального часу та контекстуальні функції для прийняття рішення про доступ. Також використання розумних контрактів для моніторингу діяльності користувачів під час сеансу доступу забезпечує важливе рішення для своєчасного виявлення порушень безпеки для захисту системних ресурсів та запобігання розкриттю конфіденційної інформації.

Модуль оцінки ризику є найважливішим елементом в моделях на основі ризику. Він несе відповідальність за оцінку вартості ризику, пов'язаного з чинниками системного ризику для визначення доступу. Однак важко виміряти ризики для безпеки, не маючи набору даних, щоб описати ймовірність різних інцидентів та їх вплив. Крім того, важливо враховувати гнучкість системи при виборі методики оцінки ризику.

2.2 Розробка алгоритму контролю доступу до бази знань підприємства з використанням нечіткої логіки

Для початку, мною було виділено кроки які необхідно виконати для успішної розробки алгоритму та побудови схеми роботи системи:

1. Словесний опис алгоритму роботи системи на основі отриманих даних після проведення досліджень. На даному кроці проводиться нотування думок про те, як має працювати система та які компоненти вона містить.
2. Визначення вхідних та вихідних змінних нечіткої системи.
3. Розробка схеми за словесним описом.
4. Опис роботи компонентів системи.
5. Порівняння із вже існуючими системами які було проаналізовано.

Запропонований алгоритм контролю доступу до бази знань підприємства використовує нечітку логіку для прийняття рішень про дозвіл. Блок-схему алгоритму представлено на рисунку 2.4.

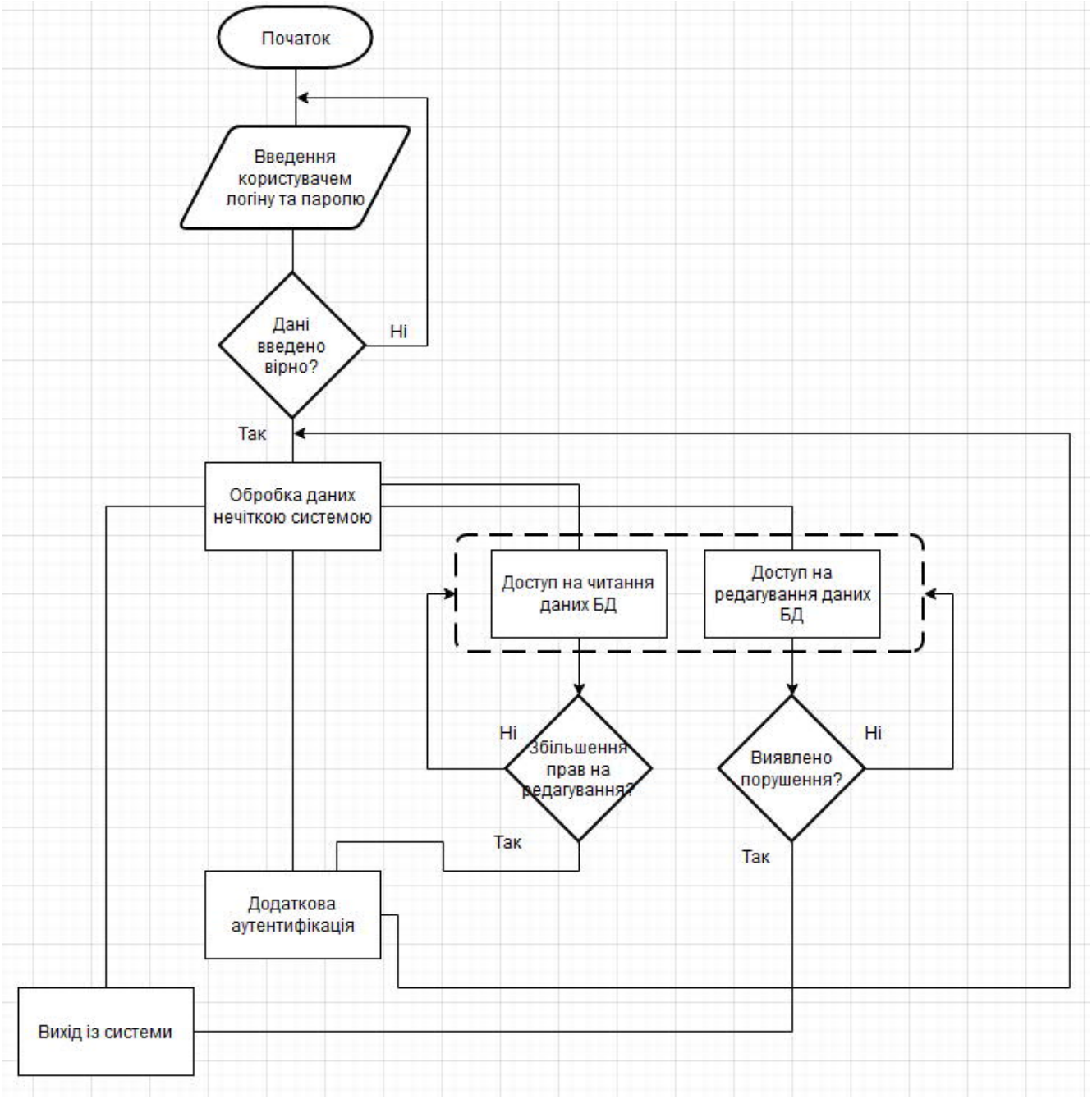

Рисунок 2.4 – Блок-схема розробленого алгоритму контролю доступу

Першим етапом є введення логіну та паролю користувачем який намагається отримати доступ до бази даних. Якщо дані введено не вірно, користувачу показується повідомлення з помилкою та пропонується ще раз ввести дані. У випадку коректної ідентифікації користувача, комп’ютерна система проводить

збирання даних про нього та подає їх на вхід до нечіткої системи. Вона в свою чергу проводить калькуляції та видає один із чотирьох результатів:

1. Надання доступу на читання даних.
2. Надання доступу на редагування даних.
3. Проведення додаткової аутентифікації користувача.
4. Відмова у наданні доступу.

Надання доступу на редагування або читання дозволяє користувачу користуватися системою управління базою даних з використанням відповідних прав. У випадку проведення додаткової аутентифікації, користувач повинен надати дані для її перевірки. Тоді, результат даної перевірки надходить до нечіткої системи яка знову приймає рішення про подальший дозвіл на доступ. Також, користувач може збільшити свої права доступу з читання на редагування пройшовши вже відомий шлях аутентифікації. Під час роботи у системі управління до БД, якщо користувач спробує вийти за межі наданих йому прав, наприклад спробує редагувати дані коли у нього є доступ лише на читання, то така спроба автоматично припиняє взаємодію користувача із СУБД та подає відповідний сигнал комп'ютерній системі, який потім слідує до нечіткої системи і вона приймає рішення про подальший розвиток подій.

Як входи нечіткої системи я вирішив виділити наступні змінні:

1. Час доступу, який ідентифікує час на момент звернення.

2. Історія ризику. Ідея взята із проаналізованої раніше системи на основі оцінки ризику. Даний вхід являє собою нормалізовану сукупність багатьох факторів, які можуть впливати на збереження цілісності даних, такі як визначення місцезнаходження користувача, його попередні дії у системі, строк експлуатації системи, тощо. Історія ризику зображує ідею сприйняття користувача системою, зокрема відповідь на питання чи може система довіряти користувачу. На мою думку, даний підхід має значний внесок у оцінці ефективності роботи системи.

3. Рівень доступу до даних. Визначає спектр дій, які може проводити користувач із даними. У даному випадку взято лише два рівні доступу - читання та редагування.

На основі блочного алгоритму та словесного опису було розроблено схему усієї системи. Повний вигляд схеми наведений у додатку Б. У тексті буде наведено зображення окремих модулів системи та проведено опис їх функціоналу.

Рисунок 2.5 – Модуль ідентифікації

На рисунку 2.5 ви можете побачити частину системи - модуль ідентифікації. Він є вхідною точкою до всієї системи. Даний модуль забезпечує ідентифікацію користувача за допомогою логіну та паролю. Це стандартний метод контролю у системах на основі довіри. Сам по собі даний метод контролю не є дуже ефективним, тому найчастіше у таку систему додають нечітку логіку, яка дозволяє брати до уваги динамічну інформацію що значно збільшує ефективність системи.

Комп’ютерна система є другим по важливості компонентом всієї структури, після нечіткої системи. Саме вона здійснює зберігання та модифікацію даних про користувача, яка в подальшому йде на входи до нечіткої системи. Вона оперує даними часу звернення до системи, визначення місцезнаходження, історії ризику даного користувача та його рівнем доступу. Також вона обробляє додаткові перевірки, запити на збільшення прав доступу та спроби користувача вийти за межі прав доступу. Ці події змінюють дані про користувача, та відповідно впливають на результат роботи нечіткої системи.

Нечітка система являється мозком всієї структури. Вона бере до уваги дані про користувача та приймає рішення про надання доступу. В основі роботи даної системи лежить алгоритм Мамдані - алгоритм нечіткого логічного виводу по базі знань (базі правил). У наступному підрозділі буде наведено значення та проведено складання бази правил для даної системи контролю доступу.

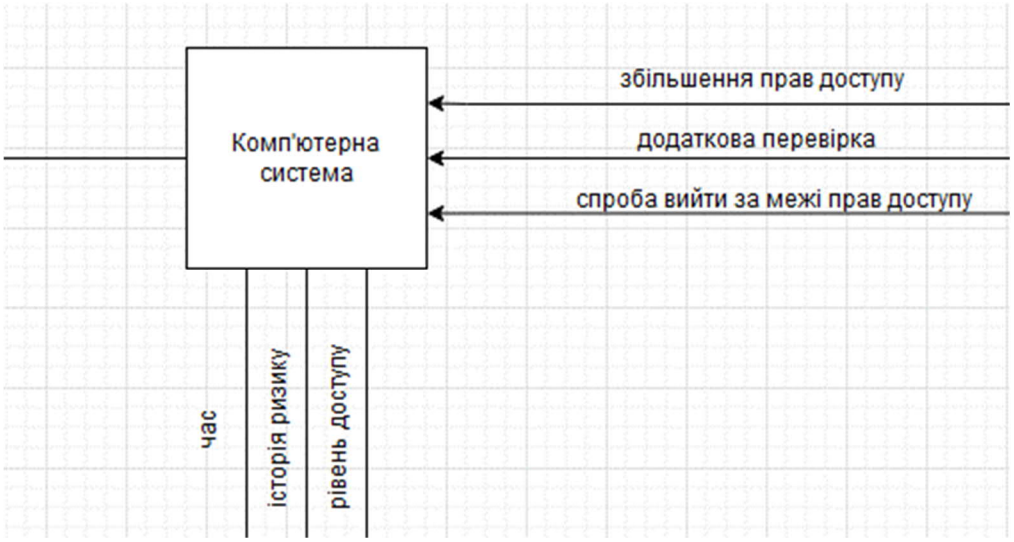

Рисунок 2.6 – Комп’ютерна система

Далі розглянемо нечітку систему. На рисунку 2.7 ви можете бачити вигляд блоку нечіткої системи у розробленій схемі.

Рисунок 2.7 – Нечітка система

Назви входів та їхні стани, які наведені у дужках, можна описати наступним чином:

–час звернення (робочий або неробочий);

–історія ризику (низька, середня, висока);

–рівень доступу (читання або редагування(і читання) даних);

Рисунок 2.8 – Модуль обробки доступу

Модуль обробки доступу, який зображений на рисунку 2.8 відповідає за обробку результату виводу нечіткої системи та запуску відповідної дії.

На рисунку 2.9 зображено два останніх блоки розробленої системи. Модуль додаткової аутентифікації відповідає за проведення перевірок, які виникають при наявності стану “аутентифікація” на виході нечіткої системи та при запиті на збільшення прав доступу.

Пристрій управління БД приймає на вхід один з доступів та розпочинає сесію взаємодії користувача із базою даних. Якщо під час сесії з доступом на читання даних користувач спробує їх редагувати, то сеанс перерветься та комп’ютерна система буде сповіщена про спробу виходу за межі прав доступу, що автоматично підвищить рівень історії ризику до високого.

У результаті було розроблено алгоритм роботи та схему реалізації системи контролю доступу до бази даних підприємства. Дану систему можна легко модифіковувати та збільшувати її можливості щодо контролю доступу. Для прикладу, до неї можна додати визначення синсетивності даних, тобто їх вагу. У такому випадку кожному користувачу від початку буде присвоєний певний набір даних до яких він буде мати доступ(в контексті бази даних - такими даними виступають таблиці або документи). В залежності від інших факторів(наприклад історії ризику, місцезнаходження користувача, часу доступу, його посади) є можливість надання доступу до недоступних у даний момент даних.

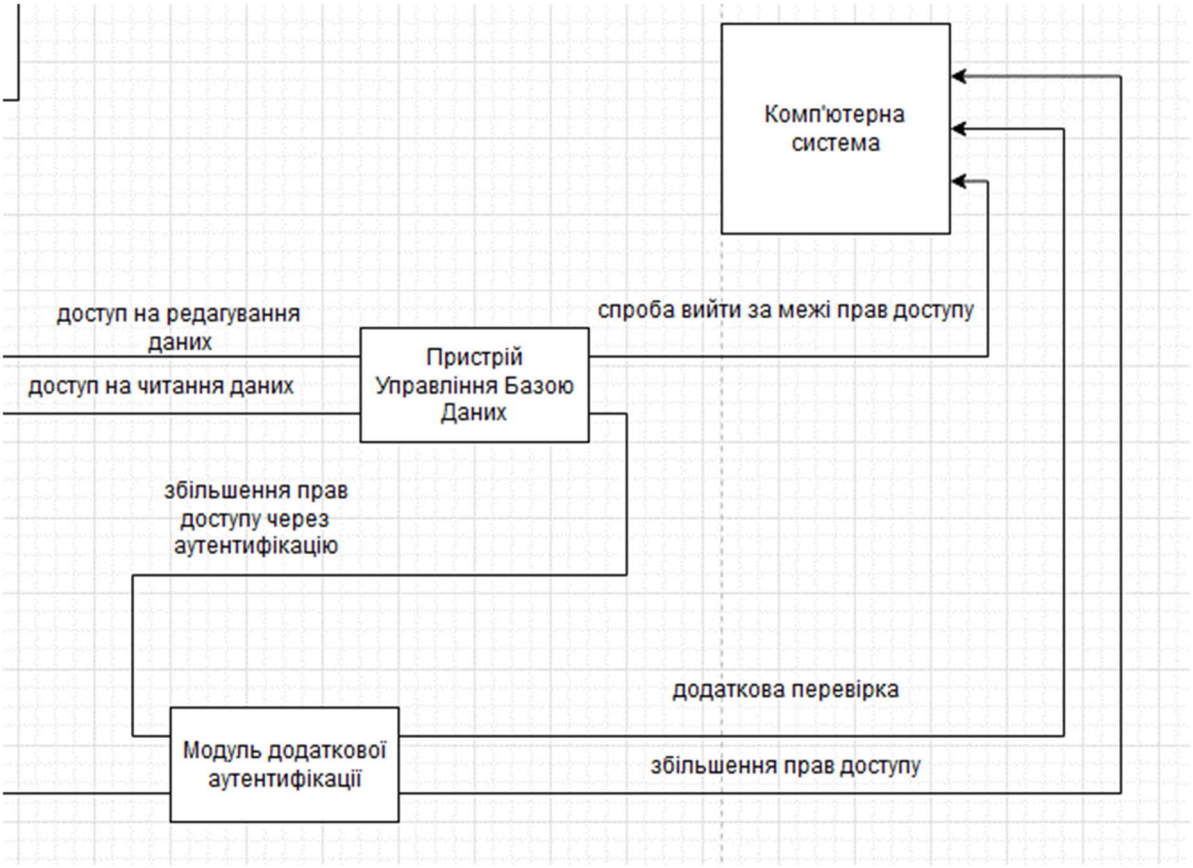

Рисунок 2.9 – Блоки аутентифікації та пристрою управління БД

2.3 Нечітка база знань та розробка продукційних правил

Якщо звичайна або класична база даних - це структурований набір інформації (записів або даних), що зберігається на комп'ютері, нечітка база даних здатна обробляти невизначену або неповну інформацію за допомогою нечіткої логіки. Існує багато форм додавання гнучкості у нечіткі бази даних [29]. Найпростіша методика - додати ступінь нечіткої приналежності до кожного запису, тобто атрибута в діапазоні [0,1]. Однак існують інші види баз даних, які дозволяють зберігати нечіткі значення в нечітких атрибутах, використовуючи нечіткі набори, розподіл можливостей або нечіткі ступені, пов'язані з деякими атрибутами та з різними значеннями (ступінь членства, ступінь важливості, ступінь виконання тощо). Звичайно, нечіткі бази даних повинні дозволяти нечіткі запити з використанням нечітких даних, і є деякі мови, які дозволяють здійснювати такий тип запитів, як FSQL або SQLf. Синтез дослідження нечітких баз даних включає

такі області: гнучкі запити в класичних або нечітких базах даних, розширення класичних моделей даних з метою досягнення нечітких баз даних (нечіткі реляційні бази даних, нечіткі об'єктно-орієнтовані бази даних тощо), нечітке концептуальне моделювання, нечіткі методи видобутку даних та застосування цих досягнень у реальних базах даних. База даних, здатна зберігати та обробляти недосконалу інформацію, яка моделюється з використанням переваг теорії нечітких множин.У звичайній базі даних зберігаються лише чіткі (ідеально описані) дані. Однак через неточність, невизначеність, невизначеність, незавершеність чи неоднозначність, у реальному світі багато даних доступні лише в недосконалій формі. Нечіткі бази даних мають намір зрозуміти недосконалу інформацію про модельовану частину світу та представити її безпосередньо, наскільки це можливо, в базі даних. Два провідних підходи до представлення недосконалої інформації в базах даних - це можливий підхід та підхід, що базується на подібності [30].

Чотири основних ознаки недосконалої інформації в системах баз даних:

–Невідповідність - це різновид смислового конфлікту, що означає той самий аспект реального світу непримірно представлений не раз у базі даних або в декількох різних базах даних. Наприклад, вік Джорджа зберігається як 34 та 37 одночасно. Інформаційна невідповідність зазвичай відбувається за рахунок інтеграції інформації.

–Інтуїтивно зрозуміло, що неточність має відношення до змісту атрибута значення, а це означає, що вибір повинен бути зроблений із заданого діапазону (інтервалу чи набору) значень, але ми не знаємо, яке саме вибрати в даний час. Загалом, розпливчаста інформація представлена лінгвістичними значеннями. Наприклад, вік Романа – це набір {18, 19, 20, 21}, фрагмент неточної інформації, а вік Назарія це лінгвістичний опис віку – “старий”, фрагмент нечіткої інформації.

–Невизначеність пов'язана зі ступенем істинності її значення атрибуту, і це означає, що ми можемо розподілити деякі, але не всі, наші переконання щодо певної цінності або групи цінностей. Наприклад, вірогідність того, що зараз Кріс становить 35 років, може становити 98%. Випадкова невизначеність, описана теорією ймовірностей, тут не розглядається.

–Неоднозначність означає, що деяким елементам моделі не вистачає повної семантики, що призводить до декількох можливих інтерпретацій. Як правило, стосовно цього може існувати декілька різних видів недосконалості та сама інформація. Наприклад, вік Романа - це набір {18, 19, 20, 21} і їхні можливості складають відповідно 70%, 95%, 98% та 85%. Неточність, невпевненість, та невизначеність - це три основні типи недосконалої інформації.

У наш час моделювання багатофакторних залежностей у технологіях та інших сферах життя все більше і більше здійснюється за допомогою нечітких баз даних знань. Нечітка база знань - це набір нечітких правил “якщо-тоді”, що визначають взаємозв'язок між входами та виходами обстежуваного об'єкта. [31]

Нечіткі правила бази знань типу Мамдані складаються з антецедентів таконсеквентів викладених у зрозумілих людині термінах. При проектуванні нечітких систем перевага віддається компактним базам знань, які містять менше правил. У MATLAB потужна панель інструментів Fuzzy Logic Toolbox має ряд функцій для створення баз знань Сугено та Мамдані.

Використовуючи графічний інтерфейс, ви можете автоматизовано створювати як базу даних Sugeno, так і базу даних Mamdani. Однак у майбутньому функції ідентифікації залежностей, як правило, будуть зосереджені на базі знань типу Сугено, що заохочує розробку альтернативних функцій для баз даних Мамдані.

Програмний комплекс параметричної та структурної ідентифікації багатофакторних залежностей при використанні бази знань типу Мамдані розширює можливості авторської програми за наступними функціями:

1. fismat = genmam(inp, outp, numMfs) - створює базу знань типу Мамдані на основі навчальної вибірки за допомогою простого рівномірного розподілу вхідних та вихідних даних залежно від кількості зазначених термів. Крім того, генеруються правила, в яких наслідок визначається як термін, якому значення функції з попередніх ядер належить найбільше. inp, outp Вихідні матриці даних кожен numMfs - кількість термінів для входів і виходів у послідовності (перший Значення відповідає кількості доданків першого введення, останнього - останнього виводу).

На рисунку 2.10 зображено базу знань типу Мамдані за допомогою функції genmam. На рисунку 2.11 представлена функція оригінал.

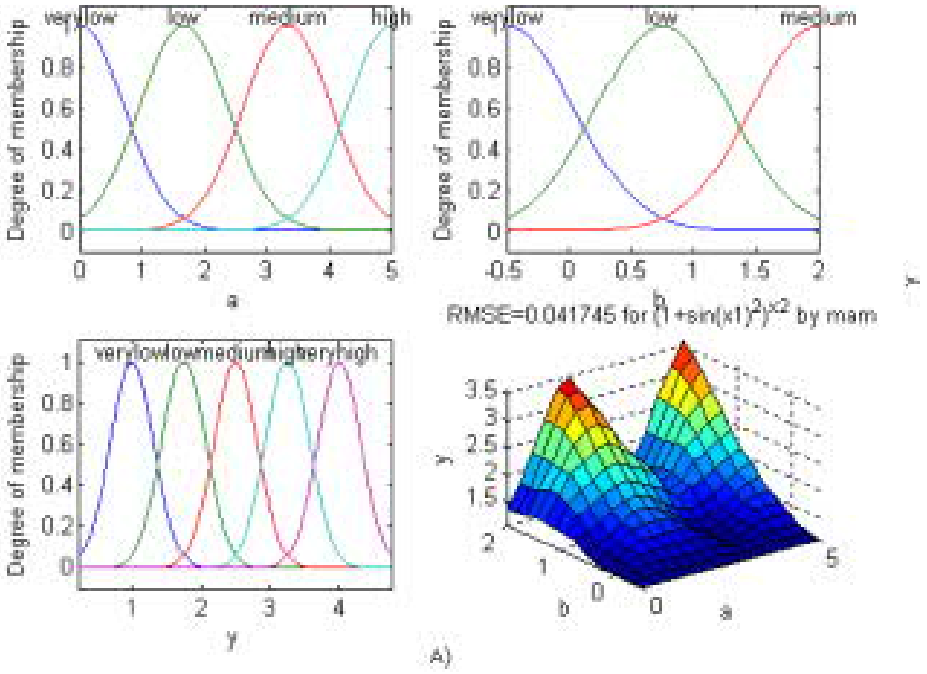

Рисунок 2.10 – База знань типу Мамдані за допомогою функції genman

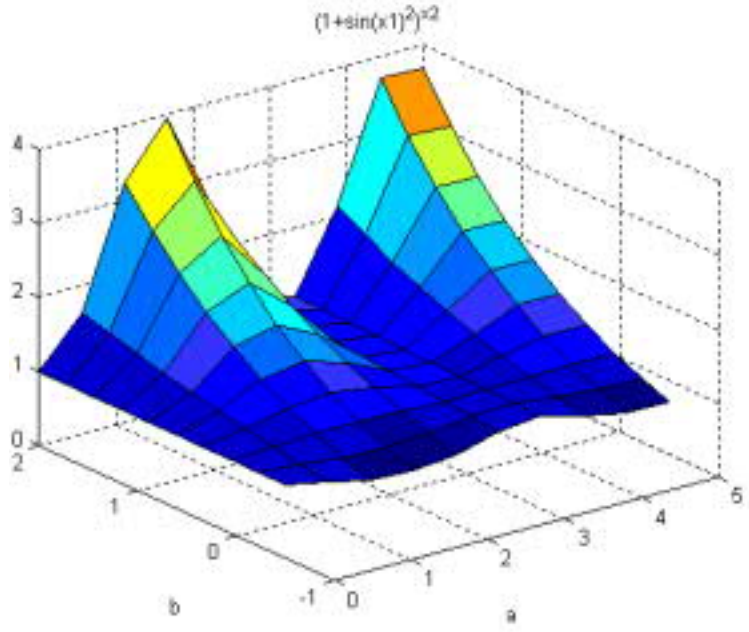

Рисунок 2.11 – Функція оригінал

2. minRMSE, mnmdmx – сценарії, які здійснюють структурну ідентифікацію бази знань методом повного пошуку правил, знаходять найменші бази знань, що мають обсяг правил від одного правила до повної бази знань.

3. outfis = optfis(fis, inp, out, options) – Функція, яка використовується набором інструментів оптимізації, щоб мінімізувати розбіжності між вихідними та вихідними даними бази знань fis із вхідних даних inp, параметри - налаштування для функції fmincon, яка є основою цієї функції.

4. outfis = setfisnparam(fis, params, enumparams) – Допоміжна функція, яка призначає витягнуті параметри параметрам зі структури FIS FIS під час навчання optfis (). Enumparams вказує позицію параметра у структурі fis.

5. RMSE = dest_fun(params, fis, inp, out, inptt, outtt, enumparams) – допоміжна функція, яка використовується як цільова функція у fmincon, виконує логічний вихід на базі знань щодо нових параметрів .params.

На рисунку 2.12 показано значення RMSE та дві бази знань, знайдені з розробленим комплексом.

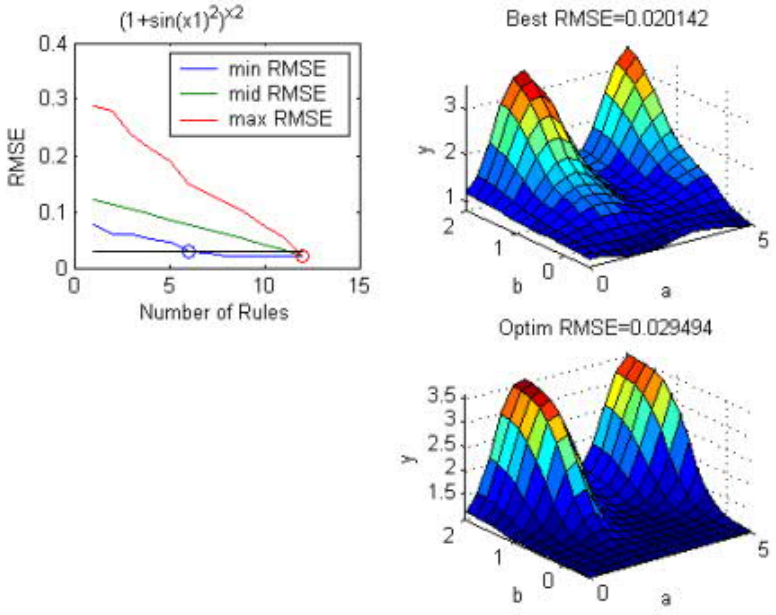

Рисунок 2.12 - Значення RMSE та дві бази знань знайдених за допомогою розробленого комплексу

Комплекс програм структурної та параметричної ідентифікації за допомогою бази знань типу Мамдані лише повністю підтримує функцію членства Гауса (рисунок 2.10). У майбутньому планується підтримка всіх функцій членства та узагальнення сценаріїв функції. Функція optfis () містить параметри перерахування, за допомогою яких користувач може визначити параметри навчання. Крім того,

розробляється модуль clearsafe, який забезпечує прозорість нечіткої бази знань під час навчання.

База знань Мамдані складається з правил, в яких антецеденти та консеквенти задаються нечіткими множинами [32]. Слід зазначити, що розробники нечітких систем намагаються створити адекватну нечітку базу знань з найменшою складністю - з невеликою кількістю правил. Таку компактну модель легше перевірити. Вона надає найвищу швидкість логічного виводу і вимагає найменшої кількості засобів реалізації апаратного забезпечення.

Структурна ідентифікація здійснюється на першому етапі вивчення бази знань. Це формування нечіткої бази знань, яка приблизно відображає нелінійні відносини вхду-вихіду через лінгвістичні правила. Ці правила створюються експертом або отримуються шляхом вилучення нечітких знань з експериментальних даних. [33]

Насправді структурна ідентифікація моделей полягає у пошуку бази знань, яка має стабільну базу лінгвістичних правил з найменшою кількістю помилок ідентифікації.

Для процесів корекції накопичених раніше знань найбільш інформативними кривими навчання є залежність невідповідності від часу навчання, а для процесів отримання нових знань - залежність невідповідності від кількості правил у базі знань.

Однією з переваг систем нечіткого висновку є те, що вони працюють задовільно, коли кількість правил у базі знань менше, ніж Nmax. Звичайно, додавання кожного нового правила покращує якість системи, але в реальних умовах обсяг надійних знань завжди обмежений. Крім того, існує так зване «явище насичення» для нечітких баз знань, суть якого полягає в тому, що додавання нового правила щодо певного обсягу баз знань не покращує продуктивність системи. Це явище на діаграмі залежності рівноваги від обсягу бази знань відповідає "плато насичення" - майже горизонтальному відрізку лінії.

На другому етапі досліджувана залежність параметрично ідентифікується шляхом пошуку параметрів у базі нечітких знань, які мінімізують відхилення

результатів нечіткого моделювання від експериментальних даних. Параметри, які можна налаштувати, це ваги правил та параметри нечітких функцій членства [34].

Вибір правил нечіткої бази знань може бути зведений до задачі оптимізації рюкзака, яка є NP-повною [35]. Правилу бази знань відповідає будь який предмет, який може потрапити в рюкзак, точність бази знань - корисність рюкзака, а кількість правил - загальна кількість вибраних предметів. Різниця між проблемами полягає у різних типах функцій корисності, які є лінійними в задачі рюкзака та нелінійними в задачі вибору правил бази знань. Відповідно, алгоритм точного вирішення цієї задачі має експоненціальну обчислювальну складність і, отже, прийнятний лише при невеликій кількості кандидатних правил.

Бази знань, що містять близько 75-80% від максимальної кількості правил, мають найбільшу точність, тобто найменшу помилку.

Для того, щоб зменшити обчислювальну складність, всю базу знань Fuzzy Mamdani [36] можна зменшити вдвічі до 10 правил (2 * 5 поділ, максимальна кількість комбінацій правил = 1024) без великої втрати точності.

Експериментально встановлено, що базу знань Мамдані можна отримати з досить невеликою помилкою ідентифікації, якщо база знань заповнена на 30-50% (6-10 правил), після чого збільшення кількості правил не суттєво зменшує розбіжності. Такі компактні бази знань є більш прозорими та легшими у засвоєнні завдяки меншій складності відповідного завдання оптимізації.

Моя база правил, розроблена для нечіткої системи контролю доступу до бази даних підприємства, налічує 35 правил. Кількість правил вираховується за формулою:

$$\left((k + 1) * \dots \mathrm{n}\right) - 1, \tag{2.1}$$

де k - кількість станів для одного входу,

n – кількість входів, тобто пар (k + 1)

До задуманої кількості станів у нечіткій системі завжди додається ще один стан - невизначений. Для моєї системи вирахування кількості правил буде виглядати наступним чином:

$$((2+1)*(3+1)*(2+1))-1 = (3*4*3)-1 = 35 \quad (2.2)$$

У таблиці 2.1 наведено лише декілька розроблених правил із яких можна виділити певні закономірності поведінки системи. Повна таблиця правил для бази знань приведена у додатку В.

Таблиця 2.1 - Правила бази знань

№	Час звернення	Історія ризику	Рівень доступу	Доступ
1	2	3	4	5
1	2	3	4	5
6	N/A	низька	читання	читання
14	робочий	N/A	редагування	відмова
15	робочий	низька	N/A	відмова
19	робочий	низька	редагування	редагування
20	робочий	середня	читання	читання
21	робочий	середня	редагування	аутентифікація
22	робочий	висока	читання	відмова
30	неробочий	низька	читання	читання
31	неробочий	низька	редагування	аутентифікація
32	неробочий	середня	читання	аутентифікація

У таблиці 2.1 значення N/A означає невизначеність. Із правил наведених у даній таблиці можна виділити наступні закономірності:

–Невизначена Історія ризику завжди приводить до відмови (14 правило).

–Невизначений рівень доступу завжди приводить до відмови (15 правило).

–Невизначений час звернення завжди приводить до відмови, окрім випадку коли Історія ризику низька та рівень доступу - читання. (6 правило).

–Висока історія ризику завжди приводить до відмови (22 правило).

2.4 Алгоритм управління системи доступу на основі нечіткого висновку Мамдані

Одним з перших алгоритмів, який знайшов застосування в системах нечіткого виведення є алгоритм Мамдані. У 1975 році він був запропонований англійським математиком Ебрагімом Мамдані як метод для управління паровим двигуном [37]. У моделі Мамдані математично взаємозв'язок між входами $X = (x_1, x_2, ..., x_n)$ та виходом y визначається нечіткою базою правил наступного формату:

$$\begin{array}{r} \{x_1 = a_{1,j1}\} I \{x_2 = a_{2,j1}\} I \ldots I \{x_n = a_{n,j1}\} \; АБО \\ ЯКЩО \;\; \{x_1 = a_{1,j1}\} I \{x_2 = a_{2,j1}\} I \ldots I \{x_n = a_{n,j1}\} \; АБО \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots АБО` \\ ТОДІ \;\; y = d_j, \; j = \overline{1,m} \end{array} \qquad (2.3)$$

де $a_{1,jp}$ – лінгвістичний терм, яким оцінюється змінна x_i в рядку з номером $jp(p = \overline{1,k_j})$;

k_j – кількість рядків-кон'юнкція, в яких вихід y оцінюється лінгвістичним термом d_j;

m – кількість термів, що використовуються для оцінки вихідної лінгвістичної змінної.

За допомогою операцій ∪ (АБО) та ∩ (І) нечітку базу правил можна переписати в більш компактному вигляді:

$$\bigcup_{p=1}^{k_j}\left(\bigcup_{i=1}^{n}\{x_i = a_{i,jp}\}\right) \to y = d_j,\ j = \overline{1,m}, \tag{2.4}$$

Згідно з алгоритмом Мамдані, логічний висновок здійснюється за наступні шість етапів [38]:

1. Формування бази правил систем нечіткого виводу у наступному вигляді:

ПРАВИЛО <1>: ЯКЩО $(x \in A_1\ I\ y \in B_1), TO, z = C_1$,

ПРАВИЛО <2>: ЯКЩО $(x \in A_2\ I\ y \in B_2), TO, z = C_2$,

де x_1, x_2 – вхідні лінгвістичні змінні;

z – вихідна лінгвістична змінна;

$A_1, A_2, B_1, B_2, C_1, C_2$ – функції належності, визначені відповідно на *x,y,z*.

2. Введення нечіткості (фазифікація). Функції належності, що визначенні на вхідних змінних x0,y0 застосовуються до їх фактичних значень $\mu_{A_1}(x0), \mu_{A_2}(x0), \mu_{B_1}(y0), \mu_{B_2}(y0)$ для визначення функції належності для кожного правила.

3. Агрегування передумов в нечітких правилах продукцій. Для знаходження ступеня істинності умов кожного з правил нечітких продукцій використовуються парні нечіткі логічні операції:

$$c_1 = \min\{\mu_{A_1}(x0), \mu_{B_1}(y0),\}, c2 = \min\{\mu_{A_2}(x0), \mu_{B_2}(y0),\} \tag{2.5}$$

Ті правила, ступінь істинності умов яких відмінна від нуля, вважаються активними і використовуються для подальших розрахунків [39].

4. Активізація висновків нечітких правилах продукцій.

$$\mu(z) = \min\{c_i, \mu_X \mu(z)\} \tag{2.6}$$

Здійснюється за формулою 2.6, коли функція належності виведення «відсікається» по висоті відповідно обчисленої функції належності передумови правила (нечітка логіка «І»):

$$C_1^{'}(z) = (c_1 * C_1(z)), C_2^{'}(z) = (c_1 * C_2(z)) \quad (2.7)$$

При цьому для скорочення часу виведення враховуються тільки активні правила нечітких продукцій.

5. Акумуляція висновків нечітких правил продукцій. Всі нечіткі підмножини, отримані для кожної вихідної змінної (у всіх правилах), об'єднуються разом, щоб сформувати одну нечітку підмножину для кожної вихідної змінної. При подібному об'єднанні використовуються операції:

$$\max: \mu_{\sum}(z) = C_1^{'}(z) \cup C_2^{'}(z) \text{ або} \quad (2.8)$$

$$\text{sum: } \mu_{\sum}(z) = (C_1^{'}(z) \cup C_2^{'}(z)) \setminus (C_1^{'}(z) \cap C_2^{'}(z)) \quad (2.9)$$

6. Приведення до чіткості (дефазифікації). Дана процедура використовується, коли необхідно перетворити нечітку вихідну множину в чітке число. Найчастіше для моделі Мамдані використовується дефазифікація центроїдним методом, коли чітке значення вихідної змінної визначається як центр ваги для кривої $\mu_{\sum}(z)$ [40].

Процедура отримання логічного висновку показана на рисунку 2.13.

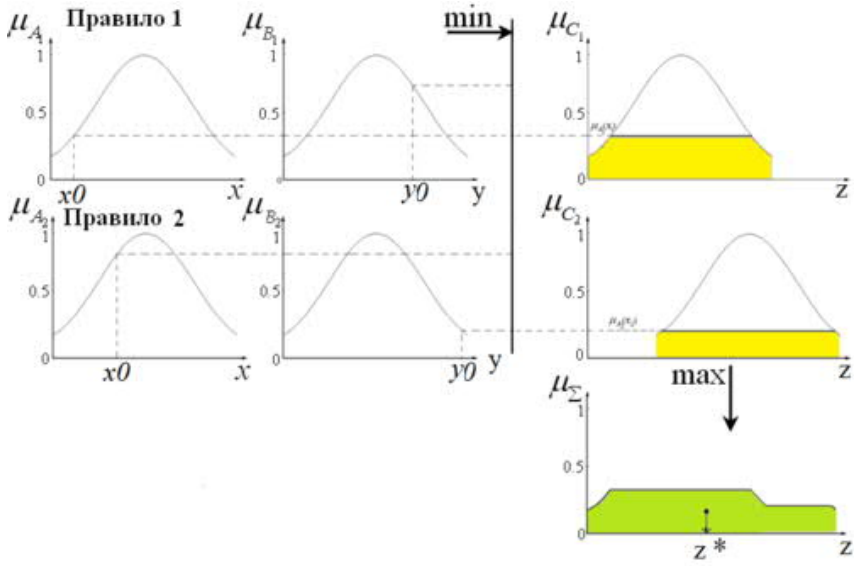

Рисунок 2.13 – Процедура отримання логічного висновку Мамдані

2.5 Висновки до розділу

У другому розділі було виконано наступні задачі:

–розглянуто принципи роботи алгоритму контролю доступу з використанням нечіткої логіки та системи оцінки ризику;

–розроблено блок-схему алгоритму контролю доступу до бази даних підприємства;

–описано принципи роботи розробленого алгоритму;

–здійснено опис входів нечіткої системи та їх станів;

–досліджено базу знань та створено таблицю продукційних правил;

3 Реалізація нечіткої системи в середовищі matlab

3.1 Розробка моделі запропонованого засобу

Для реалізації розробленого алгоритму я обрав середовище MATLAB, додаток до нього під назвою Fuzzy Logic Toolbox для роботи з нечіткою логікою та пакет Simulink для реалізації контролерної системи.

MATLAB розшифровується як Matrix Laboratory та розробляється компанією Mathworks. Це запатентована мова програмування та числове обчислювальне середовище, які являються одними з найкращих у світі, і використовуються для технічних та числових обчислень. MATLAB дозволяє будувати графіки даних [41], реалізовувати алгоритми, особливо складні маніпуляції з матрицями і навіть створювати користувальницькі інтерфейси. Спочатку він був розроблений для чисельних обчислень, але сьогодні існує безліч інструментів та пакетів, що дозволяють розширити його використання в технічних обчисленнях та вбудованих системах. Ці набори інструментів використовують символічний механізм MuPAD, що дозволяє отримати доступ до символьних обчислювальних здібностей. Додатковий пакет, Simulink, додає графічне багатодоменне моделювання та модельний дизайн для динамічних та вбудованих систем. По суті, це забезпечує легке середовище для інтеграції програмування, візуалізації та обчислень. Тому вирішення задач, які виражаються в математичних позначеннях, є простим. У ньому понад 3 мільйони користувачів, і ці користувачі походять з науки, економіки та, очевидно, машинобудування. Станом на 2020 рік MATLAB налічує понад 4 мільйони користувачів у всьому світі, які походять з різних верств техніки, науки та економіки. Програмісти MATLAB є одними з найбільш високооплачуваних у світі [42].

MATLAB широко використовується у різних наукових сферах. Ось декілька прикладів:

– Інформатика. Вона в основному базується на математичних завданнях. MATLAB має багатий набір інструментів для всіх математичних завдань. Вчені можуть швидко створити прототип алгоритму. Після цього вони можуть оцінити

ресурси розробки та доручити їх реалізації на різних мовах програмування, таких як C, C ++ та Java. Тому причиною використання MATLAB в інформатиці є підвищення продуктивності. Окрім цього, комп'ютерні вчені використовують MATLAB для перевірки результатів роботи та роботи з невідомими в алгоритмах.

– Обробка зображень. Робота з цифровими зображеннями являє собою управління пікселями за допомогою математичних методів та маніпулювання матрицею. MATLAB популярний для матричних маніпуляцій, і матриці є невід'ємною частиною обробки зображень. Починаючи від додавання матриць до їх сортування та тестування матриць, на MATLAB все можливо реалізувати з легкістю [43].

– Робота з математичними параметрами. MATLAB ідеально підходить для всіх робіт, що включають лінійну алгебру. Починаючи від прогнозування обороту компанії до її майбутніх очікувань та прогнозів, Дане середовище є важливим для економічних робіт. Це пов'язано з тим, що аналітична та дослідницька робота з ним може допомогти у прийнятті вигідних бізнес-рішень та складанні фінансових планів для отримання найкращої віддачі та переваг у короткостроковій перспективі. Переходячи до чисельного аналізу, MATLAB створений спеціально для цього на самому початку. Починаючи від будівництва будівель і закінчуючи астрономією, вхідні та вихідні дані MATLAB є критично важливими для всіх комерційних проектів. Він широко використовується архітекторами. Крім того, його використовують для дослідження різних нових теорій, особливо у галузі охорони здоров'я та фармацевтичної промисловості.

Три основних переваги середовища MATLAB над програмами конкурентами:

1. MATLAB - це мова програмування високого рівня, незалежно від того, що її крива навчання крута і трохи важка для засвоєння. Однак, якщо ви володієте певними знаннями про будь-яку загальноприйняту мову програмування високого рівня, таку як Java або Python, навчання може бути простим та зручним. Вона має всі загальні компоненти, такі як структури даних, оператори керування потоками, об'єктно-орієнтовані функції, функції тощо. Більше того, це масштабована мова

програмування, що дозволяє створювати як невеликі, так і великі комерційні програми.

2. Наявність робочого середовища. Робоче середовище важливо для будь-якої мови програмування. MATLAB має набір інструментів, які збільшують його універсальність. Крім того, він забезпечує надійну платформу для управління змінними та імпорту та експорту даних. Існують інструменти для розробки, налагодження та профілювання. Окрім цього, доступні різні засоби візуалізації для обробки зображень, анімації та подання графічних даних. Ви можете налаштувати графіку, а також створити власний графічний інтерфейс користувача відповідно до ваших вимог.

3. Бібліотека функцій мови MATLAB обширна для всіх математичних робіт та обчислювальних алгоритмів. Доступні всі основні та розширені функції, тому вона така популярна. Крім того, для зовнішньої взаємодії доступні різні API. Якщо ви хочете вивчити мову програмування, яка є досить рідкісною, ідеально підійде MATLAB. Більше того, якщо ви хочете перейти до галузі досліджень, вивчення цієї мови є обов'язковою умовою, оскільки вона є невід'ємною частиною всіх дослідницьких робіт.

Fuzzy Logic Toolbox - це сукупність функцій, побудованих на цифровому обчислювальному середовищі MATLAB®[44]. Він надає інструменти для створення та редагування нечітких систем виведення в рамках MATLAB, або якщо ви хочете, ви можете інтегрувати свої нечіткі системи в симуляції з Simulink®, або ви навіть можете створити окремі програми C, які будуть викликати нечіткі системи які ви побудували за допомогою MATLAB. Цей набір інструментів значною мірою покладається на графічний інтерфейс користувача (GUI), щоб допомогти вам виконати свою роботу, хоча ви можете працювати повністю з командного рядка, якщо хочете. Панель інструментів пропонує три категорії інструментів:

– функції командного рядка;

– графічні, інтерактивні інструменти;

– блоки та приклади Simulink;

Перша категорія інструментів складається з функцій, які ви можете викликати з командного рядка або з власних програм. Багато з цих функцій - це M-файли MATLAB, серія операторів MATLAB, що реалізують спеціалізовані алгоритми нечіткої логіки. Ви можете переглянути код MATLAB для цих функцій, використовуючи оператор “type” з таким синтаксисом “type <function_name>”. Ви можете змінити спосіб роботи будь-якої функції набору інструментів, скопіювавши та перейменувавши M-файл, а потім змінивши свою копію. Ви також можете розширити набір інструментів, додавши власні M-файли. Разом інструменти, засновані на графічному інтерфейсі, забезпечують середовище для проектування, аналізу та реалізації системи нечітких висновків. Третя категорія інструментів - це набір блоків для використання з програмним забезпеченням для моделювання Simulink. Вони спеціально розроблені для високошвидкісного нечіткого логічного висновку в середовищі Simulink.

Для моделювання нечітких систем у середовищі Matlab використовується Fuzzy Logic Toolbox версії 2.7, який містить у собі 5 компонентів, зображення яких представлено на рисунку 3.1. Для побудови, редагування та перегляду нечітких систем виведення використовують наступні інструменти [45]:

1. Fuzzy Logic Designer - для вирішення проблем високого рівня. З його допомогою створюються вхідні та вихідні змінні та присвоюється назва.
2. Membership Function Editor - для визначення форм усіх функцій членства, пов’язаних з кожною змінною.
3. Rule Editor - для редагування списку правил, які визначають поведінку системи.
4. Rule Viewer - для перегляду діаграми нечітких виводів. Він використовується як засіб діагностики, щоб побачити, наприклад, які правила активні або як окремі функції функції членства впливають на результати.
5. Surface Viewer - для перегляду залежностей виходів від входів. Даний інструмент будує карту вихідної поверхні для системи.

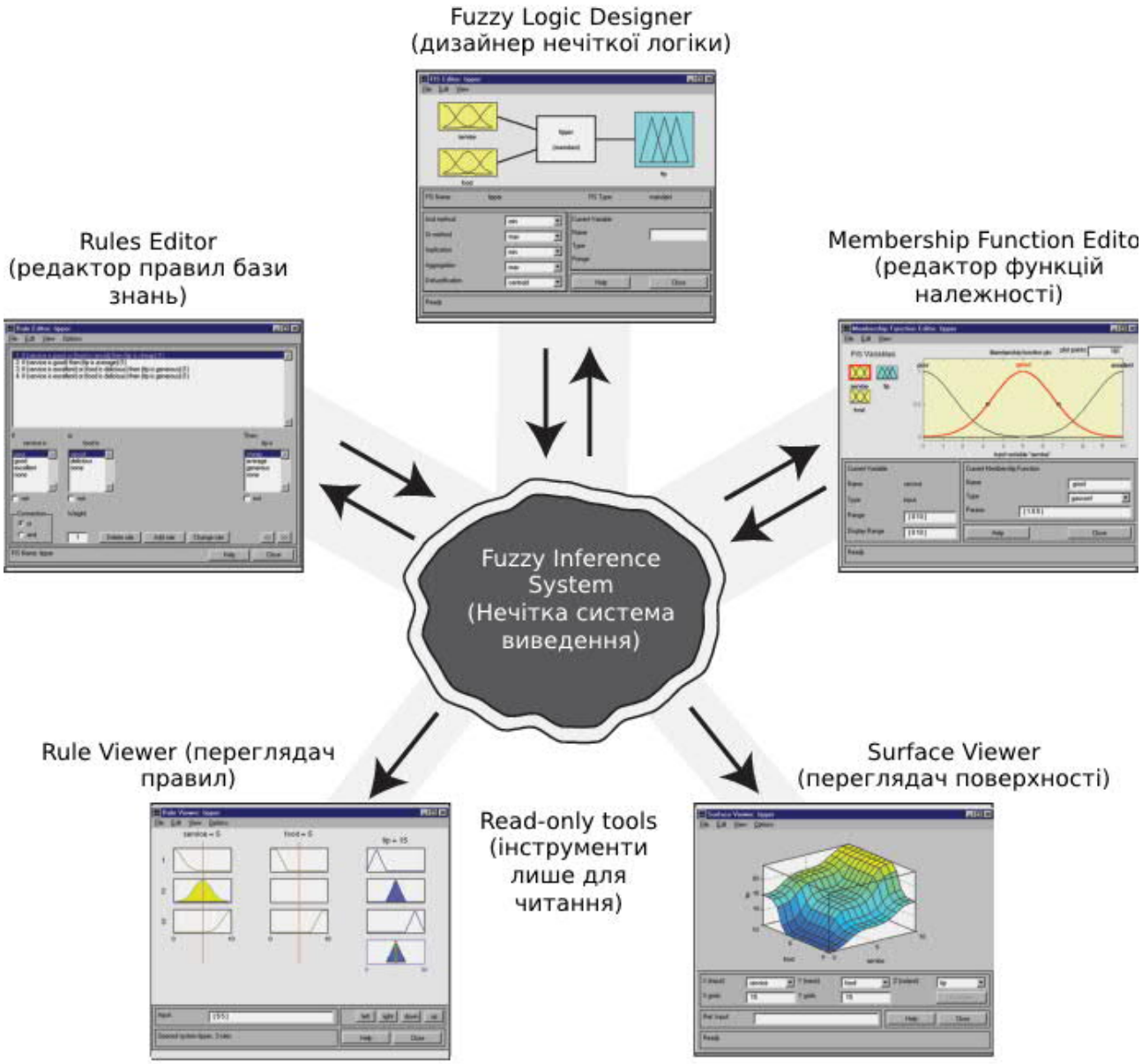

Рисунок 3.1 - Компоненти Fuzzy Logic Toolbox

Ці інтерфейси динамічно пов’язані, оскільки зміни, внесені до одного з них, впливають на те, що ви бачите в будь-якому іншому відкритому інтерфейсі. Наприклад, якщо ви змінюєте імена функцій членства в редакторі “Membership Function Editor”, зміни відображаються у правилах, показаних у редакторі правил “Rule Editor”. Ви можете використовувати інтерфейси користувача для читання та запису змінних як у робочій області MATLAB, так і у файл.

Для створення моделі описаного алгоритму в середовищі Matlab необхідно виконати 3 етапи:

1. Спроектувати структуру входів та виходів системи.

2. Побудувати функції належності для входів та виходів системи.
3. Написати базу правил за якими буде працювати система.

Першим етапом для початку моделювання системи є проектування структури входів та виходів системи. Для цього запустимо середовище Matlab та за допомогою команди “fuzzy” відкриємо графічний інтерфейс Fuzzy Logic Toolbox. Після запуску засобу ми бачимо вікно Fuzzy Logic Designer з яким ми будемо працювати. Вигляд вікна зображено на рисунку 3.1. Спочатку необхідно задати кількість входів та виходів і їхні назви [46].

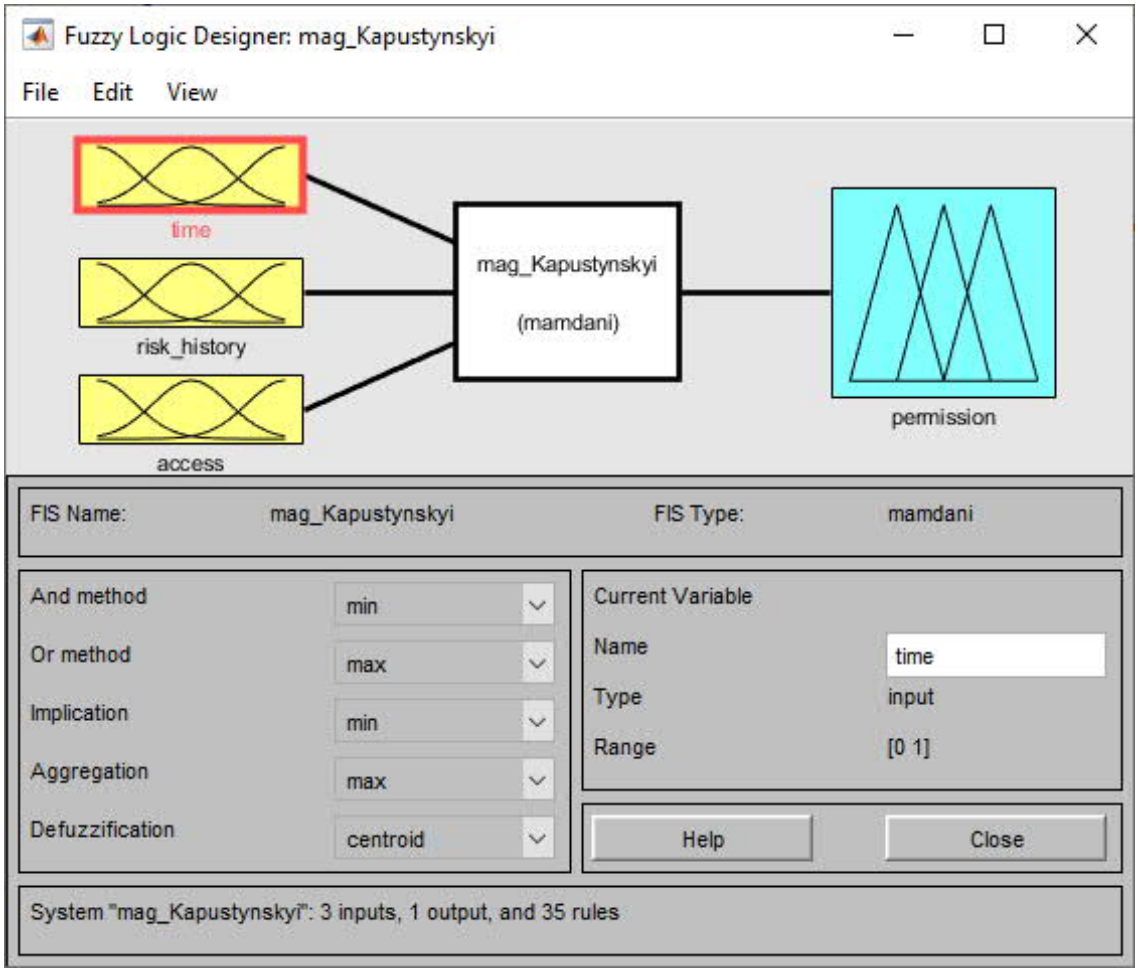

Рисунок 3.1 - Моделювання нечіткої системи у засобі Matlab Fuzzy Logic Toolbox

Другий етап - побудова функцій належності [47]. Для налаштування графіків цих функцій необхідно двічі нажати на змінну входу або виходу. Спочатку необхідно вказати діапазон чисел для змінної у полі “Range”. Також потрібно обрати правильну кількість функцій належності, в залежності від кількості станів для даного входу. Щоб змінити вигляд функції належності необхідно вибрати її

навівши курсором та клікнути клавішею. Переміщуючи точки опори графіків функцій налаштовуємо їхні параметри.

Для входів системи я обрав тип функції “gbellmf”, тому що вони візуально краще відображають зміну значень. Для виходу я застосував функцію типу “trapmf”.

Використовуючи вхідні дані в середовищі Matlab було побудовано функцію належності для входу “time” або “час доступу”, який відповідає за визначення відповідності часу входу до робочого графіку. Вигляд налаштувань даної вхідної змінної зображено на рисунку 3.2.

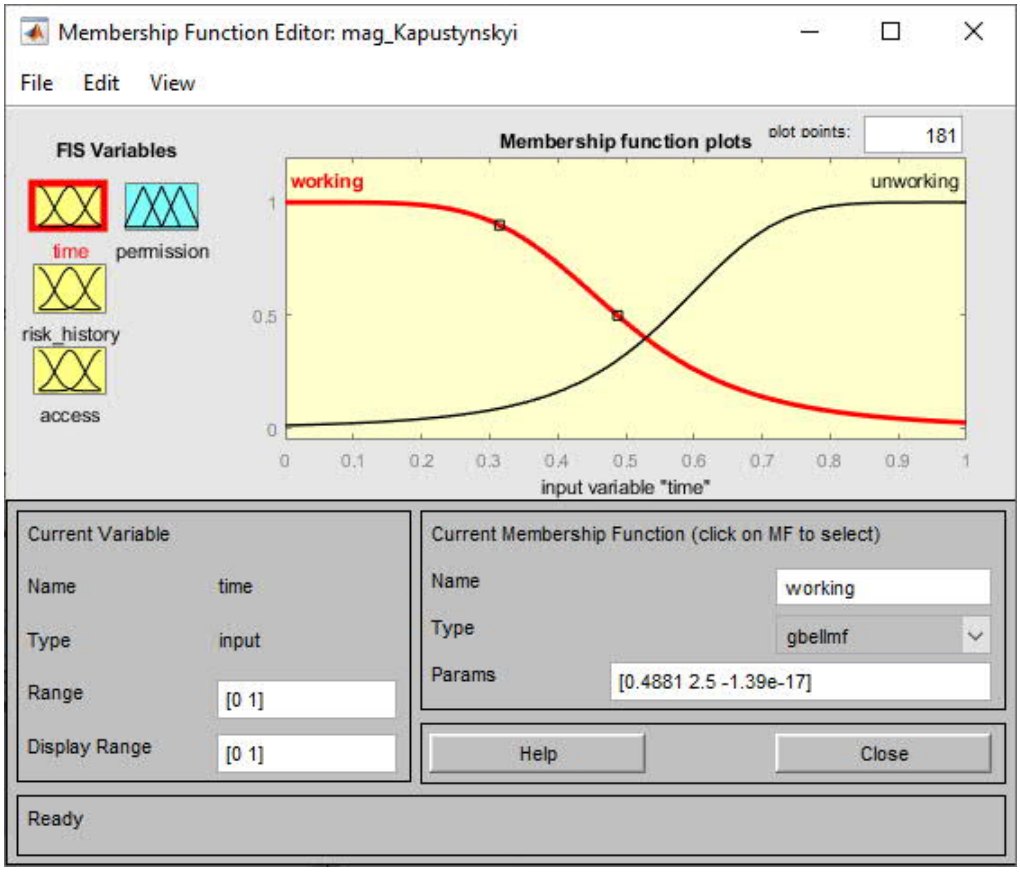

Рисунок 3.2 - Функції належності для входу “час доступу”

Наступний крок - налаштування функції належності для входу “історія ризику” або “risk_history”. У ній було встановленно діапазон [0 100] та 3 стани. Результат зображений на рисунку 3.3.

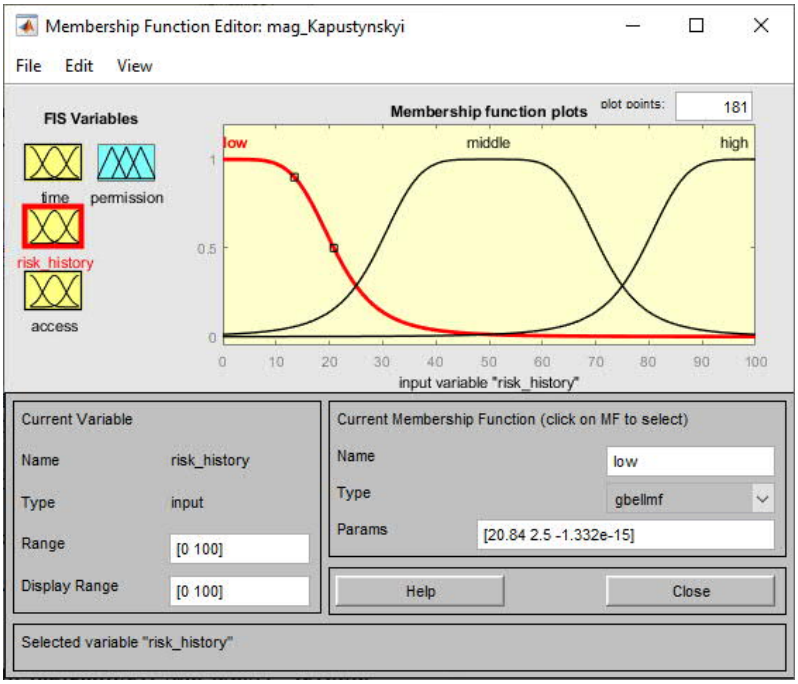

Рисунок 3.3 - Функції належності для входу “історія ризику”

Для останнього, третього по рахунку входу під назвою “режим доступу” або “access” встановлено діапазон [0 1]. Два стани - reading та editing рівномірно розподілені для збалансованого розподілу вхідних чисел. Характеристика даного входу та вигляд функцій належності для входу “режим доступу” або “access” зображено на рисунку 3.4.

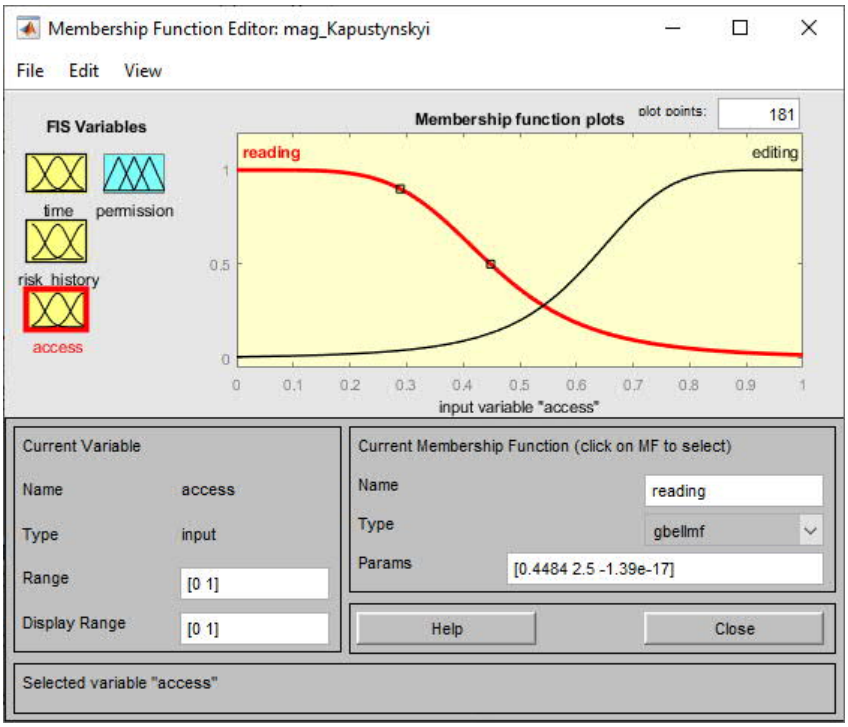

Рисунок 3.4 - Функції належності вхідної змінної “режим доступу”

Вихід розробленої нечіткої системи показує результат, який відображає дозвіл або відмову для користувача. Даний вихід має чотири стани:

1. Читання або reading.
2. Редагування або editing.
3. Аутентифікація або authentication.
4. Відмова.

Вигляд налаштувань та функцій належності для вихідної змінної зображено на рисунку 3.5.

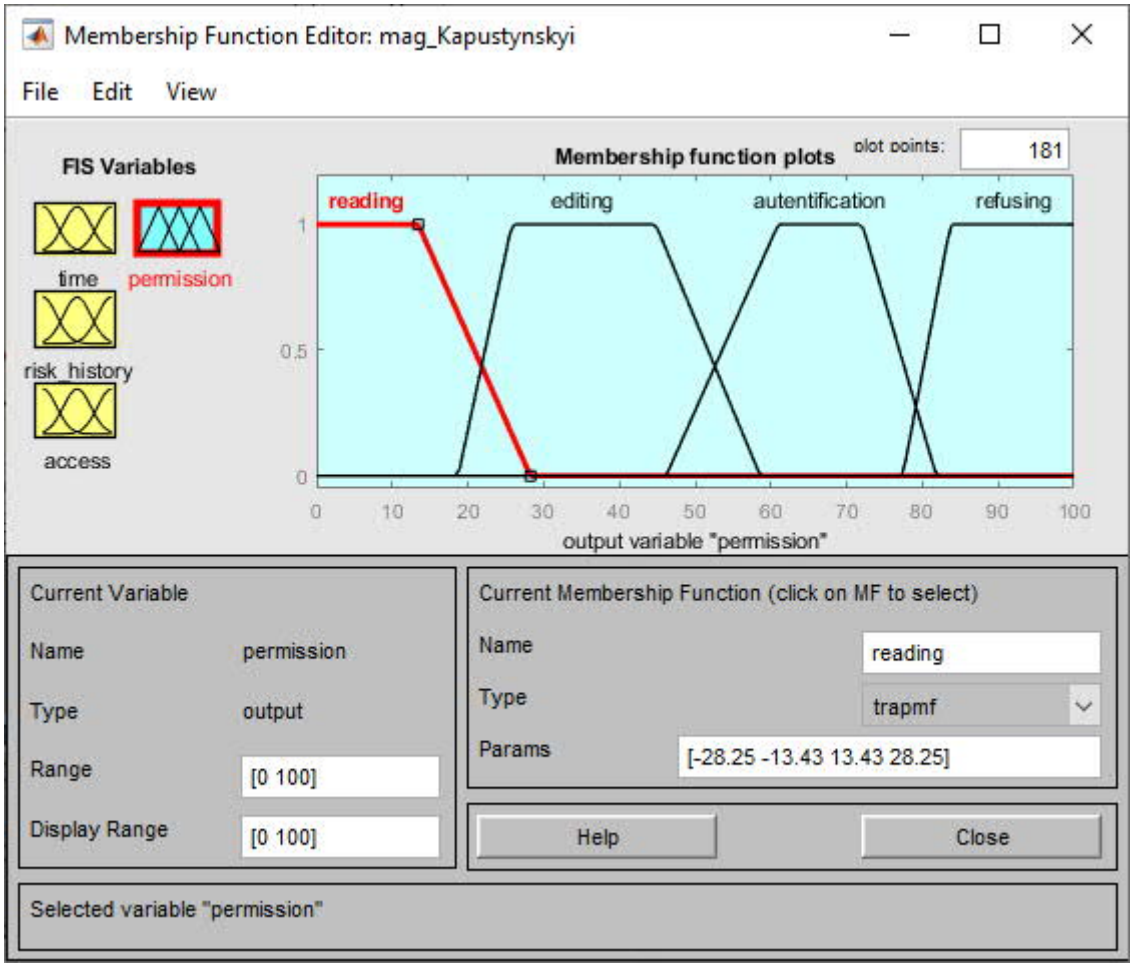

Рисунок 3.5 - Функції належності для вихідної змінної “дозвіл”

Далі, приступаємо до виконання третього етапу - створення бази знань(правил) для нечіткої системи. Вона являється необхідним критерієм для роботи всієї нечіткої системи.

Правила можна додати лише до існуючої системи нечіткого логічного висновку в робочій області Matlab. Це здійснюється функцією “addrule” яка приймає на вхід два параметри:

1. FIS_name - ідентифікатор(ім’я) системи нечіткого логічного висновку.
2. ruleList - матриця правил.

База знань складається із продуктивних правил. Кожне таке правило можна модифікувати окремо від інших. Для створення цих правил ми не будемо писати код та використовувати функцію, а скористаємось вікном “Rule editor” [48]. Щоб відкрити його, необхідно використати пункти меню “Edit” а потім “Rules”. Приклад відкриття вікна редагування правил наведено на рисунку 3.6.

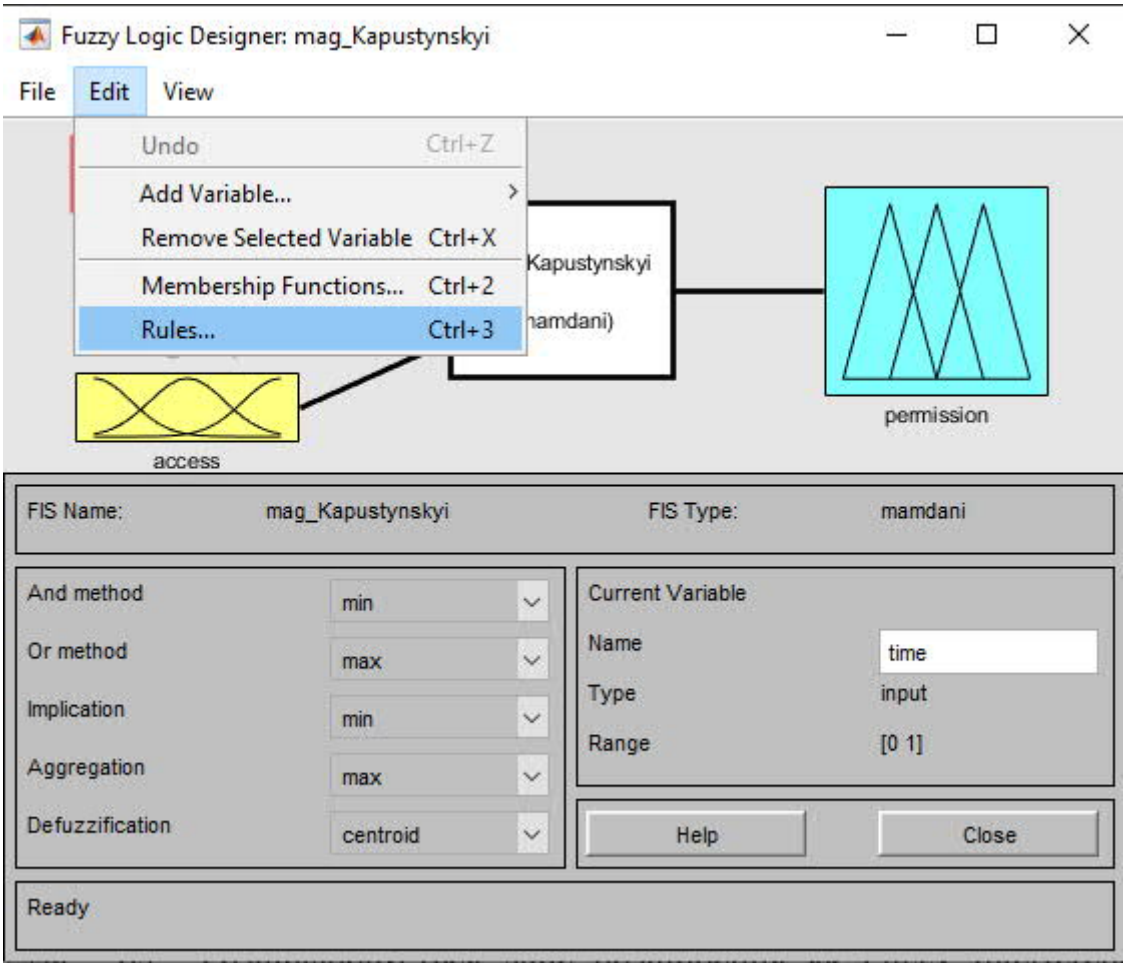

Рисунок 3.6 - Відкриття вікна “Rule editor”

Основною перевагою запису правил та формування бази знань у редакторі є простота запису, можливість їх редагування та видалення. Нечіткі системи, засновані на продукційних правилах, найпоширеніші при моделюванні складних

систем, оскільки в них використовуються лінгвістичні змінні. Ці змінні можуть бути представлені як у нечітких множинах так і у ролі логічних зв’язків між цими множинами.

Нечіткий рівень розуміння та опису складної системи виражається у вигляді певного набору обмежень на виході за певних умов на входах. Обмеження моделюються, як правило, нечіткими множинами та взаємозв'язками, такими як "і", "або", “то"[49]. Оскільки правил зазвичай багато, вони згруповані в набір, який називається базою правил. На рисунку 3.7 зображено інструмент “Rule editor” - редактор правил.

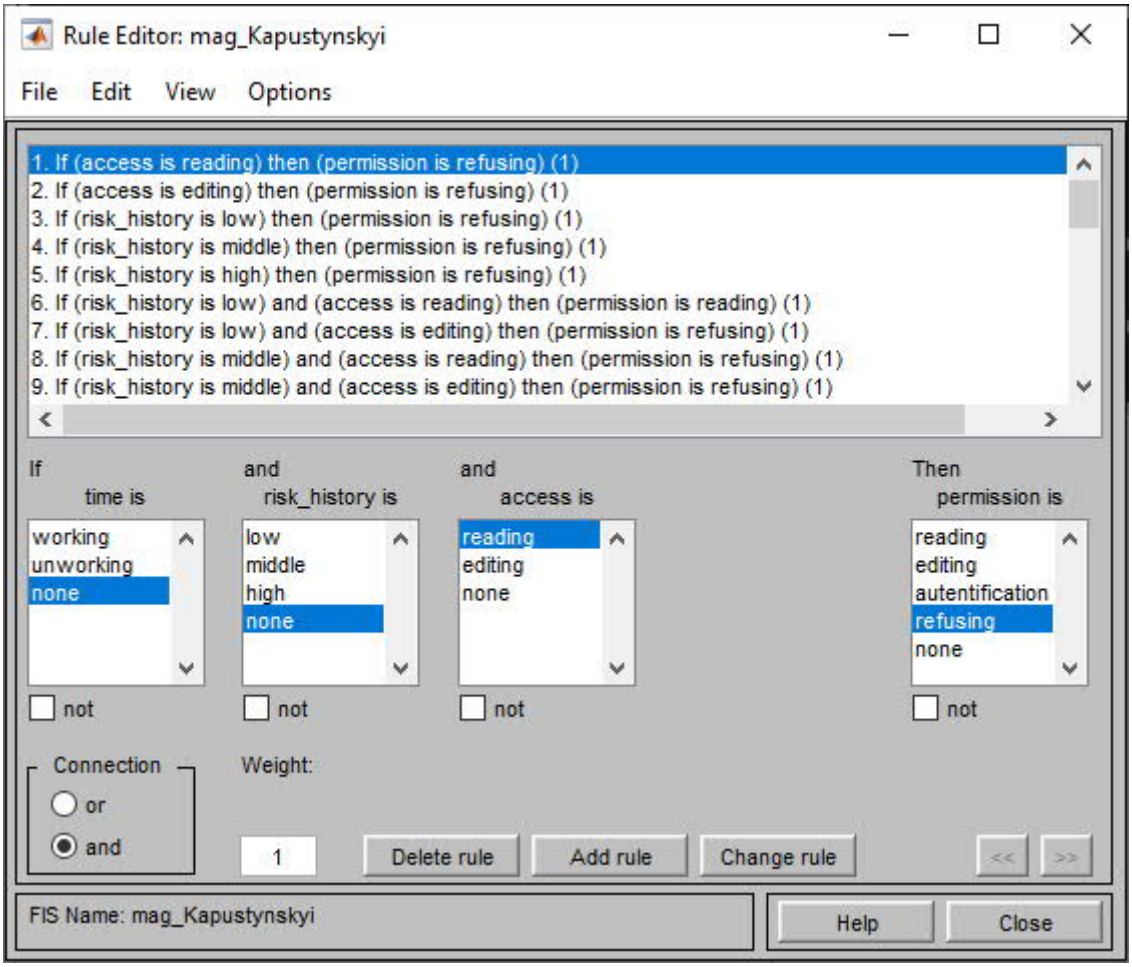

Рисунок 3.7 - Інструмент “Rule editor” для створення бази правил

У вигляді простих відповідностей станів на входах і виходах необхідно скласти повну базу правил. У даному інструменті стани представлені у вигляді слів і це значно спрощує розробку на відміну від написання коду, у якому дані правила необхідно прописувати за допомогою чисел які відповідають станам.

У результаті моделювання ми отримуємо файл з розширенням “.fis”. Розшифровується дане розширення як “Fuzzy Interference System”. Цей файл являє собою програмний код написаний мовою програмування Matlab. Програмний код розробленої нечіткої системи управління доступом зображений у додатку А.

3.2 Тестування роботи нечіткої системи

Тестування роботи нечіткої системи являється необхідним етапом під час її розробки. Воно дозволяє виявити помилки під час моделювання та впевнитись у правильності роботи розробки. Для перевірки нечіткої системи у середовищі Matlab я використав інструменти Rule Viewer та Surface Viewer.

Щоб відкрити вікно Rule Viewer необхідно обрати пункти меню View-Rules у вікні Fuzzy Logic Designer. За допомогою даного інструменту можна побачити агрегування нечіткої системи. Агрегування - це процедура визначення ступеня істинності умов згідно з кожним із правил системи нечіткої системи. На рисунку 3.8 зображено числові результати входів та виходу нечіткої системи відповідно до роботи створених правил.

У таблиці 3.1 наведено декілька прикладів числових значень входів та виходів нечіткої системи управління доступом.

Таблиця 3.1 - Числові значення входів і виходів нечіткої системи

Вхідні дані			Вихідні дані
Час доступу	Історія ризику	Рівень доступу	Доступ
0.127	51.2	0.789	74.4
0.873	62	0.295	74.8
0.175	59.6	0.151	49.7
0.0783	83.7	0.151	81.8
0.127	9.04	0.102	49.6
0.825	47.6	0.825	83.1

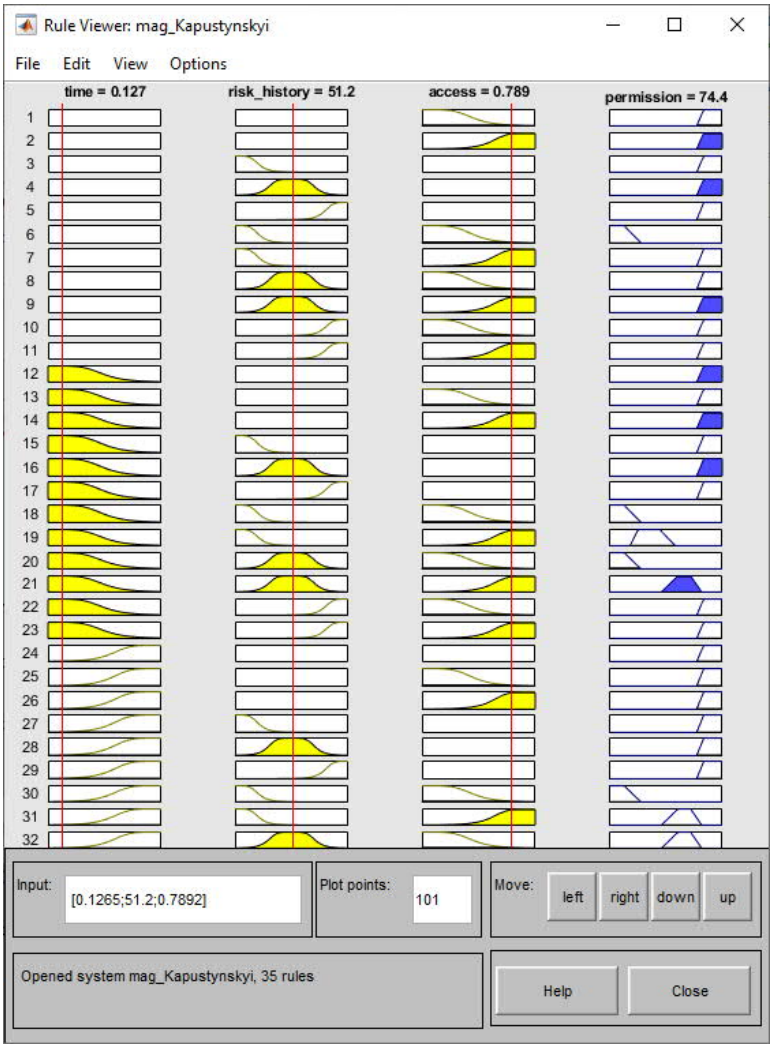

Рисунок 3.8 - Результат роботи бази правил створеної нечіткої системи

Наступним кроком тестування є акумулювання висновків продукційних нечітких правил. Акумуляція або акумулювання в системах з нечіткою логікою - це метод або процес пошуку функції належності для кожної з вихідних лінгвістичних змінних множини. Її метою є поєднання або акумулювання всіх ступенів істинності висновків для отримання функції належності кожної з вихідних змінних. Необхідність виконання даного етапу є те, що висновки, які стосуються однієї і тієї ж лінгвістичної змінної виходу, належать до різних правил системи нечіткого висновку. За допомогою інструменту Surface Viewer можна побачити результат процесу акумулювання. На рисунках 3.9, 3.10 та 3.11 зображено графіки акумулювання відповідно до різних вхідних змінних.

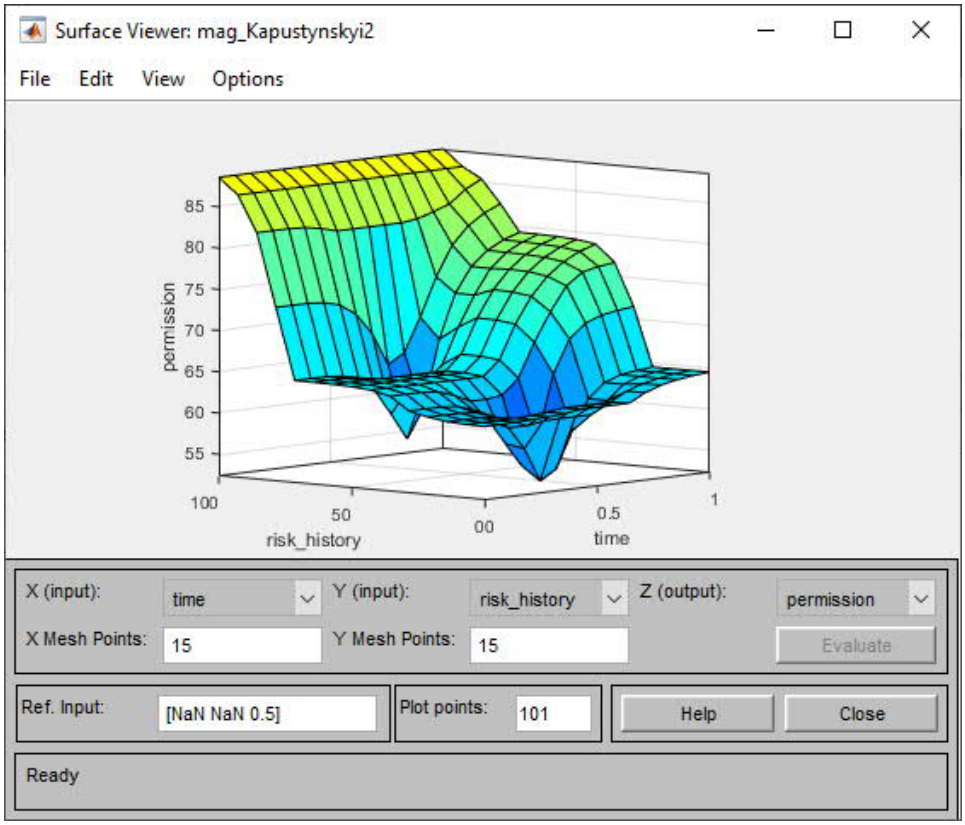

Рисунок 3.9 - Поверхня значень змінної виходу у залежності від змінних входу “Час доступу” та “Історія ризику”

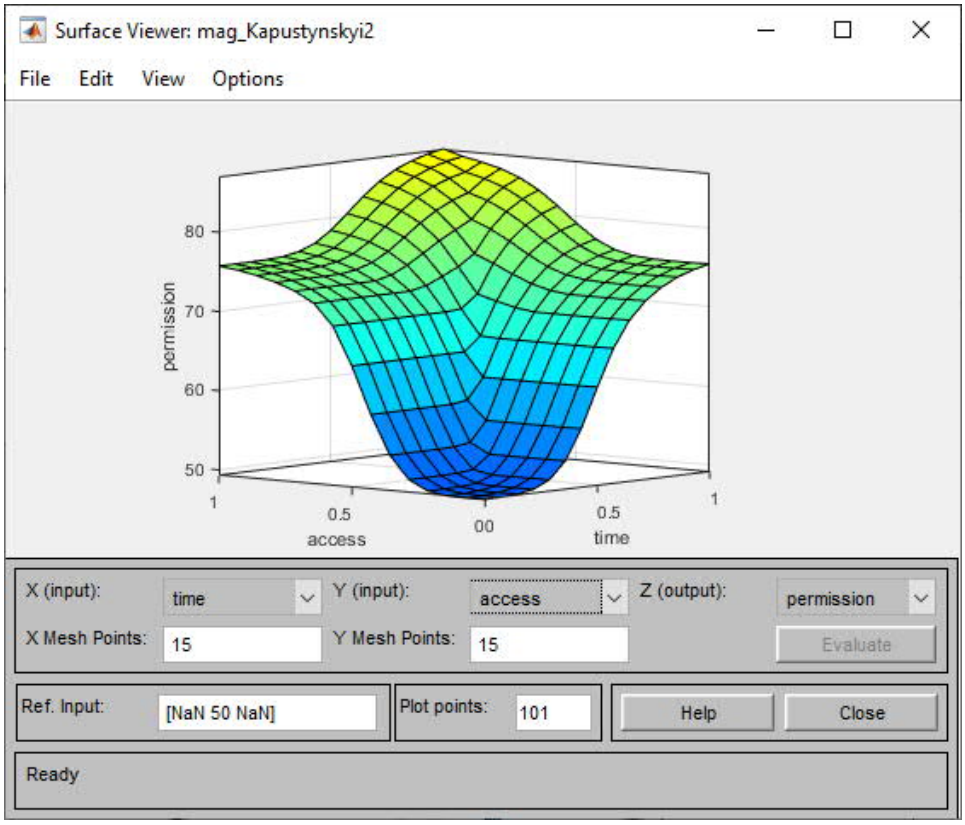

Рисунок 3.10 - Поверхня значень змінної виходу у залежності від змінних входу “Час доступу” та “Рівень доступу”

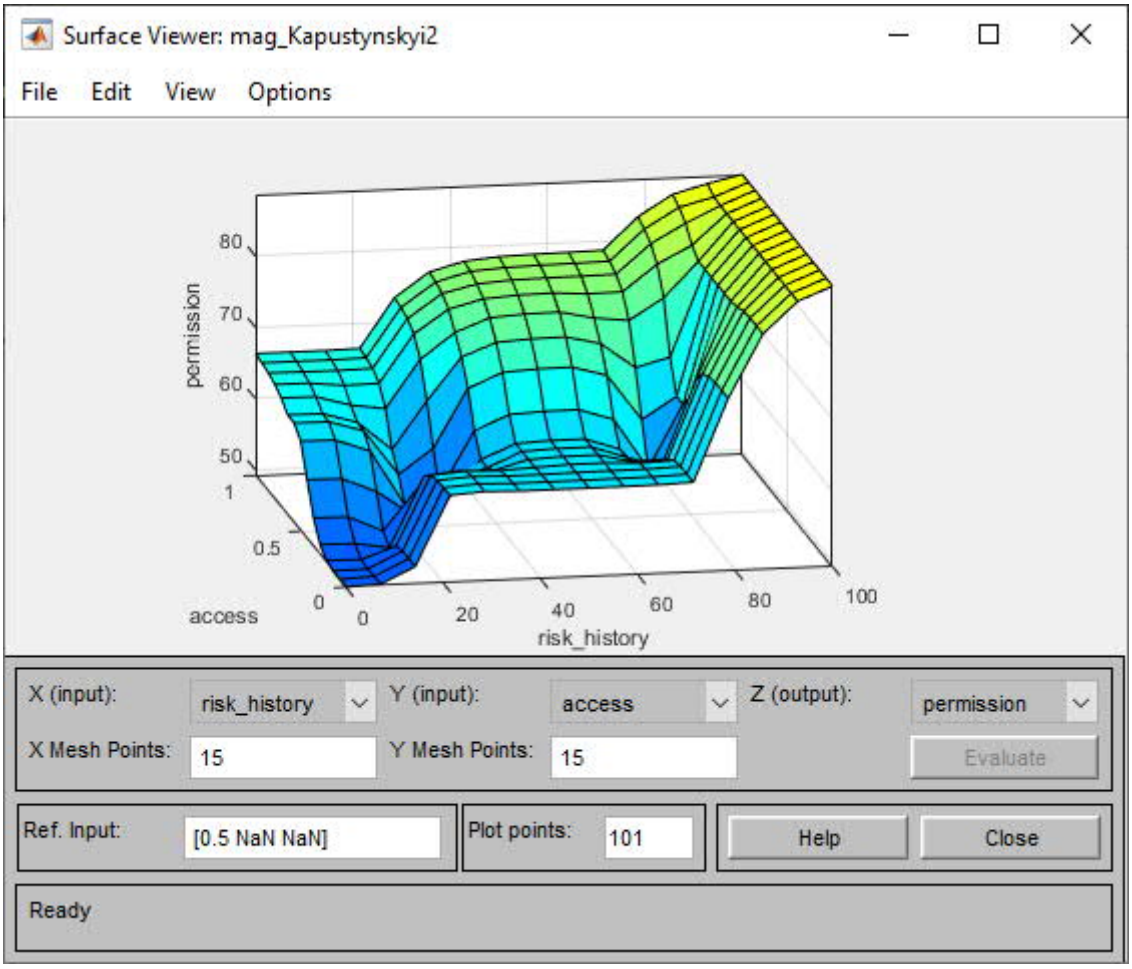

Рисунок 3.11 - Поверхня значень змінної виходу у залежності від змінних входу “Історія ризику” та “Рівень доступу”

Отож, у результаті проведення процесу верифікації розробленої нечіткої системи доступу було переглянуто результати функцій агрегування та акумулювання. Згідно з отриманими результатами зроблено висновок що розроблена система працює згідно до встановлених вимог та розробленої бази продуктивних правил.

3.3 Проектування нечіткого контролера у середовищі Simulink.

Практична реалізація моделі розробленої нечіткої системи контролю доступу до бази даних підприємства здійснюється шляхом проектування нечіткого контролера засобами середовища Simulink версії 10.1.

Щоб запустити Simulink у робочому полі Matlab необхідно виконати команду “simulink” після чого запуститься графічне вікно середовища. Наступним кроком є створення нової моделі. Після створення можна побачити пусту робочу область на якій будуть розміщуватися компоненти схеми. Вигляд програми зображений на рисунку 3.12.

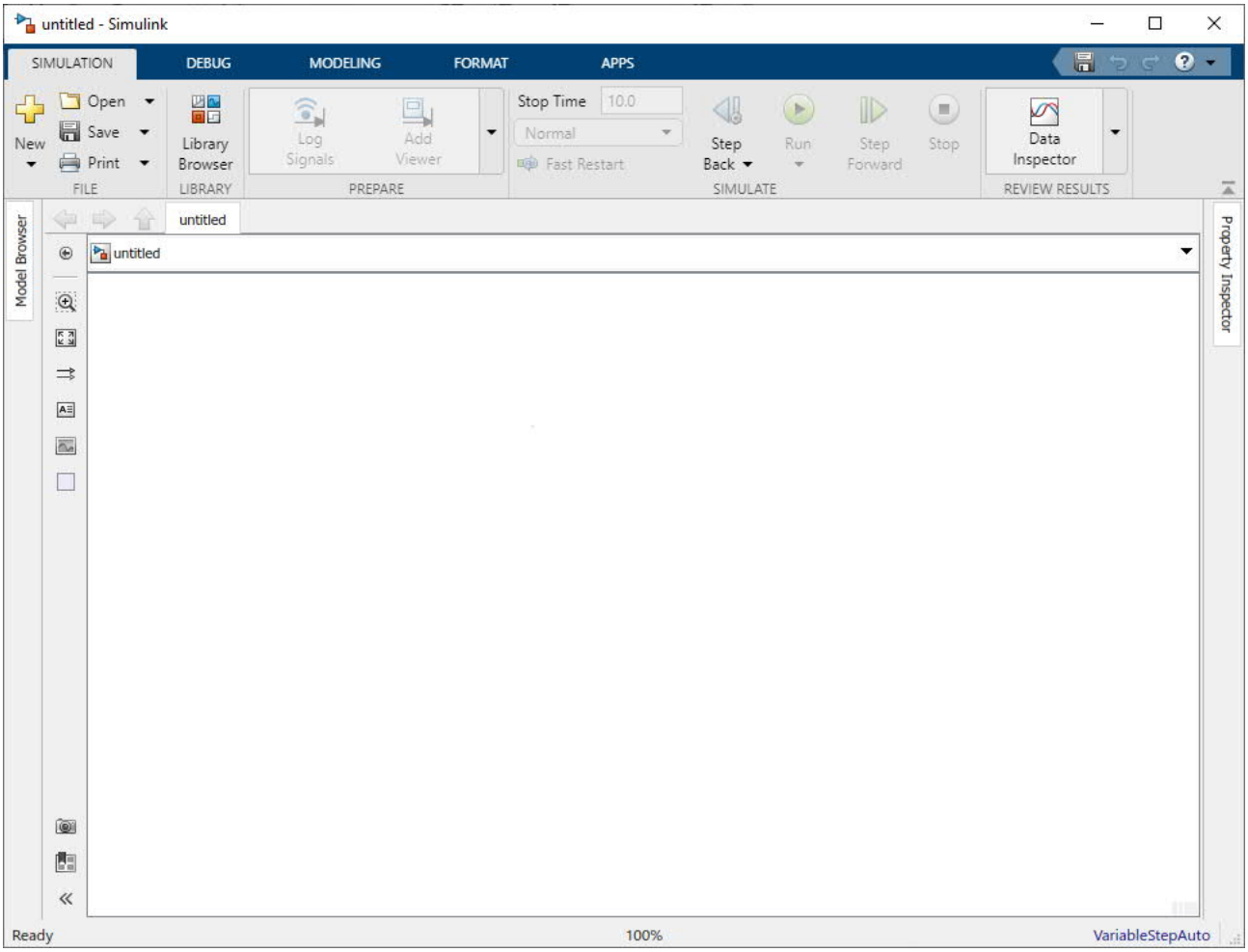

Рисунок 3.12 - Вікно середовища Simulink

Додавання компонентів здійснюється за допомогою вікна “Library Browser”. Його вигляд зображено на рисунку 3.13. У ньому знаходяться усі доступні компоненти які необхідні для побудови схеми. Навігація зроблена у лівій частині вікна шляхом об’єднання компонентів у категорії. Оскільки розроблена система включає у себе нечітку логіку, спершу необхідно додати модуль який буде її обробляти. Для цього у списку компонентів Simulink існує спеціальна категорія “Fuzzy Logic Toolbox”. До її складу входять наступні блоки:

1. Категорія Membership Functions, яка містить у собі блоки функцій належностей, призначених для побудови нетипових нечітких контролерів. Рисунок 3.14 зображує блоки, які доступні у даній категорії.

2. Fuzzy Logic Controller - контроллер нечіткого виводу.

3. Fuzzy Logic Controller with Ruleviewer - нечіткий контроллер з вікном RuleViewer для перегляду правил бази знань під час симуляції.

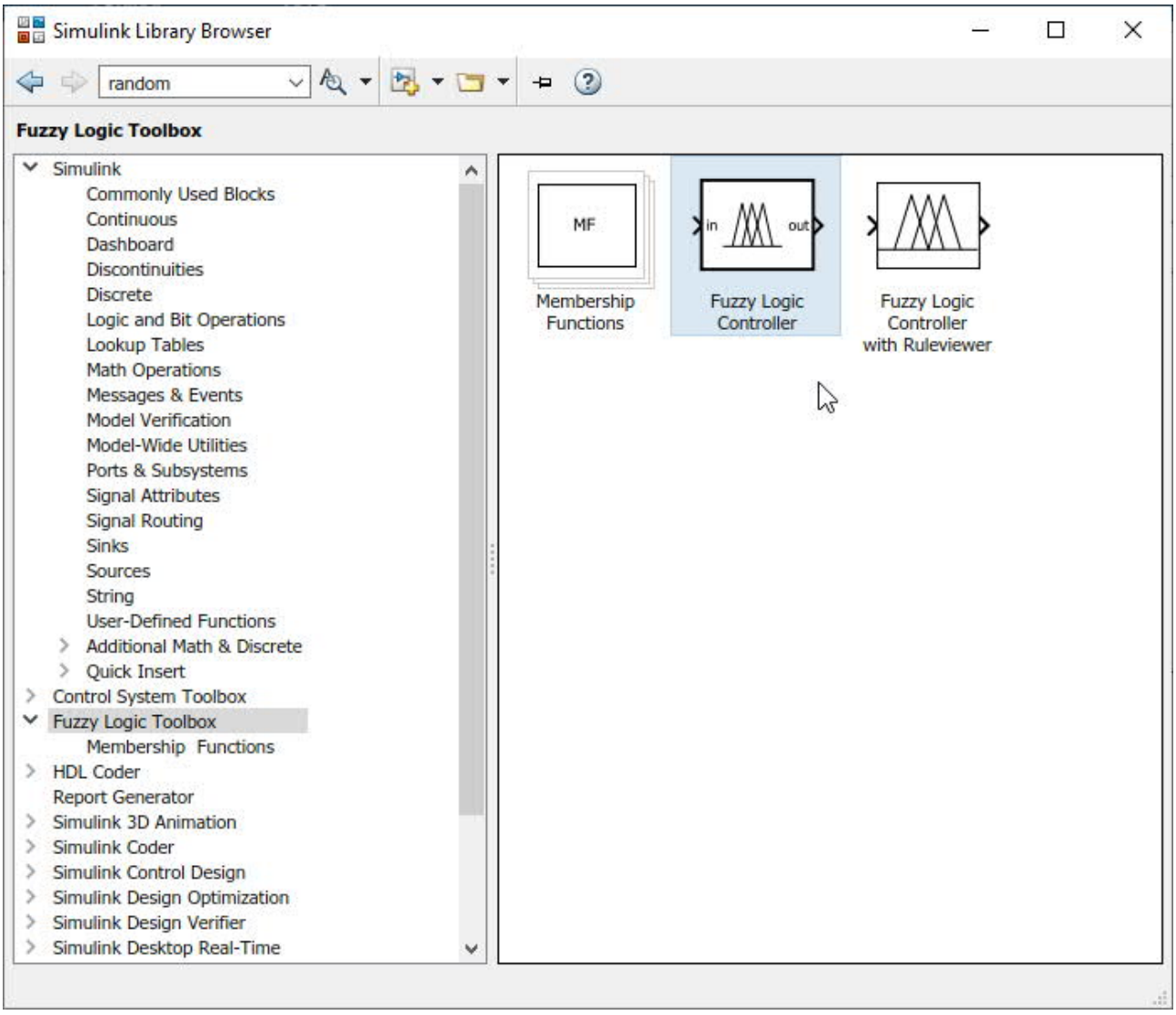

Рисунок 3.13 - Огляд компонентів категорії “Fuzzy Logic Toolbox” у вікні “Simulink Library Browser”

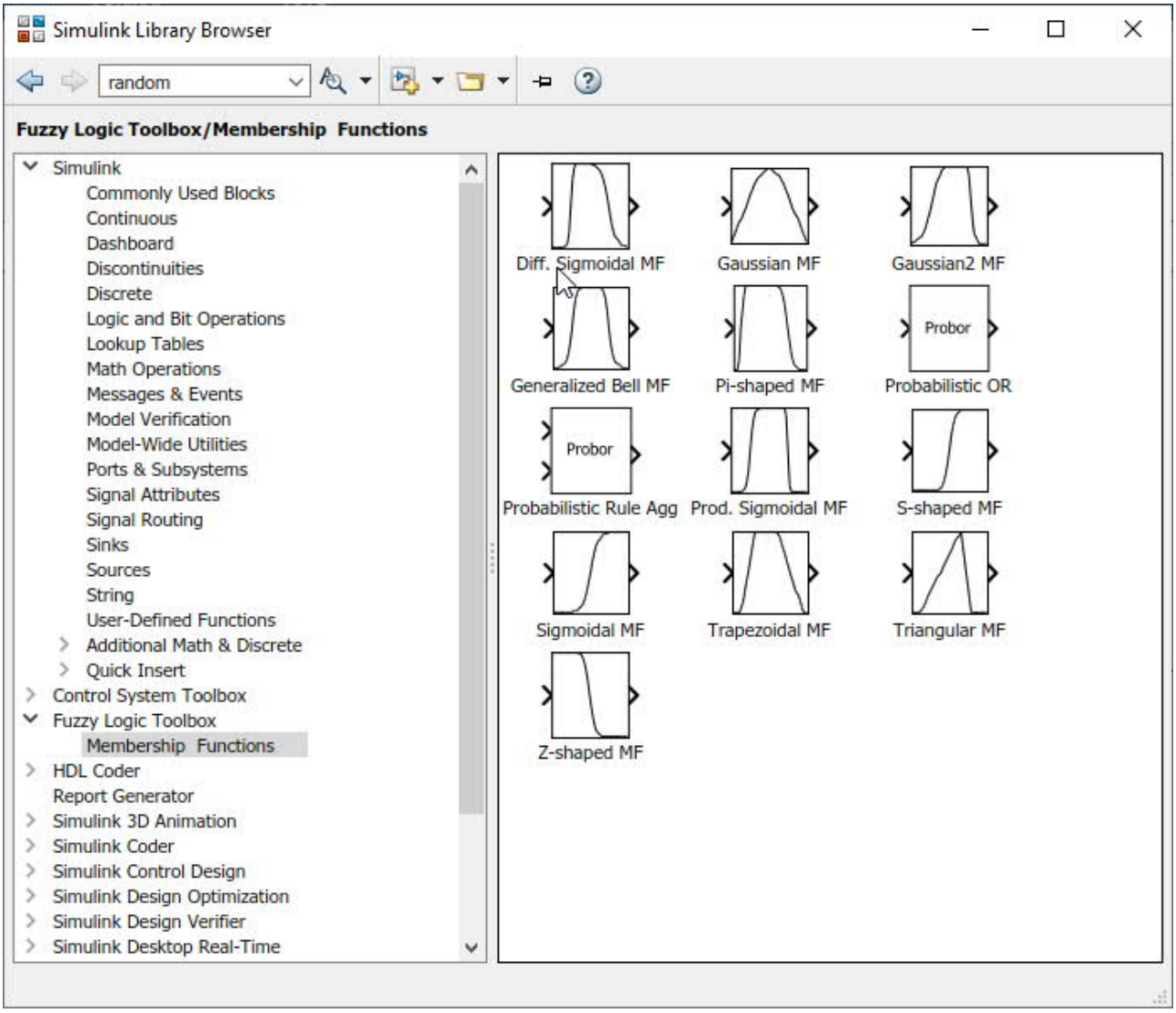

Рисунок 3.14 - Огляд компонентів категорії Membership Functions

До своєї схеми я додав контроллер нечіткої логіки під назвою - “Fuzzy Logic Controller”. Вхідними даними для його роботи є файл з розширення “.fis” який було отримано у результаті пункту 3.1. Здійснивши подвійний клік на компоненті відкривається вікно з налаштуваннями, де необхідно вписати назву файлу. Процедура налаштування контроллера зображена на рисунку 3.15.

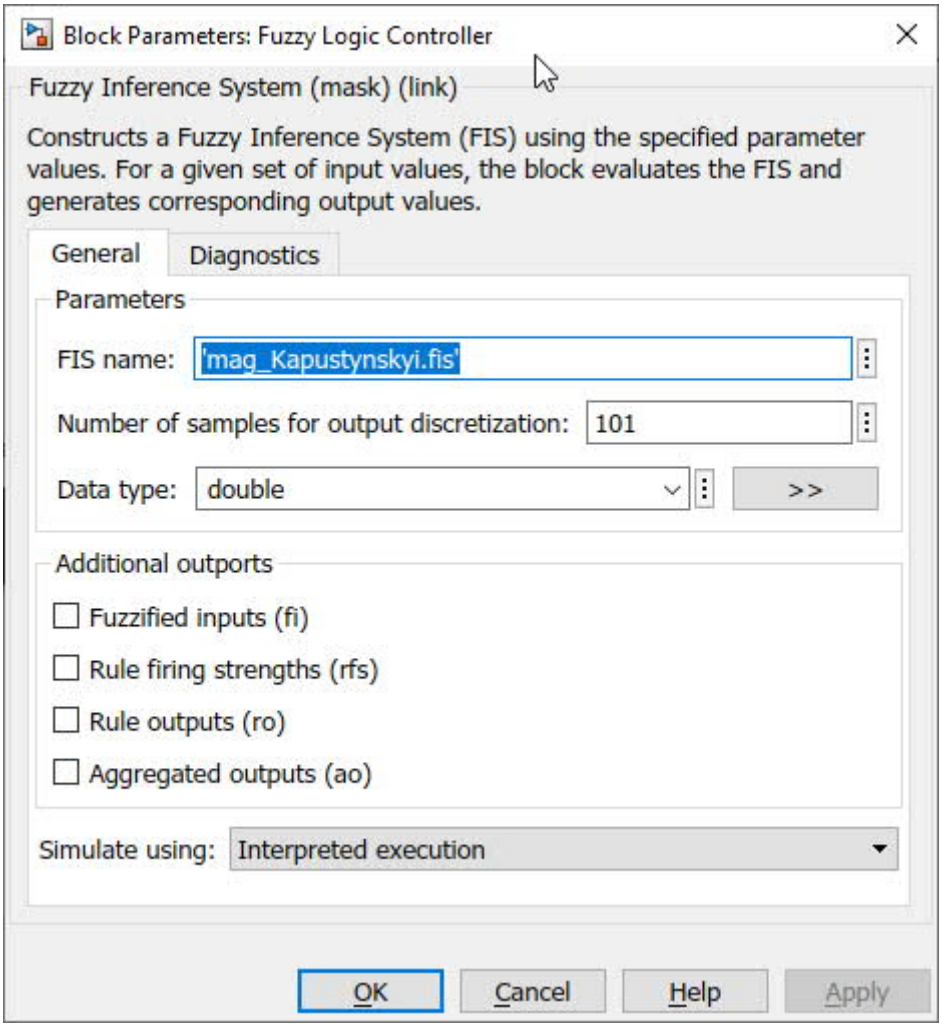

Рисунок 3.15 - Налаштування нечіткого контролера

Наступним кроком є додавання компонентів генерації вхідних змінних. Для цього було використано Uniform Random Number, який генерує рівномірно розподілені випадкові числа за вказаний інтервал.[matlab link] Якщо у нечіткій системі є декілька вхідних даних, то їх необхідно мультиплексувати перед входом у систему. Також, виходи нечіткої системи будуть представлені у вигляді однієї мультиплексної лінії у випадку наявності більше одного виходу. Для зчитування значень входів та виходу я використав компоненти Scope, які підключаються паралельно до джерел замірів. Схема розробленого контролера нечіткої логіки виводу зображена на рисунку 3.16.

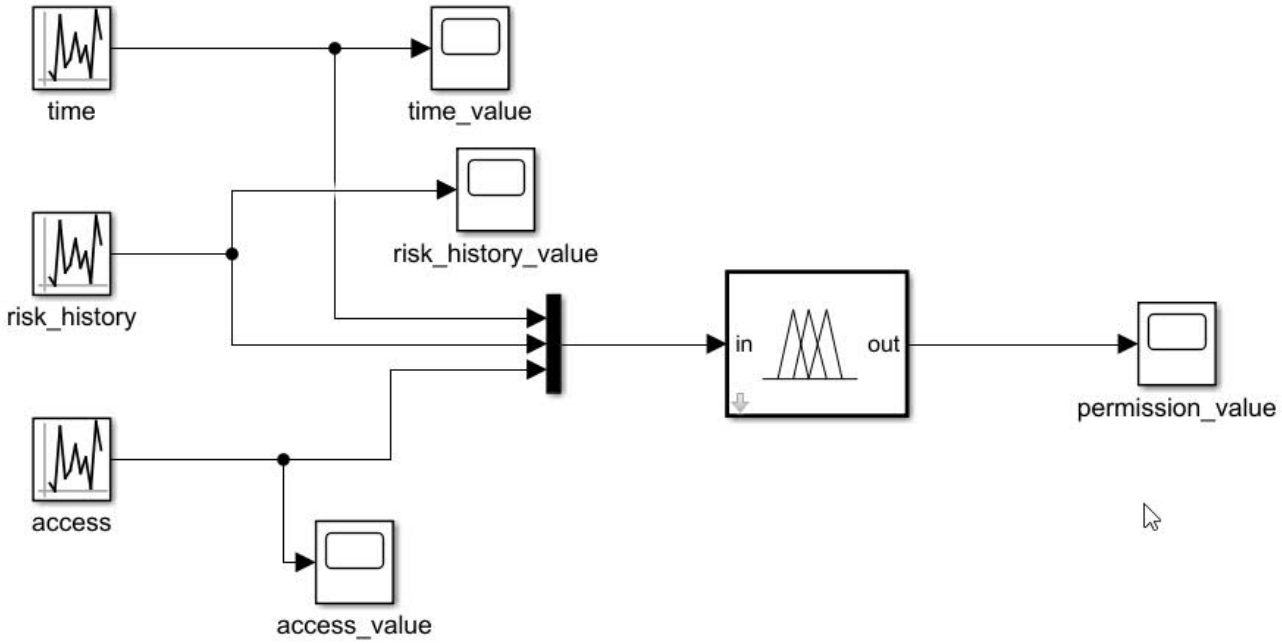

Рисунок 3.16 - Схема контролера нечіткої логіки виводу

Схема задання функцій належності для вхідних змінних зображена на рисунку 3.17. У ній відображається трапецевидне задання функції належності вхідних змінних і 3 множини розподілу: high, medium та low.

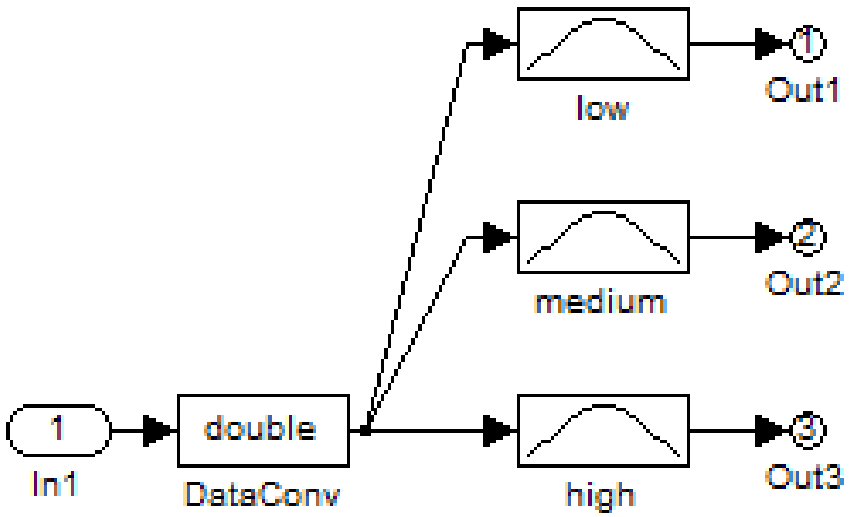

Рисунок 3.17 - Схема задання функцій належності для вхідних змінних

На рисунку 3.18 зображена структура правил нечіткого контролера. Входами цих правил є змінні time, risk_history та access.

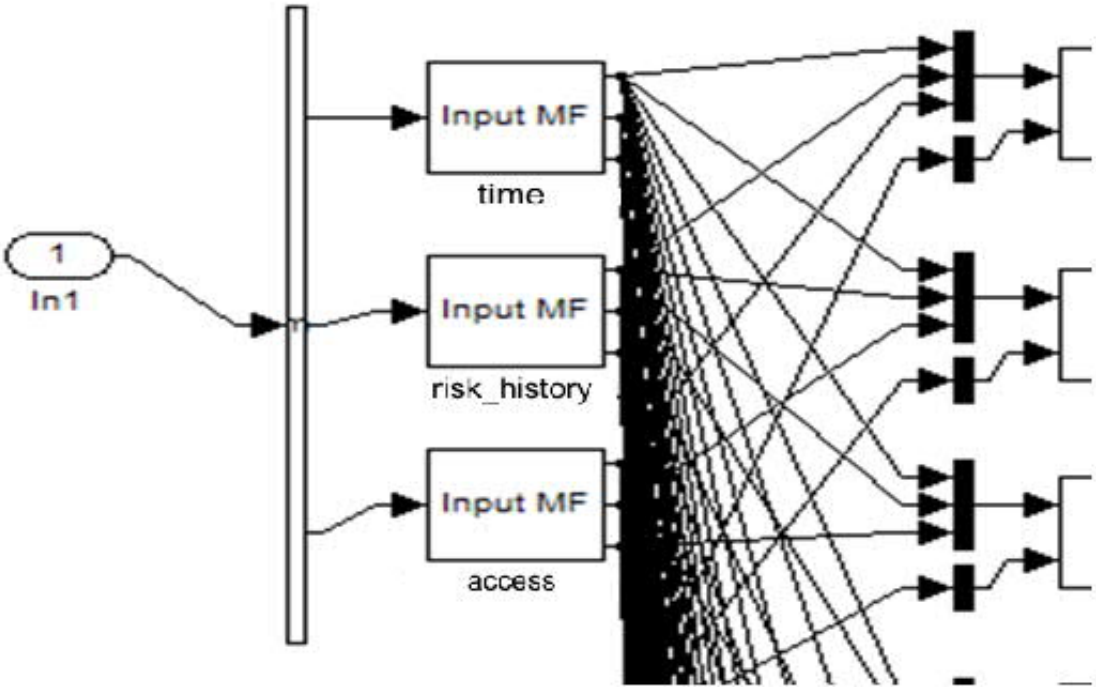

Рисунок 3.18 - Вхідні змінні правил нечіткого контролера

На даному рисунку видно, що вхідні змінні мультиплексуються для обробки нечітким контролером(In 1). Також, дані з кожного входу та виходу надходять до усіх правил, тому кожне з них має 35 виходів, що відповідає кількості правил у базі знань. Результатами кожного правила є вихідні змінні reading, editing, authentication та refusing(рис. 3.19).

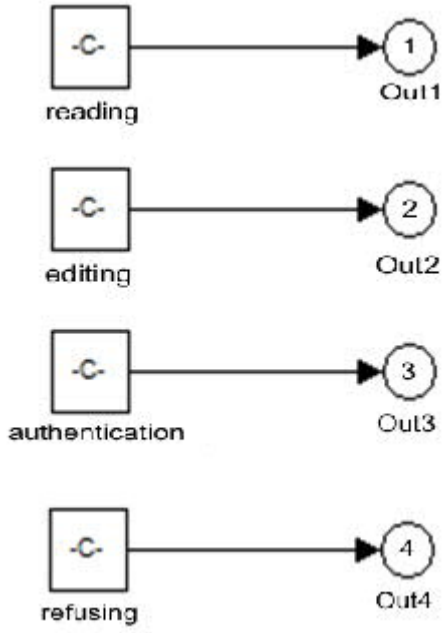

Рисунок 3.19 - Вихідні дані правил нечіткого контролера

Реалізація одного правила у вигляді схеми зображена на рисунку 3.20.

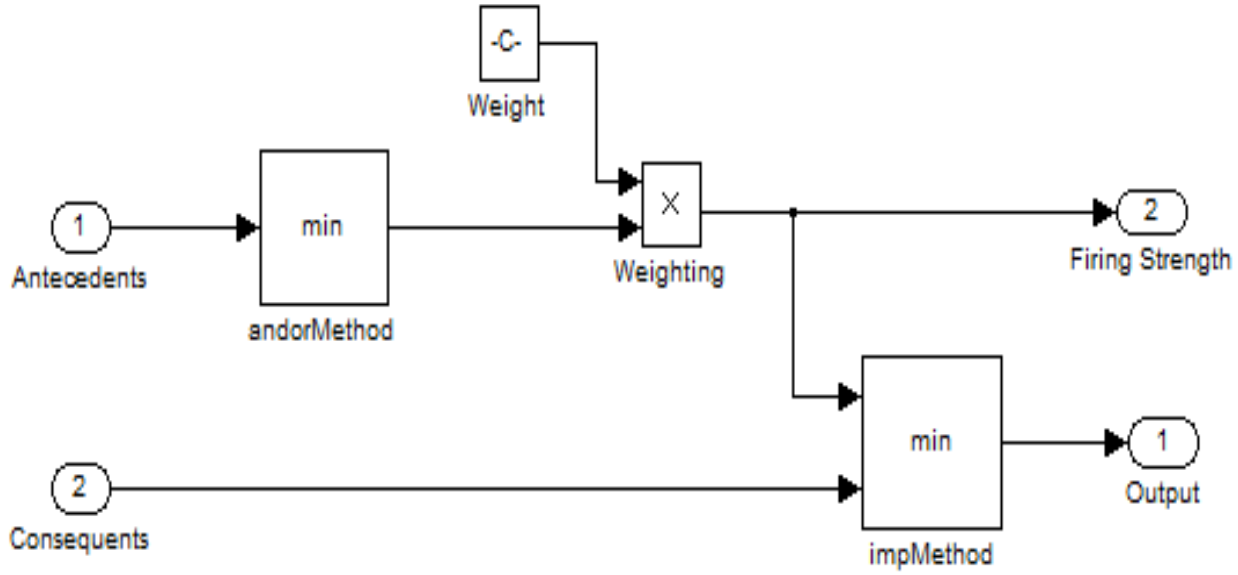

Рисунок 3.20 - Схема правила типу “якщо-то” розробленої нечіткої системи

У цій схемі чітко видно, що обробка вхідних значень полягає у визначенні мінімального значення. Крім того, Simulink враховує вагу правила при його обробці. У випадку розробленого нечіткого контролера доступу до бази даних всі правила у базі знань мають однакову вагу, тому в цьому випадку дана частина схеми не є інформативною.

У загальній схемі нечіткого контролера є 35 правил внутрішні схеми яких ідентичні. Після обробки вхідних та вихідних змінних за допомогою бази правил знайдено максимальне значення функцій приналежності вихідної змінної (блок MAX). Крім того, для перевірки правильної роботи нечіткого контролера ті ж виходи мультиплексуються за допомогою блоку Total Firing Strange. Після обробки бази правил нечіткий контролер виконує дефазифікацію, тобто приводить нечіткий логічний висновок до чіткості (рис. 3.21).

Для здобуття висновку за допомогою механізму Мамдані нечіткий контролер виконує дефазифікацію, тобто знаходження центру ваги кінцевої фігури, яка формується підсумовуванням виходів 35 правил. Формула яка реалізує дефазифікацію зображена під номером 3.1.

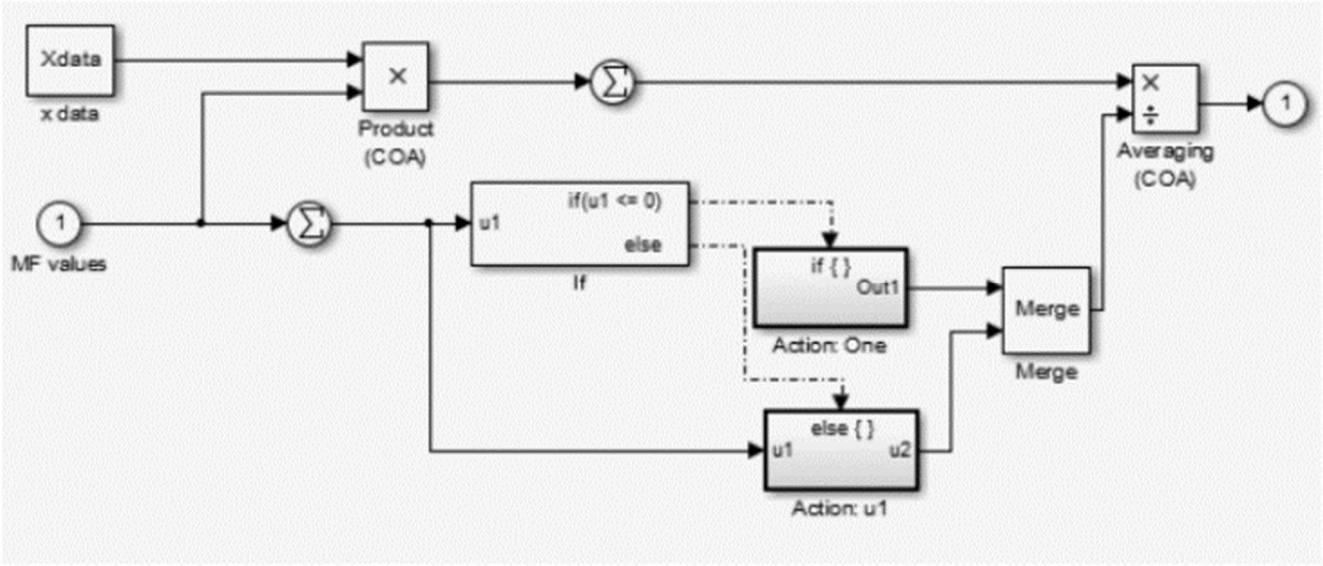

Рисунок 3.21 - Схема дефазифікації виходу нечіткого контролера

$$r_{цв} = \frac{\sum_{j=1}^{m} r_j \mu(r_j)}{\sum_{j=1}^{m} \mu(r_j)}, \qquad (3.1)$$

де $r_{цв}$ - значення абсциси центру ваги кінцевої фігури;

m - кількість прямокутників, на які поділено кінцеву фігуру;

r_j - значення абсциси;

$\mu(r_j)$ - значення ординати j-ї фігури.

Фрагмент схеми нечіткого контролера, який показує дефазифікацію та вивід виходу, перевірку правильності, показаний на рисунку 3.22.

Щоб перевірити правильність роботи нечіткого контролера, спочатку необхідно порівняти отриману довжину бітової послідовності виходів кожного правила з нулем, а потім із середнім заданим значенням. На виході нечіткого контролера є значення, отримане після обробки блоку дефазифікації та блоку перевірки.

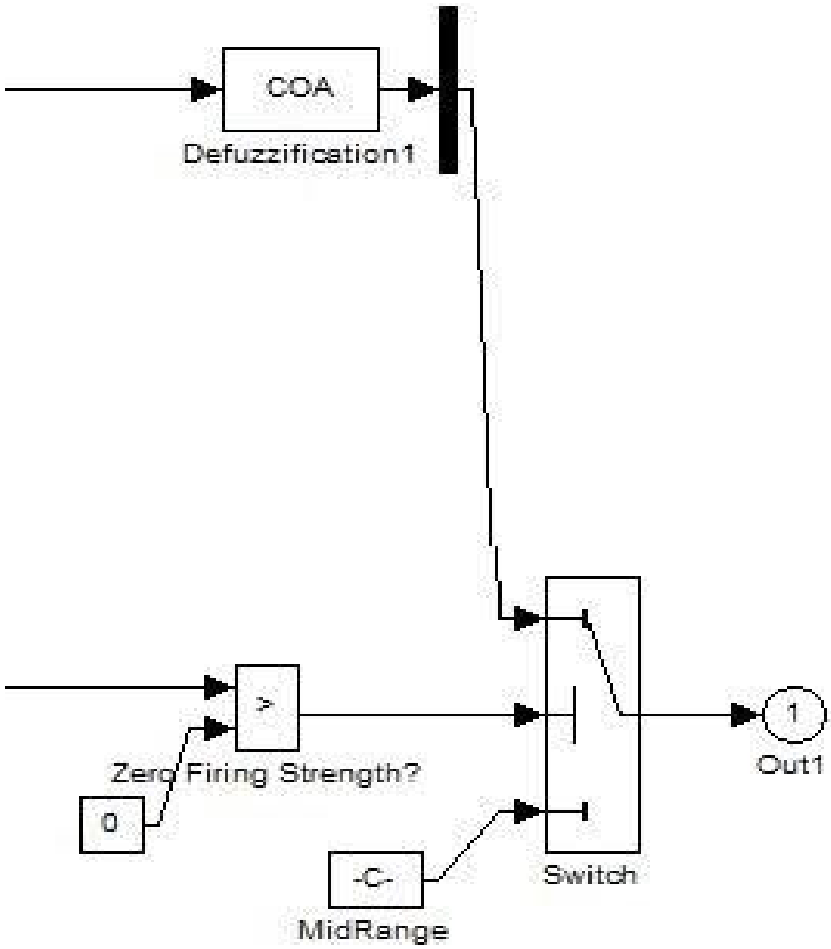

Рисунок 3.22 - Частина схеми нечіткого контролера, яка зображує вивід результату його роботи

3.4 Тестування правильності роботи моделі нечіткого контролера

Тестування правильності роботи нечіткого контролера для системи контролю доступу до бази даних здійснюється шляхом порівняння вихідних значень із значеннями, які задані вхідними змінними. Значення вхідних змінних встановлюються випадковим чином з рівномірним розподілом за допомогою генераторів нормалізованих випадкових чисел, що дає змогу встановити чіткі рамки діапазону мінімальних та максимальних чисел. Для забезпечення неповторності сигналів такі генератори використовують так зване “зерно” або “seed”, яке дозволяє задати послідновність чисел які будуть використані для

створення випадкової послідовності. При використанні одного і того ж зерна, завжди буде досягнуто однакової випадкової послідовності. Тому для кожного генерато мною було використано унікальні послідовності чисел для задання “зерна”, що забезпечило неповторність вхідних даних. Для зняття показів було використано компонент Scope який показує сигнали які генеруються під час виконання симуляції. Вигляд сигналів які було згенеровано для входів time, risk_history та access під час симуляції показано на малюнках 3.23, 3.24 та 3.25.

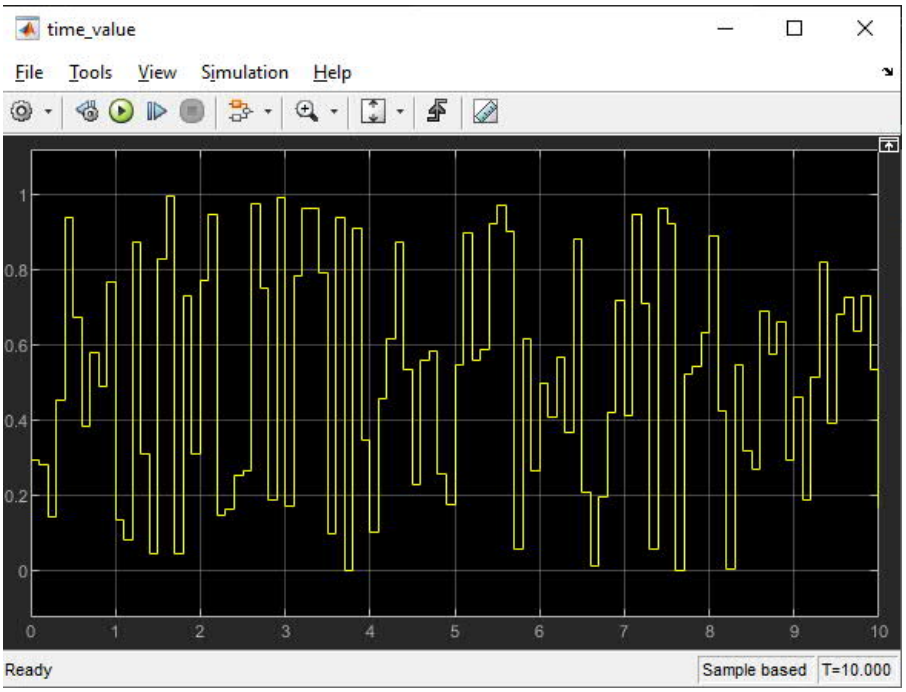

Рисунок 3.23 - Значення входу “time”

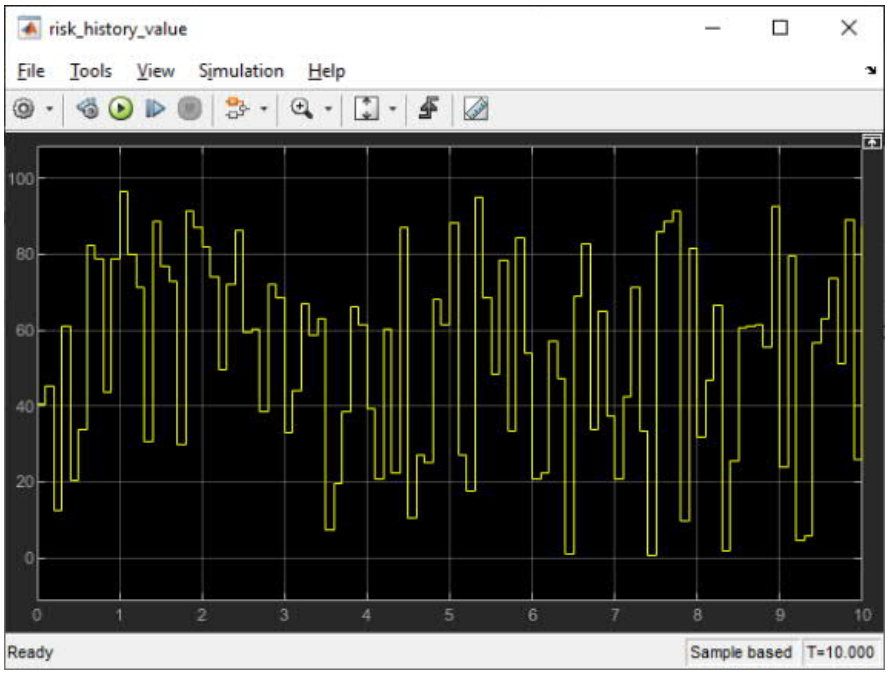

Рисунок 3.24 - Значення входу “risk_value”

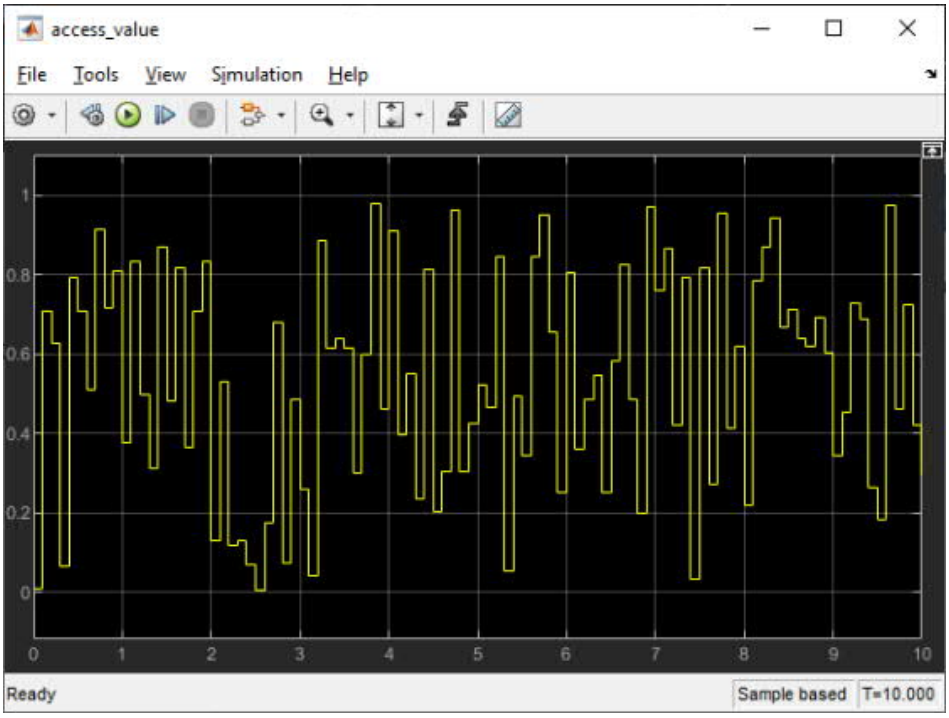

Рисунок 3.25 - Значення входу “access”

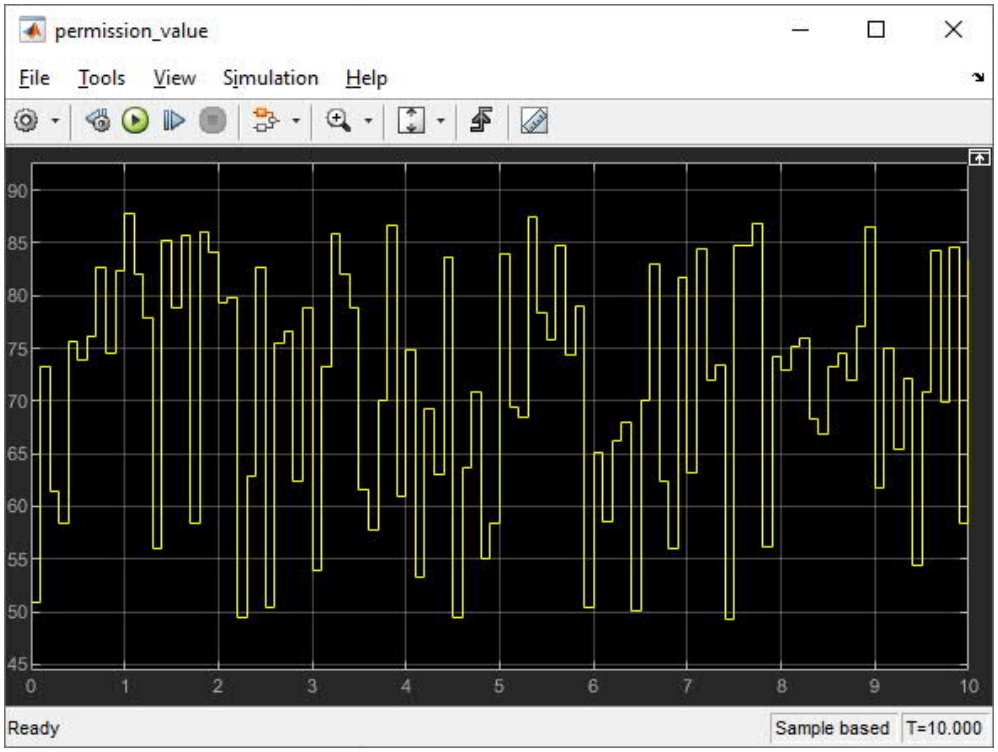

Рисунок 3.26 - Значення виходу контролера управління доступом до бази даних

За допомогою інструменту Rule Viewer який ми використовували під час тестування роботи розробленої моделі нечіткої системи, виставимо на входи значення із входів Simulink та таким чином перевіримо правильність результатів виводу. На першій ітерації дані входів time, risk_history, access наближено

дорівнюють 0.3, 40, 0. Виставивши ці дані у інструменті Rule Viewer отримуємо значення виходу 51.2 що наближено до результату у Simulink (рисунок 3.27).

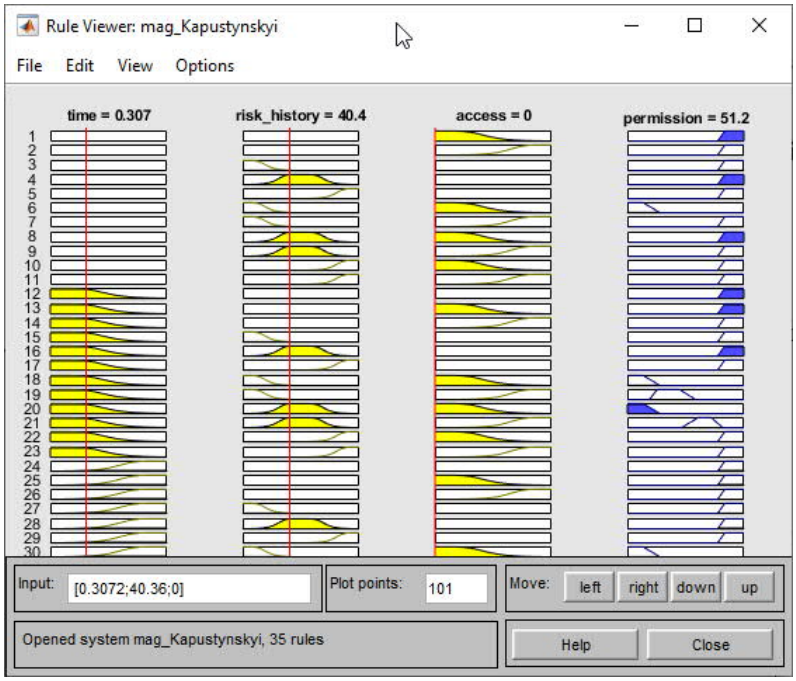

Рисунок 3.27 - Значення виходу нечіткої системи при вхідних значеннях 1 ітерації симуляції роботи контроллера

Для уточнення, мною було перевірено ще 4 ітерацію симуляції. Значення вхідних змінних та результату зображені на рисунку 3.28.

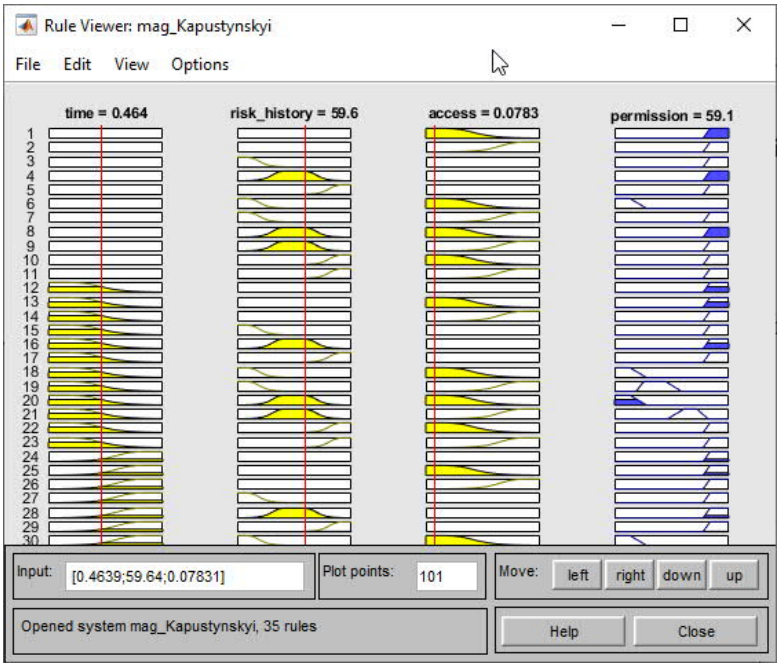

Рисунок 3.28 - Значення виходу нечіткої системи при вхідних значеннях 4 ітерації симуляції роботи контроллера

Таким чином було протестовано роботу нечіткого контролера системи доступу до бази даних. Результат симуляції роботи контролера повторює результати розробленої нечіткої системи, тому можна вважати що контроллер працює правильно та підтверджує очікування.

3.5 Висновки до розділу

У даному розділі було реалізовано нечітку системи контролю доступу до бази даних у середовищі Matlab та виконано наступний перелік задач:

- спроектовано структуру входів та виходів нечіткої системи за допомогою бібліотеки Fuzzy Logic Toolbox;
- задано функції належності для входів та виходів нечіткої системи:
- сформовано базу знань розробленого алгоритму у редакторі “Rule Editor”;
- проведено фазифікацію входів системи;
- виконано агрегацію підумови в нечітких правилах;
- проведено активізацію підзаключень;
- проведено дефазифікацію вихідних змінних правил бази знань;
- спроектовано нечітких контролер контролю доступу;
- протестовано роботу нечіткої системи та нечіткого контролера.

ВИСНОВКИ

В результаті проведених досліджень було розроблено алгоритм контролю доступу до бази даних підприємства на основі нечіткої логіки. При цьому отримано наступні результати:

1. Проаналізовано застосування нечіткої логіки в базах даних.
2. Проведено аналіз існуючих нечітких систем, виділено їх основні характеристики та перспективи розвитку, що дало базис для створення власного алгоритму контролю доступу на нечіткій логіці.
3. Розроблено алгоритм контролю доступу до бази даних підприємства.
4. Досліджено бази знань та сформовано 35 правил для роботи нечіткої системи.
5. Розроблено модель системи контролю доступу у середовищі MATLAB.
6. Здійснено проектування нечіткого контролера у середовищі Simulink.
7. Проведено тестування та верифікацію роботи системи та контролера, що дало можливість підтвердити працездатність роботи нечіткої системи управління доступом.

Розроблена нечітка система може бути реалізована як програмний засіб для використання в сучасних підприємствах чи організаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What Is Cyber Security and Why It’s so Important Nowadays?: веб-сайт . URL: https://www.fbponline.com/news/importance-of-cyber-security/ (дата зверення: 10.11.2020)
2. Katrin Sarap. Three reasons why we need strict data protection regulations: веб-сайт. URL: https://www.njordlaw.com/three-reasons-why-we-need-strict-data-protection-regulations (дата звернення: 10.11.2020)
3. Roy Maurer. Top Database Security Threats and How to Mitigate Them: веб-сайт. URL: https://www.shrm.org/resourcesandtools/hr-topics/risk-management/pages/top-database-security-threats.aspx (дата звернення: 10.11.2020)
4. What is Access Control in Database Security?: веб-сайт. URL: https://www.datasunrise.com/blog/professional-info/what-is-access-control/ (дата звернення 10.11.2020)
5. І.Я. Співак. Методи та засоби наукових досліджень в інженерії програмного забезпечення. Тернопіль: ТНЕУ, 2015. – 20 с.
6. Капустинський Р.І, Щирба З.В. Аналіз застосування нечітких систем для управління доступом до ресурсів: збірник публікацій III науково-практичної конференції. «Інтелектуальні комп’ютерні системи та мережі». Тернопіль, 2020. С.16.
7. Капустинський Р.І, Щирба З.В. Сучасні застосування нечітких систем: збірник публікацій III науково-практичної конференції. «Інтелектуальні комп’ютерні системи та мережі». Тернопіль, 2020. С.17.
8. Donal Tobin. Which Modern Database Is Right for Your Use Case?: веб-сайт. URL: https://www.xplenty.com/blog/which-database/ (дата звернення: 15.11.2019)
9. Thomas Connolly, Carolyn. Begg. Database Systems: A Practical Approach to Design Implementation and Management – 2015 р. - №6 - С. 64
10. IBM. Relational Databases Explained: веб-сайт. URL: https://www.ibm.com/cloud/learn/relational-databases (дата звернення: 15.11.2019)

11. Codecademy. What is a Relational Database Management System?: веб-сайт. URL: https://www.codecademy.com/articles/what-is-rdbms-sql (дата звернення: 15.11.2019)

12. Microsoft Docs. Non-relational data and NoSQL – Azure Architecture Center: веб-сайт. URL: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data (дата звернення: 15.11.2019)

13. Michael Madison, Mark Barnhill, Cassie Napier, Joy Godin. NoSQL Database Technologies: Journal of International Technology and Information Management. International Information Management Association – 2015p – C.3

14. Ian. What does ACID mean in Database Systems?: веб-сайт. URL: https://database.guide/what-is-acid-in-databases/ (дата звернення: 15.11.2019)

15. Cody Arsenault. The Pros And Cons of 8 Popular Databases: веб-сайт. URL: https://www.keycdn.com/blog/popular-databases (дата звернення: 15.11.2019)

16. Leah Betty. The Pros And Cons of MySQL: веб-сайт. URL: https://www.smartfile.com/blog/the-pros-and-cons-of-mysql/ (дата звернення: 15.11.2019)

17. What is PostgreSQL? Features, Advantages and Disadvantages: веб-сайт. URL: https://www.educba.com/what-is-postgresql/ (дата звернення: 15.11.2019)

18. Sunanda Chowdhary, Vipin Sharma, A. B. Patki. Fuzzy Logic System for Querying a Database. Conference: INDIACom. New Delhi – 2010, p. 269-272.

19. Зегжда Д.П., Івашко А.М. Основи безпеки інформаційних систем. М.: Гаряча лінія –Телеком, 2000.134с.

20. Рижова В.А. Проектування і дослідження комплексних систем безпеки. СПб: НДУ ІТМО.2012.

21. Юр'єв Н.Н., Васяєва Т.А., Бельков С.Д. Система контролю і управління доступом. Інформатика, керуючі системи, математичне і комп'ютерне моделювання. Київ: 2017. Вип.7. С. 601 –604.

22. Novak V., Perfilieva I., Mockor J. Mathematical principles of fuzzy logic M: Kluwer Academic Publishers, 1999.P. 15.

23. Zadeh L. Real-Life Applications of Fuzzy Logic. Fuzzy logic now and then. M: Hindawi, 2013. P. 125.

24. Cintula P. Fuzzy Logics as the Logics of Chains. Fuzzy Sets and Systems M: Libor, 2006.P. 606.

25. Godo L. Fuzzy Sets and Systems. Monoidal T-Norm Based Logic: Towards a Logic for Left-Continuous T-Norms. M: Waweland, 2001.P. 25.

26. Zimmerman H. Fuzzy set theory and its applications. Fuzzy logic introduction. M: Kluwer, 1991. P. 315.

27. MathWorks (What Is Fuzzy Logic?): веб-сайт. URL:https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html (дата звернення: 15.11.2019).

28. Atlam H.F., Walters R.J., Wills G.B. et al. Fuzzy Logic with Expert Judgment to Implement an Adaptive Risk-Based Access Control Model for IoT. Mobile Networks and Application. 2019. DOI: https://doi.org/10.1007/s11036-019-01214-w

29. José Galindo. Introduction and Trends to Fuzzy Logic and Fuzzy Databases. 2008. DOI: https://doi.org/10.4018/978-1-59904-853-6.ch001

30. Aguilera A, Mata-Toledo RA. Fuzzy databases. In: AccessScience. McGraw-Hill Education. 2013. DOI: https://doi.org/10.1036/1097-8542.YB130132.

31. Mamdani E. Application of fuzzy algorithms for the control of a simple dynamic plant. Proc. IEEE 121, 1974. P. 1585–1588.

32. Мирончук Ю., Купріненко О. Побудова функцій належності нечітких множин, які відповідають кількісним експертним оцінкам фізичних величин. Системи обробки інформації.К.:2017.207с.

33. Блюмин С., Шуйкова И., Сараев П. Нечеткая логика: алгебраические основы и приложения: Монография. К.: ЛЭГИ, 2002. 113 с.

34. Cordon O., Herrera F. General study on genetic fuzzy systems.Genetic Algorithms in computer science.M.:Tante, 1995.P. 33.

35. Леоненков А. Нечеткое моделирование в MATLAB и fuzzyTECH.СПб.: БХВ-Петербург, 2005. 736 с.

36. Segismundo S. Izquierdoa, Luis R. Izquierdob. Mamdani Fuzzy Systems for Modelling and Simulation: A Critical Assessment. 2016. DOI: https://doi.org/10.18564/jasss.3660.

37. Леоненков, А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / А.В. Леоненков. – СПб.: БХВ-Петербург, 2005. – 736 с.

38. Алгоритмы нечёткого вывода: алгоритм Мамдани и алгоритм Сугэно. // В. Дьяконов, В. Круглов. Математические пакеты расширения MATLAB. Специальный справочник. — Санкт-Петербург: Питер, 2001 — С. 307–309

39. Штовба С. Д. Проектирование нечетких систем средствами MATLAB / С. Штовба. — М: Горячая линия-Телеком, 2007. — 288 с.

40. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / А. Леоненков. — СПб: БХВ-Петербург, 2003. — 736 с.

41. Лозинський А., Демків Л. Дослідження впливу вигляду функції належності на динамічні показники системи при багатокритеріальній оптимізації зі змінними ваговими коефіцієнтами. Електротехнічні та комп'ютерні системи. К.: 2012. Вип.5. С. 137–144

42. MathWorks(SimulationandModel-BasedDesign):веб-сайт. URL: https://www.mathworks.com/products/ simulink.html (дата звернення: 06.09.2020).

43. What is matlab and why is it used?: веб-сайт. URL: https://www.freelancinggig.com/blog/2019/01/12/what-is-matlab-and-why-is-it-used/ (дата звернення: 06.09.2020)

44. Штовба С. Д. Введення в теорію нечітких множин и нечітку логіку. Вінниця: Видавництво вінницького державного технічного університету, 2001. 198 с.

45. Build Fuzzy Systems Using Fuzzy Logic Designer: веб-сайт. URL: https://www.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html (дата звернення: 06.09.2020)

46. Ротштейн А., Штовба С. Идентификация нелинейной зависимости нечеткой базой знаний с нечеткой обучающей выборкой. Кибернетика и системный анализ. Х.:2006. Вип. 2. С. 17–24.

47. Штовба С.Д. Побудова функцій належності нечітких множин за кластеризацією експериментальних даних. Інформаційні технології та комп'ютерна інженерія. К.: 2006. Вип. 2. С. 92–95.

48. Open rule editor - Matlab ruleedit: веб-сайт. URL: https://www.mathworks.com/help/fuzzy/ruleedit.html (дата звернення: 10. 09.2020)

49. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы.М.: Горячая линия –Телеком, 2004. 452 с

50. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія"/ Під ред. О.М. Березького. Тернопіль:ЗУНУ,2020.32 с.

51. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія»/Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.