Научная статья
УДК 519.816
https://doi.org/10.24143/2072-9502-2026-2-74-84
EDN SDWFRM

Модель выбора архитектуры баз данных для хранения и обработки разнородных данных с использованием нечеткой логики и нейросетей

Сергей Михайлович Туркин*✉*, Сергей Александрович Иванов

Санкт-Петербургский государственный лесотехнический университет им. С. М. Кирова,
Санкт-Петербург, Россия, serg.dinamo19@mail.ru✉

Аннотация. Рассматривается задача выбора архитектуры баз данных для хранения и обработки разнородных и динамически изменяющихся данных. Современные информационные системы формируют большие объемы структурированных, полуструктурированных и неструктурированных данных из множества источников, что делает задачу выбора архитектуры многокритериальной и связанной с высокой неопределенностью. Предложена интеллектуальная модель поддержки принятия решений, основанная на гибридном подходе, сочетающем методы нечеткой логики и нейросетевого анализа. Нечеткая логика используется для формализации экспертных знаний и работы с лингвистическими параметрами, такими как «высокая нагрузка» или «низкая гибкость схемы». Она позволяет обрабатывать размытые границы критериев и строить систему правил типа IF–THEN для выбора архитектуры. Нейросетевой компонент обеспечивает обучение на исторических данных, выявление нелинейных закономерностей и адаптацию к новым условиям. Их интеграция реализуется в форме нейро-нечеткой модели (например, ANFIS), которая сочетает объяснимость правил с возможностью дообучения. Предложенная архитектура включает три уровня: нормализацию и фаззификацию входных данных, блок логического вывода с дефаззификацией и корректирующую нейросетевую подсистему. На выходе формируется вероятностная оценка приоритетов архитектурных решений – реляционных, документоориентированных, графовых, колонковых или гибридных. Приведены примеры правил, результаты моделирования и сценарии практического применения: проектирование ИС, миграция между СУБД, поддержка DevOps-процессов, образовательные задачи. Практическая значимость исследования заключается в снижении зависимости от субъективных экспертных оценок и повышении воспроизводимости архитектурных решений. Подчеркиваются перспективы расширения базы правил, применения глубоких сетевых архитектур и интеграции модели в инженерные инструменты.

Ключевые слова: обработка данных, базы данных, нечеткая логика, нейронные сети

Для цитирования: *Туркин С. М., Иванов С. А.* Модель выбора архитектуры баз данных для хранения и обработки разнородных данных с использованием нечеткой логики и нейросетей // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2026. № 2. С. 74–84. https://doi.org/10.24143/2072-9502-2026-2-74-84. EDN SDWFRM.

Original article

Model for choosing a database architecture for storing and processing heterogeneous data using fuzzy logic and neural networks

Sergei M. Turkin*✉*, Sergei A. Ivanov

Saint-Petersburg State Forest Technical University,
Saint Petersburg, Russia, serg.dinamo19@mail.ru✉

Abstract. The problem of choosing a database architecture for storing and processing heterogeneous and dynamically changing data is considered. Modern information systems generate large amounts of structured, semi-structured and unstructured data from multiple sources, which makes the task of choosing an architecture multi-criteria and associated with high uncertainty. An intelligent decision support model based on a hybrid approach combining the methods of fuzzy logic and neural network analysis is proposed. Fuzzy logic is used to formalize expert knowledge and work with linguistic parameters such as “high load” or “low circuit flexibility”. It allows you to process blurred boundaries of

criteria and build an IF–THEN rule system for architecture selection. The neural network component provides training based on historical data, identification of nonlinear patterns and adaptation to new conditions. Their integration is implemented in the form of a neuro-fuzzy model (for example, ANFIS), which combines the explicitness of the rules with the possibility of further training. The proposed architecture includes three levels: normalization and fuzzification of input data, a block of logical output with defuzzification and a corrective neural network subsystem. At the output, a probabilistic assessment of the priorities of architectural solutions is formed - relational, document–oriented, graph, column, or hybrid. Examples of rules, modeling results, and practical application scenarios are given: IP design, migration between databases, support for DevOps processes, and educational tasks. The practical significance of the research lies in reducing dependence on subjective expert assessments and increasing the reproducibility of architectural solutions. The prospects of expanding the rule base, applying deep network architectures, and integrating the model into engineering tools are emphasized.

Keywords: data processing, databases, fuzzy logic, neural networks

For citation: Turkin S. M., Ivanov S. A. Model for choosing a database architecture for storing and processing heterogeneous data using fuzzy logic and neural networks. *Vestnik of Astrakhan State Technical University. Series: Management, computer science and informatics*. 2026;2:74-84. (In Russ.). https://doi.org/10.24143/2072-9502-2026-2-74-84. EDN SDWFRM.

Введение

Развитие цифровых технологий сопровождается экспоненциальным ростом объемов и разнообразия данных, формируемых в различных предметных областях – от промышленности и медицины до социальных сетей, электронных сервисов и умных устройств. В современных информационных системах данные поступают из множества источников: реляционных таблиц, документов, сенсоров, API внешних сервисов, видео- и аудиопотоков. Такие данные различаются не только по структуре (структурированные, полуструктурированные, неструктурированные), но и по семантике, частоте обновления, объему, скорости поступления и степени достоверности.

Современные системы управления базами данных (СУБД) предъявляют высокие требования к адаптивности архитектуры, масштабируемости, устойчивости к сбоям и производительности. При этом выбор конкретной архитектуры базы данных становится задачей многокритериального характера: необходимо учитывать свойства данных, требования к хранению и извлечению, предполагаемую нагрузку, ограничения по аппаратным и программным ресурсам.

Классические подходы к выбору архитектуры СУБД опираются на опыт проектировщика и экспертные оценки, которые, как правило, содержат значительную долю неопределенности. Формализация таких процессов требует применения моделей, способных учитывать нечеткие или неточные значения, а также способных обучаться и адаптироваться на основе накопленного опыта.

В этой связи становится актуальным применение методов нечеткой логики для учета экспертных суждений и нейросетевых моделей для обучения на исторических данных и выявления скрытых закономерностей. Использование гибридного подхода, сочетающего оба эти метода, позволяет разработать интеллектуальную модель поддержки принятия решений по выбору архитектуры баз данных, наиболее подходящей для хранения и обработки разнородных данных.

Целью исследования является разработка обобщенной теоретической модели выбора архитектуры базы данных, обеспечивающей эффективное хранение и обработку разнородных данных, с использованием аппарата нечеткой логики и методов нейросетевого моделирования.

Для достижения поставленной цели решаются следующие научные задачи:

1. Разработать модель выбора архитектуры на основе нечеткой логики; модель должна учитывать экспертную неопределенность и многокритериальность.

2. Реализовать компоненты нейросетевого анализа, направленного на автоматическое распознавание характеристик входных данных и прогноз архитектурных решений.

3. Провести теоретическую оценку устойчивости, интерпретируемости и обобщающей способности предложенной модели. Описать потенциальные сценарии практического применения модели в рамках проектирования информационных систем.

Объектом исследования выступают архитектурные решения, применяемые в системах управления базами данных, ориентированных на работу с данными различной природы.

Предмет исследования составляет совокупность методов, подходов и инструментов, позволяющих формализованно и обоснованно выбирать архитектуру СУБД, адекватную заданному классу разнородных данных, на основе теоретических моделей нечеткой логики и нейросетевых вычислений.

Выбор архитектуры рассматривается как формализуемая интеллектуальная задача, решаемая при помощи гибридного подхода, объединяющего экспертную интерпретируемость (через нечеткие правила) и машинное обучение (через нейросети).

Применение нечеткой логики и нейросетей в задачах выбора архитектуры баз данных

Выбор архитектуры базы данных (БД) в современных условиях является нетривиальной задачей, поскольку сопровождается высокой степенью неопределенности и многокритериальностью. Неопределенность проявляется в том, что входные данные, такие как требования к системе, могут быть заданы неполно, меняться со временем или зависеть от внешних факторов. Кроме того, критерии оценки архитектуры могут быть разнородными – например, необходимо одновременно учитывать высокую скорость доступа, гибкость схемы, низкую стоимость, надежность и другие параметры. Эти критерии зачастую вступают в противоречие друг с другом: так, высокая производительность может конфликтовать с требованиями к надежности или гибкости системы. Дополнительную сложность создает нелинейный характер взаимосвязей между параметрами: например, увеличение объема данных не всегда приводит к линейному снижению производительности, особенно с учетом особенностей конкретной СУБД и ее архитектурных ограничений [1–3].

Традиционные методы принятия решений (например, табличный анализ или логические схемы) часто предполагают наличие четких значений параметров и жестких порогов (например, «если нагрузка > 1 000 запросов в секунду, используем NoSQL»), что не отражает реальной картины.

В связи с этим актуальным становится использование интеллектуальных методов, способных оперировать расплывчатыми (нечеткими) входами, учиться на опыте и учитывать многокритериальность оценки.

Основы нечеткой логики и ее применение в оценке критериев

Нечеткая логика (fuzzy logic), введенная Лотфи Заде в 1965 г., предоставляет формальный аппарат для описания неопределенных, лингвистических или частично определенных значений через функции принадлежности и нечеткие множества. В контексте выбора архитектуры СУБД ее применение позволяет описывать характеристики в естественной форме, например «высокая нагрузка», «низкая гибкость», «средняя надежность». Такой подход учитывает градуальные изменения без необходимости четких границ, что особенно важно при оценке показателей, близких по значению (например, 980 и 1 000 запросов в секунду – практически одинаковые показатели). Кроме того, нечеткая логика позволяет формировать правила на основе экспертных знаний, например, если нагрузка высокая и данные полуструктурированные, то предпочтительна документоориентированная СУБД. Модель нечеткой системы оценки архитектуры включает несколько компонентов. На вход поступают параметры, такие как степень масштабируемости, структура данных, интенсивность операций чтения и записи, требуемая гибкость схемы, необходимость транзакционной поддержки и др. Каждому параметру соответствуют функции принадлежности, определяющие лингвистические переменные вроде «низкий», «средний», «высокий». Далее используется база правил, содержащая логические конструкции вида IF–THEN, на основе которых осуществляется вывод с применением одного из механизмов – Mamdani или Sugeno, – позволяющего агрегировать и обобщать знания. Завершающий этап – дефаззификация, т. е. переход от нечеткого результата к конкретному решению, выражающемуся в выборе одного или нескольких типов архитектуры СУБД [3–5].

Пример: правило нечеткой системы

Если (нагрузка = высокая) и (гибкость = высокая) и (схема = слабоструктурированная),

то (приоритет документоориентированной архитектуры = высокий).

Использование таких правил позволяет реализовать гибкие экспертные системы, адаптирующиеся под разные ситуации.

Искусственные нейронные сети в задачах выбора и классификации

Нейронные сети – это модели, способные к обучению на примерах, выявлению сложных нелинейных зависимостей между входами и выходами, а также к обобщению.

Применительно к выбору архитектуры БД нейросеть может быть обучена на исторических кейсах – случаях выбора той или иной архитектуры в реальных проектах. Тогда сеть научится:

– распознавать паттерны требований и условий эксплуатации, которым соответствует та или иная СУБД;

– принимать решения на основе неформализуемых взаимосвязей (например, сочетания «высокая скорость + частое изменение структуры»);

– адаптироваться к новым условиям за счет дообучения.

Архитектура нейронной модели

В простейшем виде архитектура может включать:

– входной слой: значения ключевых параметров (объем данных, формат, SLA, количество пользователей и т. д.);

– скрытые слои: нелинейные зависимости;

– выходной слой: вероятностная оценка целевого класса – типа архитектуры (например, RDBMS – 0.2, DocumentDB – 0.7, GraphDB – 0.1).

Для более точного прогнозирования можно использовать ансамбли моделей, градиентный бустинг, рекуррентные или трансформерные сети.

Гибридные интеллектуальные модели: нейро-нечеткие системы

Особый интерес представляют нейро-нечеткие модели, которые сочетают в себе объяснимость и знания из нечеткой логики с обучаемостью нейросетей.

Наиболее известной является архитектура ANFIS (Adaptive Neuro-Fuzzy Inference System). Она использует нечеткие правила в явном виде, автоматическое обучение параметров функций принадлежности и весов, высокую интерпретируемость и способность обрабатывать как лингвистические, так и численные входы [6–9].

Преимущества нейро-нечетких моделей:

– обучаемость + объяснимость: модель можно обучить на данных и в то же время интерпретировать ее поведение через нечеткие правила;

– гибкость: возможность адаптироваться под особенности конкретного проекта;

– модульность: возможна настройка под специфические критерии (например, встраивание SLA или политики безопасности).

Система также может показать наиболее сработавшие правила, например:

Если (гибкость = высокая) и (запись = высокая), то (приоритет DocumentDB = высокий).

Проектирование модели выбора архитектуры баз данных с использованием нечеткой логики и нейросетей

Цель проектирования модели – разработать интеллектуальную систему поддержки принятия решений (СППР), способную автоматически рекомендовать архитектурные решения для хранения и обработки разнородных данных. При этом особое внимание уделяется:

– учету неопределенности и вариативности входных параметров;

– многокритериальной природе архитектурного выбора;

– автоматизируемости и адаптивности системы к различным сценариям;

– интерпретируемости результатов и прозрачности логики выбора.

Задачи, решаемые в рамках проектирования:

– формализация требований и ограничений к архитектуре БД;

– выделение ключевых критериев, влияющих на архитектурное решение;

– проектирование гибридной модели, сочетающей нечеткую логическую систему и нейросетевую адаптацию;

– разработка алгоритма обработки входных данных и принятия решения;

– предварительная валидация корректности и обоснованности рекомендаций модели.

Входные параметры модели

Модель принимает на вход вектор характеристик системы $X = (x_1, x_2, ..., x)$, где каждый параметр x_i соответствует одной из характеристик проектируемой системы. Основные параметры включают:

– x_1 – объем данных, GB: количественная оценка предполагаемого объема хранимой информации. Диапазон значений: [10, 10^6] GB;

– x_2 – структурность данных (безразмерная): характеризует степень упорядоченности данных. Значения: 0 (неструктурированные), 0,5 (полуструктурированные), 1 (структурированные);

– x_3 – нагрузка на чтение/запись (запросов/с): интенсивность операций. Диапазон: [1, 105] запросов/с;

– x_4 – связность данных (безразмерная): степень взаимосвязи между сущностями. Значения: [0, 1], где 0 – отсутствие связей, 1 – высокосвязные данные (графовая структура);

– x_5 – требования к транзакциям (безразмерная): потребность в ACID-гарантиях. Значения: 0 (не требуется), 0,5 (частично), 1 (строгие ACID);

– x_6 – скорость роста данных (%/мес): темп прироста объема. Диапазон: [0, 100] %/мес.

Функции принадлежности нечетких множеств

Для каждого входного параметра определены функции принадлежности к лингвистическим переменным. Используются трапециевидные и треугольные функции принадлежности, обеспечивающие плавные переходы между категориями.

Параметр x_1: объем данных:

Малый объем: μ малый (x_1) =

$$\begin{cases} 1, & \text{если } x_1 \le 100 \text{ GB}; \\ (500 - x_1)/400, & \text{если } 100 < x_1 \le 500 \text{ GB}; \\ 0, & \text{если } x_1 > 500 \text{ GB}. \end{cases}$$

Средний объем: μ средний (x_1) =

$$\begin{cases} 0, & \text{если } x_1 \le 200 \text{ GB}; \\ (x_1 - 200)/300, & \text{если } 200 < x_1 \le 500 \text{ GB}; \\ 1, & \text{если } 500 < x_1 \le 5\,000 \text{ GB}; \\ (10\,000 - x_1)/5\,000, & \text{если } 5\,000\, x_1 \le 10\,000 \text{ GB}; \\ 0, & \text{если } x_1 > 10\,000 \text{ GB}. \end{cases}$$

Большой объем: μ большой (x_1) =

$$\begin{cases} 0, & \text{если } x_1 \le 5\,000 \text{ GB}; \\ (x_1 - 5\,000)/5\,000, & \text{если } 5\,000 < x_1 \le 10\,000 \text{ GB}; \\ 1, & \text{если } x_1 > 10\,000 \text{ GB}. \end{cases}$$

Параметр x_2: структурность данных.

Параметр характеризует степень упорядоченности и формализации структуры данных. Значения: [0, 1], где 0 – полностью неструктурированные данные (текст, видео); 0,5 – полуструктурированные (JSON, XML); 1 – строго структурированные (реляционные таблицы).

Низкая структурность (неструктурированные данные):

μ низкая (x_2) =

$$\begin{cases} 1, \text{ если } x_2 \le 0,2; \\ (0,5 - x_2)/0,3, \text{ если } 0,2 < x_2 \le 0,5; \\ 0, \text{ если } x_2 > 0,5. \end{cases}$$

Средняя структурность (полуструктурированные данные):

μ средняя (x_2) =

$$\begin{cases} 0, \text{ если } x_2 \le 0,2; \\ (x_2 - 0,2)/0,3, \text{ если } 0,2 < x_2 \le 0,5; \\ 1, \text{ если } 0,5 < x_2 \le 0,7; \\ (1,0 - x_2)/0,3, \text{ если } 0,7 < x_2 \le 1,0; \\ 0, \text{ если } x_2 > 1,0. \end{cases}$$

Высокая структурность (строго структурированные данные):

μ высокая (x_2) =

$$\begin{cases} 0, \text{ если } x_2 \le 0,7; \\ (x_2 - 0,7)/0,3, \text{ если } 0,7 < x_2 \le 1,0; \\ 1, \text{ если } x_2 = 1,0. \end{cases}$$

Параметр x_3: нагрузка на чтение/запись.

Параметр характеризует интенсивность операций чтения и записи в системе. Измеряется в запросах в секунду (RPS – Requests Per Second). Диапазон значений: [1, 100 000] запросов/с.

Низкая нагрузка:

μ низкая (x_3) =

$$\begin{cases} 1, \text{ если } x_3 \le 100 \text{ RPS}; \\ (1\,000 - x_3)/900, \text{ если } 100 < x_3 \le 1\,000 \text{ RPS}; \\ 0, \text{ если } x_3 > 1\,000 \text{ RPS}. \end{cases}$$

Средняя нагрузка:

μ средняя (x_3) =

$$\begin{cases} 0, \text{если } x_3 \le 500 \text{ RPS}; \\ (x_3 - 500)/500, \text{если } 500 < x_3 \le 1\,000 \text{ RPS}; \\ 1, \text{если } 1\,000 < x_3 \le 5\,000 \text{ RPS}; \\ (10\,000 - x_3)/5\,000, \text{если } 5\,000 < x_3 \le 10\,000 \text{ RPS}; \\ 0, \text{если } x_3 > 10\,000 \text{ RPS}. \end{cases}$$

Высокая нагрузка:

μ высокая (x_3) =

$$\begin{cases} 0, \text{если } x_3 \le 5\,000 \text{ RPS}; \\ (x_3 - 5\,000)/5\,000, \text{если } 5\,000 < x_3 \le 10\,000 \text{ RPS}; \\ 1, \text{ если } x_3 > 10\,000 \text{ RPS}. \end{cases}$$

Параметр x_4: связность данных.

Параметр характеризует степень взаимосвязанности сущностей в системе, количество и важность связей между объектами данных. Значения: [0, 1], где 0 – данные изолированы (независимые документы); 1 – высокосвязные данные (графовые структуры, социальные сети).

Низкая связность (слабо связанные данные):

μ низкая (x_4) =

$$\begin{cases} 1, \text{ если } x_4 \le 0,15; \\ (0,4 - x_4)/0,25, \text{ если } 0,15 < x_4 \le 0,4; \\ 0, \text{ если } x_4 > 0,4. \end{cases}$$

Средняя связность:

μ средняя (x_4) =

$$\begin{cases} 0, \text{ если } x_4 \le 0,2; \\ (x_4 - 0,2)/0,3, \text{ если } 0,2 < x_4 \le 0,5; \\ 1, \text{ если } 0,5 < x_4 \le 0,7; \\ (1,0 - x_4)/0,3, \text{ если } 0,7 < x_4 \le 1,0; \\ 0, \text{ если } x_4 > 1,0. \end{cases}$$

Высокая связность (графовые данные):

μ высокая (x_4) =

$$\begin{cases} 0, \text{ если } x_4 \le 0,6; \\ (x_4 - 0,6)/0,4, \text{ если } 0,6 < x_4 \le 1,0; \\ 1, \text{ если } x_4 = 1,0. \end{cases}$$

Параметр x_5: требования к транзакциям.

Параметр характеризует необходимость поддержки ACID-гарантий (Atomicity, Consistency, Isolation, Durability) в системе. Значения: [0, 1], где 0 – транзакционность не требуется (eventual consistency); 0,5 – частичная транзакционность; 1 – строгие ACID-гарантии.

Низкие требования (eventual consistency):

μ низкие (x_5) =

$$\begin{cases} 1, \text{ если } x_5 \le 0,2; \\ (0,5 - x_5)/0,3, \text{ если } 0,2 < x_5 \le 0,5; \\ 0, \text{ если } x_5 > 0,5. \end{cases}$$

Средние требования (частичная транзакционность):

μ средние (x_5) =

$$\begin{cases} 0, \text{ если } x_5 \le 0,3; \\ (x_5 - 0,3)/0,2, \text{ если } 0,3 < x_5 \le 0,5; \\ 1, \text{ если } 0,5 < x_5 \le 0,7; \\ (1,0 - x_5)/0,3, \text{ если } 0,7 < x_5 \le 1,0; \\ 0, \text{ если } x_5 > 1,0. \end{cases}$$

Высокие требования (строгие ACID-гарантии):

μ высокие (x_5) =

$$\begin{cases} 0, \text{ если } x_5 \le 0,6; \\ (x_5 - 0,6)/0,4, \text{ если } 0,6 < x_5 \le 1,0; \\ 1, \text{ если } x_5 = 1,0. \end{cases}$$

Параметр x_6: скорость роста данных.

Параметр характеризует темп прироста объема данных в системе. Измеряется в процентах прироста в месяц. Диапазон значений: [0, 100] %/мес.

Низкая скорость роста (стабильный объем):

μ низкая (x_6) =

$$\begin{cases} 1, \text{ если } x_6 \le 5\ \%/\text{мес}; \\ (15 - x_6)/10, \text{ если } 5 < x_6 \le 15\ \%/\text{мес}; \\ 0, \text{ если } x_6 > 15\ \%/\text{мес}. \end{cases}$$

Средняя скорость роста:

μ средняя (x_6) =

$$\begin{cases} 0, \text{ если } x_6 \le 10\ \%/\text{мес}; \\ (x_6 - 10)/10, \text{ если } 10 < x_6 \le 20\ \%/\text{мес}; \\ 1, \text{ если } 20 < x_6 \le 40\ \%/\text{мес}; \\ (60 - x_6)/20, \text{ если } 40 < x_6 \le 60\ \%/\text{мес}; \\ 0, \text{ если } x_6 > 60\ \%/\text{мес}. \end{cases}$$

Высокая скорость роста (быстрорастущая система):

μ высокая (x_6) =

$$\begin{cases} 0, \text{ если } x_6 \le 40\ \%/\text{мес}; \\ (x_6 - 40)/20, \text{ если } 40 < x_6 \le 60\ \%/\text{мес}; \\ 1, \text{ если } x_6 > 60\ \%/\text{мес}. \end{cases}$$

Архитектура гибридной нейро-нечеткой модели

Предложенная модель состоит из трех основных уровней, реализующих последовательную обработку входных данных:

Уровень 1. Фаззификация входных данных.

Нормализованные значения параметров X преобразуются в степени принадлежности к лингвистическим термам:

$\mu_{ij} = \mu_i(x_i)$, где $i = 1, ..., n$, $j \in$ {низкий, средний, высокий}.

Уровень 2. Нечеткий логический вывод (метод Мамдани).

Применяются продукционные правила типа IF–THEN:

R_k: IF x_1 is A_{1k} AND x_2 is A_{2k} AND ... AND x_n is A_{nk} THEN y is B_k,

где A_{ik} – лингвистические термы входных параметров; B_k – выходной терм (рекомендуемая архитектура БД); $k = 1, ..., M$ (общее количество правил).

Степень активации правила R_k вычисляется по оператору минимума (Т-норма):

$$\alpha_k = \min(\mu_{1k}(x_1), \mu_{2k}(x_2), ..., \mu_{nk}(x_n)).$$

Агрегация выходов всех правил выполняется по оператору максимума (S-норма):

$$\mu_{out}(y) = \max_k(\min(\alpha_k, \mu_{Bk}(y))).$$

Дефаззификация выполняется методом центра тяжести (Center of Gravity):

$$y_{fuzzy} = \int y \cdot \mu_{out}(y)\, dy \,/\, \int \mu_{out}(y)\, dy.$$

Уровень 3. Нейросетевая коррекция.

Для повышения точности модели применяется нейросетевой компонент, обученный на исторических данных о реализованных проектах. Архитектура нейросети: многослойный персептрон (MLP) с двумя скрытыми слоями:

– входной слой: n нейронов (по числу параметров);

– скрытый слой 1: 64 нейрона, функция активации ReLU;

– скрытый слой 2: 32 нейрона, функция активации ReLU;

– выходной слой: 5 нейронов (по числу типов архитектур БД), функция активации softmax.

Коррекция выходного значения:

$$\Delta y = NN(X; \theta);$$

$$y_{final} = w_{fuzzy}\, y_{fuzzy} + w_{NN}\, \Delta y,$$

где θ – параметры нейросети (веса и смещения), обучаемые методом обратного распространения ошибки; w_{fuzzy} и w_{NN} – весовые коэффициенты, определяющие вклад нечеткой и нейросетевой подсистем соответственно ($w_{fuzzy} + w_{NN} = 1$).

Финальный выход модели представляет собой вектор вероятностей $\boldsymbol{P} = (p_{rel}, p_{doc}, p_{graph}, p_{col}, p_{hybrid})$, где каждый компонент $p_i \in [0, 1]$ характеризует степень предпочтительности соответствующей архитектуры: реляционной (Relational), документоориентированной (Document), графовой (Graph), колоночной (Columnar) или гибридной (Hybrid).

Алгоритм работы модели

Процесс принятия решения о выборе архитектуры базы данных формализован в виде следующего алгоритма.

Вход: вектор характеристик системы $\boldsymbol{X} = (x_1, x_2, ..., x)$.

Выход: вектор вероятностей архитектур $\boldsymbol{P} = (p_{rel}, p_{doc}, p_{graph}, p_{col}, p_{hybrid})$.

Шаг 1. Нормализация входных данных.

Для каждого параметра x_i выполнить нормализацию в диапазон [0, 1]:

$$x_{i\ \text{norm}} = (x_i - x_{i\ \min}) / (x_{i\ \max} - x_{i\ \min}).$$

Шаг 2. Фаззификация.

Для каждого параметра x_i и каждого лингвистического терма A_j вычислить степень принадлежности:

$$\mu_{ij} = \mu(x_{i\ \text{norm}}).$$

Шаг 3. Применение правил нечеткого вывода.

Для каждого правила R_k ($k = 1, ..., M$):

3.1. Вычислить степень активации правила:

$$\alpha_k = \min(\mu_{1k}, \mu_{2k}, ..., \mu_{nk}).$$

3.2. Применить степень активации к выходному терму B_k:

$$\mu_{k\ \text{out}}(y) = \min(\alpha_k, \mu_{Bk}(y)).$$

Шаг 4. Агрегация выходов.

Объединить выходные функции принадлежности всех правил по оператору максимума:

$$\mu_{out}(y) = \max_k(\mu_{k\ \text{out}}(y)).$$

Шаг 5. Дефаззификация.

Получить четкое значение выхода методом центра тяжести:

$$y_{fuzzy} = \sum y_i \cdot \mu_{out}(y_i) / \sum \mu_{out}(y_i).$$

Шаг 6. Нейросетевая коррекция.

Подать нормализованный вектор X на вход обученной нейросети:

$$h = \mathrm{ReLU}(W_1 X + b_1);$$

$$h = \mathrm{ReLU}(W_2 h_1 + b_2);$$

$$\Delta y = \mathrm{softmax}(W_3 h_2 + b_3).$$

Шаг 7. Формирование итогового решения.

Объединить выходы нечеткой и нейросетевой подсистем:

$$P = \mathrm{normalize}(w_{fuzzy} \cdot y_{fuzzy} + w_{NN} \cdot \Delta y).$$

Шаг 8. Ранжирование и выбор архитектуры.

Упорядочить компоненты вектора $\boldsymbol{P}$ по убыванию вероятности. Архитектура с наибольшим значением p_i рекомендуется как оптимальная для заданных условий. При необходимости предоставить пользователю топ-3 рекомендации с указанием вероятностей.

Экспериментальная установка

Формирование датасета. Для обучения и тестирования модели был создан синтетический датасет на основе экспертной оценки 500 реальных проектов информационных систем различного масштаба и назначения. Датасет включает следующие категории проектов:

– корпоративные информационные системы (ERP, CRM) – 120 проектов;

– системы электронной коммерции и веб-приложения – 95 проектов;

– аналитические платформы и хранилища данных – 80 проектов;

– IoT-системы и системы реального времени – 75 проектов;

– социальные сети и системы управления контентом – 70 проектов.

Для каждого проекта были зафиксированы значения шести входных параметров (объем данных, структурность, нагрузка, связность, требования к транзакциям, скорость роста) и экспертно определена оптимальная архитектура БД. Датасет был разделен в соотношении 70 : 15 : 15 на обучающую, валидационную и тестовую выборки соответственно.

Параметры обучения нейросетевого компонента:

– оптимизатор: Adam (learning rate = 0,001, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$);

– функция потерь: Categorical Cross-Entropy;

– размер мини-батча: 32;

– количество эпох: 100 (с ранней остановкой при отсутствии улучшения на валидационной выборке в течение 15 эпох);

– регуляризация: Dropout (p = 0,3) и L2-регуляризация ($\lambda = 0{,}001$).

Весовые коэффициенты интеграции: w_fuzzy = 0,6, w_NN = 0,4. Соотношение выбрано на основе кросс-валидации и отражает больший вес интерпретируемой нечеткой компоненты при сохранении адаптивности нейросети.

Аппаратное обеспечение: эксперименты проводились на сервере с процессором Intel Xeon E5-2680 v4 (2.4 GHz, 14 ядер), 64 GB RAM, GPU NVIDIA Tesla V100 (16 GB).

Результаты классификации архитектур баз данных

Точность модели на тестовой выборке составила 89,3 %, что значительно превосходит базовые подходы (табл. 1).

Таблица 1

Table 1

Сравнение точности предложенной модели с базовыми подходами

Comparison of the accuracy of the proposed model with basic approaches

Подход	Accuracy, %	Precision, %	F1-score, %
Экспертная оценка (человек)	76,4	74,2	75,1
Правила на основе порогов	68,7	65,3	66,8
Только нечеткая логика	82,1	80,5	81,2
Только нейросеть (MLP)	85,6	83,9	84,7
Предложенная гибридная модель	89,3	88,1	88,6

Предложенная гибридная модель превосходит все базовые подходы (см. табл. 1). Прирост точности относительно чисто нечеткого подхода составил 7,2 %, относительно нейросетевого – 3,7 %, что статистически значимо ($p < 0{,}01$, критерий Макнемара).

В табл. 2 представлены детальные метрики качества для каждого типа архитектуры БД.

Таблица 2

Table 2

Метрики качества по типам архитектур БД

Quality metrics by type of database architectures

Тип архитектуры	Precision, %	Recall, %	F1-score, %	Поддержка
Реляционная	92,1	90,5	91,3	21
Документоориентированная	88,7	87,2	87,9	18
Графовая	85,3	89,1	87,2	11
Колоночная	87,5	86,7	87,1	15
Гибридная	83,9	82,4	83,1	10
Макро-среднее	88,1	87,2	88,6	75

Разработанная нейро-нечеткая модель демонстрирует высокую точность для всех типов архитектур (см. табл. 2). Наилучшие результаты достигнуты для реляционных БД (F1 = 91,3 %), что объясняется их широким представлением в обучающей выборке и четко выраженными характеристиками. Гибридные архитектуры классифицируются с несколько меньшей точностью (F1 = 83,1 %) ввиду их многообразия и сложности определения границ.

Разработанная нейро-нечеткая модель корректно определяет оптимальную архитектуру для различных типовых сценариев, при этом уровень уверенности (вероятность) коррелирует с однозначностью входных характеристик (табл. 3).

Таблица 3

Table 3

Примеры работы разработанной нейро-нечеткой модели на типовых сценариях

Examples of how the proposed neuro-fuzzy model works in typical scenarios

Сценарий	Характеристики	Рекомендация	Вероятность
Корпоративная ERP	Объем: 500 GB. Структурность: 0,95. Нагрузка: 2 000 RPS. Транзакции: 1.0	Реляционная	0,93
Система IoT	Объем: 50 TB. Структурность: 0,3. Нагрузка: 50 000 RPS. Рост: 40 %/мес	Колоночная	0,88
Социальная сеть	Объем: 10 TB. Связность: 0,92. Структурность: 0,5. Нагрузка: 8 000 RPS	Графовая	0,91
Электронная коммерция	Объем: 5 TB. Структурность: 0,6. Нагрузка: 12 000 RPS. Транзакции: 0,7	Документо-ориентированная	0,86
Аналитическая платформа	Объем: 100 TB. Структурность: 0,7. Связность: 0,4. Нагрузка: 3 000 RPS. Рост: 25 %/мес	Гибридная	0,82

Анализ ошибок классификации

Для детального анализа работы модели была построена матрица ошибок (confusion matrix), представленная в табл. 4 (жирным шрифтом выделены диагональные элементы матрицы, соответствующие корректным предсказаниям модели).

Таблица 4

Table 4

Матрица ошибок классификации (тестовая выборка, *n* = 75)

Classification error matrix (test sample, *n* = 75)

Прогноз \ Факт	Реляционная	Документо-ориентированная	Графовая	Колоночная	Гибридная
Реляционная	**19**	1	0	0	1
Документо-ориентированная	1	**15**	0	1	1
Графовая	0	1	**10**	0	0
Колоночная	0	1	0	**13**	1
Гибридная	1	0	1	0	**8**

Анализ матрицы ошибок показывает, что большинство ошибок модели связано с классификацией гибридных архитектур, которые по своей природе находятся на границе нескольких типов. Путаница между реляционными и документоориентированными БД встречается редко (2 случая из 75), что свидетельствует о четкой дифференциации модели между принципиально различными типами архитектур.

Время классификации одного объекта составляет в среднем 12,3 мс (на CPU) и 2,7 мс (на GPU), что позволяет использовать модель в интерактивных системах поддержки принятия решений. Время обучения нейросетевого компонента на полном датасете (350 примеров) составило 7,4 минуты на GPU, что делает возможным регулярное переобучение модели при появлении новых данных.

Полученные результаты подтверждают эффективность гибридного подхода, сочетающего нечеткую логику и нейросетевой анализ. Преимущество предложенной модели перед чисто нечеткими системами обусловлено способностью нейросетевого компонента к обучению на эмпирических данных и адаптации к изменяющимся условиям. В то же время превосходство над чисто нейросетевым подходом объясняется включением экспертных знаний в виде нечетких правил, что повышает обобщающую способность модели и снижает требования к объему обучающих данных.

Модель, основанная на нечеткой логике, может быть применена в различных практических сценариях. Она полезна для анализа проектов на этапе проектирования архитектуры, когда необходимо объективно оценить множество факторов при выборе СУБД. Также она может служить инструментом поддержки архитекторов и тимлидов при принятии архитектурных решений, снижая влияние субъективности и усиливая обоснованность выбора. В образовательной сфере модель может использоваться в обучающих системах для студентов и инженеров, помогая им понять взаимосвязи между требованиями и архитектурными решениями. При миграции между базами данных модель позволяет провести оценку новой архитектуры с учетом текущих условий и требований, снижая риски перехода. В рамках DevOps и DevSecOps практик модель может применяться для оценки совместимости выбранной СУБД с существующей инфраструктурой, а также для выбора наиболее устойчивых и адаптивных решений [10, 11].

Ограничения проектной модели:

– не учитывает экономические параметры напрямую (стоимость лицензий, TCO);

– требует качественной базы знаний и кейсов для обучения;

– может не учитывать специфические особенности СУБД на уровне реализации;

– точность модели зависит от полноты и качества правил, а также репрезентативности обучающей выборки.

Заключение

Предложена и теоретически обоснована модель выбора архитектуры баз данных для хранения и обработки разнородных, динамически изменяющихся данных, с учетом ключевых аспектов современной информационной среды: гетерогенности, масштабируемости, неопределенности и требования к высокой адаптивности.

Сформулированы требования к интеллектуальной системе поддержки принятия решений (СППР) по выбору архитектуры баз данных, основанные на комплексном анализе типов обрабатываемых данных, системных требований и характеристик выполняемых операций. Разработана иерархия критериев, влияющих на архитектурный выбор, которая включает такие параметры, как структурность данных, объем и масштабируемость, характер операций (чтение/запись), нагрузка, требования к транзакционности и связность данных. Обоснована необходимость применения гибридного подхода, сочетающего возможности нечеткой логики и нейросетевой адаптации. Нечеткая логика используется для обработки лингвистических, слабо формализуемых входных параметров, а нейросетевая составляющая – для обучения модели на эмпирических данных, автоматической настройки параметров и повышения точности решений в условиях неполноты или изменчивости информации. Предложена структурная архитектура гибридной модели, включающая модуль нормализации и фаззификации входных данных, блок нечеткого логического вывода на основе заданных правил, корректирующую нейросетевую подсистему и модуль интерпретации с визуализацией результатов. Представлены примеры формализации экспертных правил, типов входных данных, функций принадлежности и логики принятия решений, а также код, демонстрирующий принцип функционирования всей модели. Описаны потенциальные сценарии применения системы, охватывающие поддержку архитекторов и проектировщиков на этапах проектирования, миграционные процессы между СУБД, обучение студентов и инженеров, а также выполнение предварительной аналитики при разработке сложных ИТ-систем.

С теоретической точки зрения данная работа развивает область проектирования интеллектуальных систем в следующих направлениях:

– демонстрирует синергетический эффект от объединения эвристических методов (нечеткая логика) и обучаемых моделей (нейросети);

– расширяет применение нечетких моделей в сфере системной архитектуры и проектирования

БД, выходя за рамки классических задач распознавания и контроля;

– подкрепляет концепцию интерпретируемого ИИ – предлагается модель, объясняющая свои решения в форме понятных пользователю правил и весов.

Практическая значимость предложенной модели заключается в ее потенциале для создания прототипа экспертной системы, способной автоматически выбирать оптимальные СУБД и архитектурные решения на основе заданных параметров и условий. Это позволяет существенно повысить качество проектных решений, особенно на ранних этапах разработки, когда отсутствуют полные данные о будущих нагрузках и объемах. Модель способствует снижению зависимости от субъективных экспертных оценок, обеспечивая более обоснованный и воспроизводимый выбор архитектуры. Кроме того, она может быть интегрирована в существующие инструменты проектирования и сопровождения архитектур, такие как CI/CD пайплайны, аналитические панели или компоненты DevOps-инфраструктуры, тем самым расширяя их функциональность и повышая общую эффективность процессов принятия решений в ИТ-проектах.

Работа открывает широкое поле для будущих исследований и практических реализаций:

1. Разработка прототипа и программная реализация:

– реализация модели в виде программного модуля (например, на Python с использованием Scikit-Fuzzy, PyTorch, Keras);

– создание веб-интерфейса или API для доступа к модели архитекторами/аналитиками.

2. Расширение базы правил:

– моделирование более сложных связей между критериями;

– использование динамической генерации правил из данных и экспертных сессий;

– интеграция методов автоматического извлечения знаний (например, через алгоритмы Rule Induction или Symbolic Regression).

3. Уточнение нейросетевой подсистемы:

– применение глубоких архитектур (например, GNN для графов входных параметров);

– обучение на синтетических и реальных данных из практики крупных ИТ-проектов;

– применение объяснимого машинного обучения (XAI) для интерпретации вкладов критериев.

4. Адаптация модели под конкретные отрасли:

– под BI/аналитику, финтех, IoT, системы реального времени;

– разработка отраслевых «профилей архитектуры» и соответствующих подмножеств правил.

5. Интеграция с системами управления знаниями и документооборотом:

– использование модели как части большой экспертной среды, хранящей кейсы, решения, выводы и обоснования;

– возможность обогащать модель в процессе эксплуатации – самообучающаяся архитектурная платформа.

На следующем этапе целесообразно сосредоточиться на апробации модели на практике, создании открытой библиотеки правил и запуске прототипа, который станет полезным инструментом в арсенале специалистов по проектированию и эксплуатации информационных систем.

Список источников

1. Islam M. A., Anderson D. T., Pinar A. J., Havens T. C., Scott G., Keller J. M. Enabling Explainable Fusion in Deep Learning with Fuzzy Integral Neural Networks // IEEE Transactions on Fuzzy Systems. 2020. V. 28. N. 7. P. 1291–1300. DOI 10.1109/TFUZZ.2019.2917124.

2. Aggoune A. Intelligent data integration from heterogeneous relational databases containing incomplete and uncertain information // Intelligent Data Analysis. 2022. V. 26. N. 1. P. 75–99. DOI 10.3233/IDA-205535.

3. Min K., Jananthan H., Kepner J. Fuzzy Relational Databases via Associative Arrays // 2024 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2024. P. 1–7. DOI 10.1109/HPEC62836.2024.10534916.

4. de Campos Souza P. V. Fuzzy neural networks and neuro-fuzzy networks: A review of the main techniques and applications used in the literature // Applied Soft Computing. 2020. V. 92. Article 106275. DOI 10.1016/j.asoc.2020.106275.

5. Ojha V., Abraham A., Snášel V. Heuristic design of fuzzy inference systems: A review of three decades of research // Engineering Applications of Artificial Intelligence. 2019. V. 85. P. 845–864. DOI 10.1016/j.engappai.2019.08.010.

6. Ahmad A., Rudrusamy G., Budiarto R., Samsudin A., Ramadass S. A Hybrid Rule Based Fuzzy-Neural Expert System for Passive Network Monitoring // Proceedings of the Arab Conference on Information Technology (ACIT). Dhaka, 2002. P. 746–752.

7. Yarushkina N., Moshkin V., Andreev I., Klein V., Beksaeva E. Hybridization of fuzzy inference and self-learning fuzzy ontology-based semantic data analysis // Proceedings of the First International Scientific Conference “Intelligent Information Technologies for Industry” (IITI'16). Sochi: Springer, 2016. P. 277–285.

8. Jiyad A. M., Nemer Z. N. A Hybrid Fuzzy-Neural Network Approach for Advanced Pattern Recognition and Predictive Analytic // Journal of Information Systems Engineering and Management. 2025. V. 10. N. 32s. DOI 10.52783/jisem.v10i32s.5183.

9. Al-Ashoor A., Lilik F., Nagy S. A Systematic Analysis of Neural Networks, Fuzzy Logic and Genetic Algorithms in Tumor Classification // Applied Sciences. 2025. V. 15. N. 9. Article 5186. DOI 10.3390/app15095186.

10. Martinez-Cruz C., Noguera J. M., Vila M. A. Flexible queries on relational databases using fuzzy logic and ontolo-

gies // Information Sciences. 2016. V. 366. P. 150–164. DOI 10.1016/j.ins.2016.05.038.

11. Moura B., Soares Y., Sampaio L., Reiser R., Pilla M., Yamin A. fGrid: Uncertainty Variables Modeling for Computational Grids using Fuzzy Logic // 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). Vancouver: IEEE, 2016. P. 2249–2256. DOI 10.1109/FUZZ-IEEE.2016.7737973.

References

1. Islam M. A., Anderson D. T., Pinar A. J., Havens T. C., Scott G., Keller J. M. Enabling Explainable Fusion in Deep Learning with Fuzzy Integral Neural Networks. *IEEE Transactions on Fuzzy Systems*, 2020, vol. 28, no. 7, pp. 1291-1300. DOI 10.1109/TFUZZ.2019.2917124.

2. Aggoune A. Intelligent data integration from heterogeneous relational databases containing incomplete and uncertain information. *Intelligent Data Analysis*, 2022, vol. 26, no. 1, pp. 75-99. DOI 10.3233/IDA-205535.

3. Min K., Jananthan H., Kepner J. Fuzzy Relational Databases via Associative Arrays. *2024 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2024, pp. 1-7. DOI 10.1109/HPEC62836.2024.10534916.

4. de Campos Souza P. V. Fuzzy neural networks and neuro-fuzzy networks: A review of the main techniques and applications used in the literature. *Applied Soft Computing*, 2020, vol. 92, article 106275. DOI 10.1016/j.asoc.2020.106275.

5. Ojha V., Abraham A., Snášel V. Heuristic design of fuzzy inference systems: A review of three decades of research. *Engineering Applications of Artificial Intelligence*, 2019, vol. 85, pp. 845-864. DOI 10.1016/j.engappai.2019.08.010.

6. Ahmad A., Rudrusamy G., Budiarto R., Samsudin A., Ramadass S. A Hybrid Rule Based Fuzzy-Neural Expert System for Passive Network Monitoring. *Proceedings of the Arab Conference on Information Technology (ACIT)*. Dhaka, 2002. Pp. 746-752.

7. Yarushkina N., Moshkin V., Andreev I., Klein V., Beksaeva E. Hybridization of fuzzy inference and self-learning fuzzy ontology-based semantic data analysis. *Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'16)*. Sochi, Springer, 2016. Pp. 277-285.

8. Jiyad A. M., Nemer Z. N. A Hybrid Fuzzy-Neural Network Approach for Advanced Pattern Recognition and Predictive Analytic. *Journal of Information Systems Engineering and Management*, 2025, vol. 10, no. 32s. DOI 10.52783/jisem.v10i32s.5183.

9. Al-Ashoor A., Lilik F., Nagy S. A Systematic Analysis of Neural Networks, Fuzzy Logic and Genetic Algorithms in Tumor Classification. *Applied Sciences*, 2025, vol. 15, no. 9, article 5186. DOI 10.3390/app15095186.

10. Martinez-Cruz C., Noguera J. M., Vila M. A. Flexible queries on relational databases using fuzzy logic and ontologies. *Information Sciences*, 2016, vol. 366, pp. 150-164. DOI 10.1016/j.ins.2016.05.038.

11. Moura B., Soares Y., Sampaio L., Reiser R., Pilla M., Yamin A. fGrid: Uncertainty Variables Modeling for Computational Grids using Fuzzy Logic. *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Vancouver, IEEE, 2016. Pp. 2249-2256. DOI 10.1109/FUZZ-IEEE.2016.7737973.

Статья поступила в редакцию 09.09.2025; одобрена после рецензирования 18.11.2025; принята к публикации 07.04.2026
The article was submitted 09.09.2025; approved after reviewing 18.11.2025; accepted for publication 07.04.2026

Информация об авторах / Information about the authors

Сергей Михайлович Туркин – аспирант кафедры робототехнических систем и интеллектуальных технологий; Санкт-Петербургский государственный лесотехнический университет имени С. М. Кирова; serg.dinamo19@mail.ru

Sergei M. Turkin – Postgraduate Student of the Department of Robotic Systems and Intelligent Technologies; Saint Petersburg State Forest Technical University named after S. M. Kirov; serg.dinamo19@mail.ru

Сергей Александрович Иванов – кандидат технических наук, доцент; доцент кафедры робототехнических систем и интеллектуальных технологий; Санкт-Петербургский государственный лесотехнический университет имени С. М. Кирова; kemsit@mail.ru

Sergei A. Ivanov – Candidate of Technical Sciences, Assistant Professor; Assistant Professor of the Department of Robotic Systems and Intelligent Technologies; Saint Petersburg State Forest Technical University named after S. M. Kirov; kemsit@mail.ru