Разработка интеллектуального модуля интерактивного анализа данных в реляционных базах данных

Выпускная квалификационная работа

Студент: Гимазетдинов Дмитрий Русланович

Московский авиационный институт
(Национальный исследовательский университет)

26 декабря 2025 г.

Содержание

Актуальность темы

Тенденции

- Стремительный рост объемов копроративных данных. По прогнозам **IDC**, объем данных в ближайшем будующем достигнет 175 зеттабайт.
- Согласно рейтингу **DB-Engines**, основным стандартом хранения остаются реляционные базы данных.

Проблема

Высокий порог входа для аналитики: бизнес-пользователи не владеют **SQL**, а ожидание отчетов от IT-отдела замедляет принятие решений. Классические BI-инструменты требуют ручной настройки дашбордов.

Решение: Внедрение NLIDB (Natural Language Interface to Database) на базе больших языковых моделей (LLM) для автоматической генерации SQL-запросов и визуализации ответов.

Цель и задачи работы

Цель: Разработать программное решение, которое устранит проблему обращения к большим данным, пользователям без должного технического образования

Задачи исследования:

1. Протестировать все современные методы для трансляции естественного языка в язык запросов (text2sql);
2. Спроектировать RAG-архитектуру, для работы с таблицами в схеме БД, с выявлением метаинформации, связи таблиц;
3. Разработка механизма самокоррекции ИИ-агента, который позволит улучшить качество готового ответа;
4. Провести оценку качества генерации, при добавлении в проект механизма самокоррекции;
5. Реализовать микросервисную архитектуру;

Используемая база данных: «Авиаперевозки»

В качестве предметной области для тестирования системы выбрана демонстрационная база данных **Postgres Pro «Airlines»**.

Характеристики:

- **Схема:** `bookings`
- **Объем:** Содержит данные о полетах за 3 месяца (версия *demo-medium*), что позволяет оценивать производительность запросов.
- **Реалистичность:** База моделирует реальные бизнес-процессы авиакомпании: от бронирования билета до выдачи посадочного талона.

Почему эта БД?

В отличие от синтетических датасетов (TPC-H), здесь присутствуют:

- Неоднозначности естественного языка (города, имена).
- Сложные связи (многие-ко-многим).
- Временные ряды (расписание рейсов).

Структура данных (ER-модель)

База данных состоит из 8 основных таблиц, связанных внешними ключами, что создает необходимую сложность для генерации JOIN-запросов.

- **`bookings`, `tickets`**: Бронирования и билеты (пассажиры, суммы, контакты).
- **`flights`**: Рейсы (время вылета/прилета, статусы, задержки).
- **`ticket_flights`, `boarding_passes`**: Стыковочные рейсы и регистрация на рейс.
- **`airports_data`, `aircrafts_data`**: Справочники аэропортов и самолетов (включая данные в `JSONB`).

Сложность для AI-агента: Модели необходимо различать понятия «рейс» (по расписанию) и «фактический вылет», а также корректно работать с географическими координатами и часовыми поясами.

Сравнительный анализ аналогов

Табл. 1 – Сравнение разрабатываемого решения с существующими инструментами

Функционал	**Vanna.AI**	**LangChain SQL**	**Metabase**	**Разработка**
Генерация SQL (Text-to-SQL)	+	+	+	+
RAG для схемы БД	+	Ограничено	-	+
Локальный запуск (Privacy)	+	+	-	+
Механизм самокоррекции (Self-Heal)	-	-	-	+
Авто-визуализация данных	±	-	+	+

Ключевое преимущество: Объединение приватности (Local LLM), точности RAG и автоматической визуализации в едином контуре.

Применение RAG в задачах NLIDB

В работе обосновано использование парадигмы **RAG (Retrieval-Augmented Generation)** для преодоления ограничений стандартных LLM при работе с базами данных:

- **Динамическое извлечение метаданных:** В отличие от статической передачи DDL, наш подход позволяет извлекать только релевантные описания таблиц и связей, минимизируя шум в промпте.
- **Семантическое связывание (Schema Linking):** Использование векторных представлений для сопоставления пользовательских терминов с техническими именами колонок и таблиц.

Концепция решения

Переход от модели «Текст-в-SQL» к архитектуре **интеллектуального агента**, который оперирует не просто текстом, а графом знаний о структуре БД.

Научная новизна работы

Научная новизна исследования заключается в разработке и реализации следующих положений:

1. **Разработан адаптивный метод семантического поиска контекста схемы БД:** В отличие от существующих решений, **наше решение** использует многоуровневое индексирование (названия таблиц, комментарии к колонкам, примеры данных), что повышает точность генерации на сложных схемах (более 50 таблиц).
2. **Предложен оригинальный алгоритм многошаговой самокоррекции (Self-Correction):** Реализован механизм «критика-исполнителя», где агент анализирует логи ошибок СУБД и итеративно исправляет синтаксические и логические ошибки в сгенерированном SQL-коде без участия пользователя.

Оценка эффективности предложенных решений

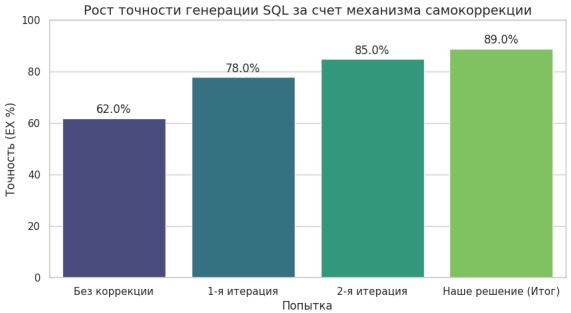

Рис. 1. Влияние самокоррекции на точность (EX)

Ключевые показатели:

- **Точность:** Механизм самокоррекции (Self-Correction) повышает долю успешных запросов с **62% до 89%**.
- **Надежность:** Система устойчива к галлюцинациям LLM благодаря строгому связыванию схемы.

Итог

Наше решение пригодно для промышленных СУБД со сложной структурой данных.

Технологический стек реализации

В работе используется современный Open-Source стек для обеспечения гибкости и безопасности данных:

Backend & API:

- **Python**: Основной язык разработки.
- **FastAPI**: Высокопроизводительный фреймворк для реализации REST API агента.
- **SQLAlchemy**: ORM для взаимодействия с целевой базой данных.

LLM Orchestration:

- **LangChain**: Фреймворк для построения цепочек рассуждений (Chain-of-Thought).
- **Ollama**: Локальный запуск LLM (Llama 3, Mistral) для предотвращения утечек данных.

Data & Storage:

- **PostgreSQL**: Целевая СУБД.
- **Vector DB (Chroma)**: Хранение эмбеддингов описаний таблиц для семантического поиска.

Frontend & Visualization:

- **Streamlit**: Быстрое создание интерактивного UI.
- **Plotly**: Библиотека для динамической визуализации результатов SQL-запросов.

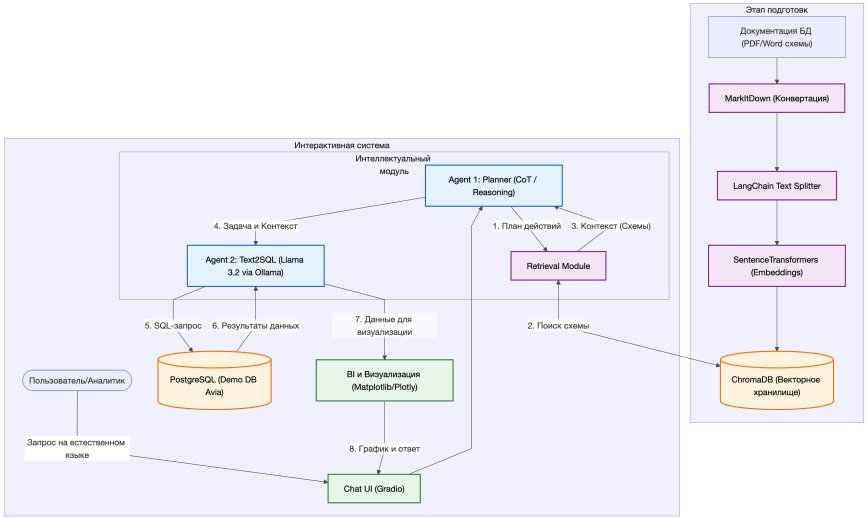

Рис. 2: Верхнеуровневая диаграмма

Flow

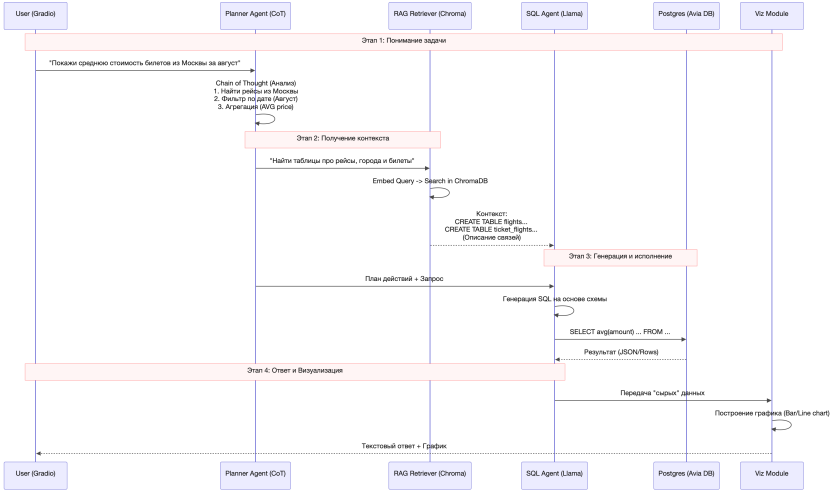

Рис. 3: Pipeline использования

Дальнейшее развитие

Дальнейшие шаги развития проекта

1. Проэксперементировать с пайплайном RAG системы
2. Добиться высокого показателя RAG системы при увеличении кол-ва таблиц в схеме
3. Развертывание более большого дампа (за больший отчетный период)
4. Довести все остальное до ума, доделать шаблон LaTeX и подготовить презентацию.

Список использованных источников

1. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. — 8-е изд. — М. : Вильямс, 2016. — 1328 с.
2. Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. — 3-е изд. — М. : Вильямс, 2019. — 1408 с.
3. Lewis, P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis, E. Perez et al. // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — P. 9459–9474.
4. Zhong, V. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning / V. Zhong, C. Xiong, R. Socher // arXiv preprint arXiv:1709.00103. — 2017.
5. Postgres Pro Standard : документация к СУБД. — 2024. — URL: https://postgrespro.ru/docs/ (дата обращения: 15.12.2024).
6. LangChain Documentation : Retrieval-Augmented Generation (RAG). — 2024. — URL: https://python.langchain.com/docs/ (дата обращения: 06.11.2025).
7. Ollama: Get up and running with large language models locally. — 2024. — URL: https://ollama.com/ (дата обращения: 20.12.2024).
8. Yu, T. Spider: A Large-Scale Hierarchical Dataset for Semantic Parsing and Text-to-SQL Task / T. Yu, R. Zhang et al. // Proceedings of EMNLP. — 2018.