MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE REPUBLIC OF KAZAKHSTAN

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
KAZAKHSTAN

ХАЛЫҚАРАЛЫҚ УНИВЕРСИТЕТІ INTERNATIONAL UNIVERSITY SINCE 2009 МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Published since 2020.
Volume 7. 2 (26). 2026
April–June

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

2020 жылдан бері шығарылады
Том 7. 2 (26). 2026
Сәуір-Маусым

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Издается с 2020 г.
Том 7. 2 (26). 2026
Апрель-Июнь

Свидетельство о постановке на учет периодического печатного издания в Министерство информации и общественного развития Республики Казахстан № KZ82VPY00020475, выданное от 20.02.2020 г.

Зарегистрировано в Международном центре регистрации серийных изданий ISSN (ЮНЕСКО, Париж, Франция). ISSN 2708–2032 (print), ISSN 2708–2040 (online)

Журнал входит в Перечень научных изданий, рекомендуемых КОКНВО МНВО РК для публикации основных результатов научной деятельности.

International Journal of Information and Communication Technologies
Periodicity: 4 times a year.
Languages: Kazakh, Russian, English
DOI prefix: 10.54309
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Thematic focus: "Information technology"; "Digital technologies in the development of socio-economic systems"; "Information security and communication technologies".

Journal website: https://journal.iitu.edu.kz
Owner: International Information Technology University JSC (Almaty).

Халықаралық ақпараттық және коммуникациялық технологиялар журналы
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Префикс DOI: 10.54309
Мерзімділігі: жылына 4 рет.
Басылым тілі: қазақ, орыс, ағылшын.
Тақырып бағыты: "Ақпараттық технологиялар"; "Ақпараттық қауіпсіздік және коммуникациялық технологиялар"; "Әлеуметтік-экономикалық жүйелерді дамытудағы цифрлық технология".
Журнал сайты: https://journal.iitu.edu.kz

Меншік иесі: АҚ «Халықаралық ақпараттық технологиялар университеті» (Алматы қ.).

Международный журнал информационных и коммуникационных технологий
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Префикс DOI: 10.54309
Периодичность: 4 выпусков в год.
Язык издания: казахский, русский, английский.
Тематическая направленность: "Информационные технологии"; "Информационная безопасность и коммуникационные технологии"; "Цифровые технологии в развитии социо-экономических систем".
Сайт журнала: https://journal.iitu.edu.kz

Собственник: АО «Международный университет информационных технологий» (г. Алматы).

CONTENTS

DIGITAL TECHNOLOGIES IN THE DEVELOPMENT OF SOCIO-ECONOMIC SYSTEMS

INFORMATION TECHNOLOGY

INFORMATION SECURITY AND COMMUNICATIONTECHNOLOGIES

МАЗМҰНЫ

ӘЛЕУМЕТТІК-ЭКОНОМИКАЛЫҚ ЖҮЙЕЛЕРДІ ДАМЫТУДАҒЫ ЦИФРЛЫҚ ТЕХНОЛОГИЯЛАР

АҚПАРАТТЫҚ ТЕХНОЛОГИЯЛАР

АҚПАРАТТЫҚ ҚАУІПСІЗДІК ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАРҒА АРНАЛҒАН

СОДЕРЖАНИЕ
ЦИФРОВЫЕ ТЕХНОЛОГИИ В РАЗВИТИИСОЦИО-ЭКОНОМИЧЕСКИХ СИСТЕМ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Vol. 7. Is.2. Number 26 (2026). Pp. 110–130
Journal homepage: https://journal.iitu.edu.kz
https://doi.org/10.54309/IJICT.2026.26.2.008
UDC 004.8:004.42:004.65

A MATHEMATICAL AND ALGORITHMIC APPROACH TO THE DEVELOPMENT OF AN INTELLIGENT TEXT-TO-SQL SYSTEM BASED ON LARGE LANGUAGE MODELS

B.Z. Kenzhegulov[1], Zh.T. Bilyalova[1], K.N. Uteuliyeva[1]*, L. Nurgaliyeva[2], Sh.S. Nurzhanova[1]

[1]Atyrau University named after Kh. Dosmukhamedov, Atyrau, Kazakhstan;
[2]Saidot Ltd., Хельсинки, Финляндия.
E-mail: kamka_n@mail.ru

Beket Kenzhegulov — Doctor of Technical Sciences, Professor of the Department of Mathematics and Methods of Teaching Mathematics, Atyrau University named after Kh. Dosmukhamedov, Kazakhstan
https://orcid.org/0000-0001-6230-2926;
Zhupar Bilyalova — Candidate of Pedagogical Sciences, Professor of the Department of Mathematics and Methods of Teaching Mathematics, Atyrau University named after Kh. Dosmukhamedov, Kazakhstan
https://orcid.org/0000-0002-3362-8086;
Kamka Uteuliyeva — Candidate of Physical and Mathematical Sciences, Associate Professor of the Department of Mathematics and Methods of Teaching Mathematics, Atyrau University named after Kh. Dosmukhamedov, Kazakhstan
E-mail: kamka_n@mail.ru, https://orcid.org/0009-0004-1195-1642;
Lunara Nurgaliyeva — Master's degree holder in AI Security and Machine Learning, Saidot Ltd., Helsinki, Finland
https://orcid.org/0009-0005-5252-9525;
Sharbat Nurzhanova — Candidate of Pedagogical Sciences, Professor of the Department of Mathematics and Methods of Teaching Mathematics, Atyrau University named after Kh. Dosmukhamedov, Kazakhstan
https://orcid.org/0009-0006-7726-1999.

Abstract. The article addresses the problem of developing an intelligent Text-to-SQL system designed to automatically transform user queries expressed in natural language into SQL queries for relational databases. The relevance of the topic is associated with the rapid growth of structured data and the need to increase the accessibility of analytical tools for users who do not possess professional-level SQL skills. The aim of the study is to analyze and practically test methods for applying large language models to automate SQL analytics, as well as to compare three approaches: fine-tuning, RAG, and agent-based systems. To achieve this aim, the paper considers the mathematical formulation of the problem, the architecture of the RAG pipeline, the mechanism of vector search, and agent-based error self-correction. The results show that the combination of LLM, RAG, and an agent-based approach improves the adaptability, accuracy, and practical applicability of Text-to-SQL systems in corporate analytics. It is concluded that a combined approach is appropriate for complex analytical queries and dynamically changing databases.

Keywords: Text-to-SQL, large language models, SQL, RAG, agent-based systems, vector search, relational databases

For citation: B.Z. Kenzhegulov, Zh.T. Bilyalova, K.N. Uteuliyeva, L. Nurgaliyeva, Sh.S. Nurzhanova (2026). A mathematical and algorithmic approach to the development of an intelligent text-to-sql system based on large language models // International journal of information and communication technologies. Vol. 7. No. 26. Pp. 110–130. https://doi.org/10.54309/IJICT.2026.26.2.008. (In Eng.).

Conflict of interest: The authors declare that there is no conflict of interest.

ҮЛКЕН ТІЛДІК МОДЕЛЬДЕР НЕГІЗІНДЕ ИНТЕЛЛЕКТУАЛДЫ TEXT-TO-SQL ЖҮЙЕСІН ӘЗІРЛЕУДІҢ МАТЕМАТИКАЛЫҚ-АЛГОРИТМДІК ТӘСІЛІ

Б.З. Кенжегулов[1], Ж.Т. Билялова[1], К.Н. Утеулиева[1]*, Л. Нургалиева[2], Ш.С. Нуржанова[1]

[1]Х. Досмұхамедов атындағы Атырау университет, Атырау, Қазақстан;
[2]Saidot Ltd., Хельсинки, Финляндия.
E-mail: kamka_n@mail.ru

Бекет Кенжегулов — техника ғылымдарының докторы, Х. Досмұхамедов атындағы Атырау университетінің математика және математики оқыту әдістемесі кафедрасының профессоры, Атырау , Қазақстан
https://orcid.org/0000-0001-6230-2926;
Жупар Билялова — педагогика ғылымдарының кандидаты, Х. Досмұхамедов атындағы Атырау университетінің математика және математики оқыту әдістемесі кафедрасының профессоры, Атырау, Қазақстан
https://orcid.org/0000-0002-3362-8086;
Камка Утеулиева — физика - математика ғылымдарының кандидаты, Х. Досмұхамедов атындағы Атырау университетінің Математика және математики оқыту әдістемесі кафедрасының қауымдастырылған профессоры, Атырау, Қазақстан
E-mail: kamka_n@mail.ru, https://orcid.org/0009-0004-1195-1642;
Лунара Нургалиева — магистр, AI Safety ML/LLM Engineering, Saidot Ltd., Хельсинки, Финляндия

https://orcid.org/0009-0005-5252-9525;
Шарбат Нуржанова — педагогика ғылымдарының кандидаты, Х. Досмұхамедов атындағы Атырау университетінің математика және математики оқыту әдістемесі кафедрасының қауымдастырылған профессоры , Атырау, Қазақстан
https://orcid.org/0009-0006-7726-1999.

Аннотация. Мақалада пайдаланушылардың табиғи тілде берілген сұраныстарын реляциялық деректер қорларына арналған SQL-сұраныстарға автоматты түрде түрлендіруге бағытталған интеллектуалды Text-to-SQL жүйесін әзірлеу мәселесі қарастырылады. Зерттеу тақырыбының өзектілігі құрылымдалған деректер көлемінің қарқынды өсуімен және SQL тілін кәсіби деңгейде меңгермеген пайдаланушылар үшін аналитикалық құралдардың қолжетімділігін арттыру қажеттілігімен байланысты. Зерттеудің мақсаты — SQL-аналитиканы автоматтандыруда үлкен тілдік модельдерді қолдану әдістерін талдау және практикалық апробациядан өткізу, сондай-ақ Fine-tuning, RAG және агенттік жүйелер сияқты үш тәсілді салыстыру. Осы мақсатқа жету үшін жұмыста мәселенің математикалық қойылымы, RAG-пайплайн архитектурасы, векторлық іздеу механизмі және агенттік қателерді өздігінен түзету тәсілі қарастырылды. Нәтижесінде LLM, RAG және агенттік тәсілді біріктіру корпоративтік аналитикадағы Text-to-SQL жүйелерінің бейімделгіштігін, дәлдігін және практикалық қолданылуын арттыратыны көрсетілді. Қорытындыда күрделі аналитикалық сұраныстар мен динамикалық түрде өзгеретін деректер қорлары үшін біріктірілген тәсілді қолдану тиімді екені анықталды.

Түйінді сөздер: Text-to-SQL, үлкен тілдік модельдер, SQL, RAG, агенттік жүйелер, векторлық іздеу, реляциялық деректер қоры

Дәйексөздер үшін: Кенжегулов Б.З., Билялова Ж.Т., Утеулиева К.Н., Нургалиева Л., Нуржанова Ш.С. (2026). Үлкен тілдік модельдер негізінде интеллектуалды text-to-sql жүйесін әзірлеудің математикалық-алгоритмдік тәсілі // Халықаралық ақпараттық және коммуникациялық технологиялар журналы. Т. 7. No. 26. Б. 110–130. https://doi.org/10.54309/IJICT.2026.26.2.008. (Ағыл. тіл.).

Мүдделер қақтығысы: Авторлар осы мақалада мүдделер қақтығысы жоқ деп мәлімдейді.

МАТЕМАТИКО-АЛГОРИТМИЧЕСКИЙ ПОДХОД К РАЗРАБОТКЕ ИНТЕЛЛЕКТУАЛЬНОЙ TEXT-TO-SQL СИСТЕМЫ НА ОСНОВЕ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Б.З. Кенжегулов[1], Ж.Т. Билялова[1], К.Н. Утеулиева[1]*, Л. Нургалиева[2], Ш.С. Нуржанова[1]

[1]Атырауский университет имени Х.Досмухамедова, Атырау, Қазақстан;
[2]Saidot Ltd., Хельсинки, Финляндия.

E-mail: kamka_n@mail.ru

Бекет Кенжегулов — доктор технических наук, профессор кафедры математики и методики преподавания математики Атырауского университета имени Х. Досмухамедова, Атырау, Казахстан
https://orcid.org/0000-0001-6230-2926;
Жупар Билялова — кандидат педагогических наук, профессор кафедры математики и методики преподавания математики. Атырауский университет имени Х.Досмухамедова; Казахстан
https://orcid.org/0000-0002-3362-8086;
Камка Утеулиева — кандидат физико-математических наук, доцент кафедры мате-матики и методики преподавания математики Атырауского университета имени Х. Досмухамедова, Атырау, Казахстан
E-mail: kamka_n@mail.ru, https://orcid.org/0009-0004-1195-1642;
Лунара Нургалиева — магистр, AI Safety ML/LLM Engineering, Saidot Ltd., Хельсинки, Финляндия
https://orcid.org/0009-0005-5252-9525;
Шарбат Нуржанова — кандидат педагогических наук, профессор кафедры математики и методики преподавания математики Атырауского университета имени Х. Досмухамедова, Атырау, Казахстан
E https://orcid.org/0009-0006-7726-1999.

Аннотация. В статье рассматривается проблема разработки интеллектуальной системы Text-to-SQL, предназначенной для автоматического преобразования пользовательских запросов на естественном языке в SQL-запросы к реляционным базам данных. Актуальность темы связана с ростом объемов структурированных данных и необходимостью повышения доступности аналитических инструментов для пользователей, не владеющих SQL на профессиональном уровне. Цель исследования заключается в анализе и практической апробации методов применения больших языковых моделей для автоматизации SQL-аналитики, а также в сравнении трех подходов: Fine-tuning, RAG и агентных систем. Для достижения цели рассмотрены математическая постановка задачи, архитектура RAG-пайплайна, механизм вектор-ного поиска и агентная самокоррекция ошибок. В результате показано, что сочета-ние LLM, RAG и агентного подхода повышает адаптивность, точность и практиче-скую применимость Text-to-SQL систем в корпоративной аналитике. Сделан вывод о целесообразности комбинированного подхода для сложных аналитических запросов и динамически изменяющихся баз данных.

Ключевые слова: Text-to-SQL, большие языковые модели, SQL, RAG, агент-ные системы, векторный поиск, реляционные базы данных

Для цитирования: Кенжегулов Б.З., Билялова Ж.Т., Утеулиева К.Н., Нур-

галиева Л., Нуржанова Ш.С. (2026). Математико-алгоритмический подход к разработке интеллектуальной text-to-sql системы на основе больших языковых моделей // Международный журнал информационных и коммуникационных технологий. Т. 7. No. 26. Стр. 110–130. https://doi.org/10.54309/IJICT.2026.26.2.008. (На англ.).

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Introduction.

The modern digital economy generates unprecedented volumes of structured data: according to Statista, the total volume of global data will reach 175 zettabytes by 2025. In this environment, the ability to quickly extract accurate analytical information from databases is a significant competitive advantage for organizations. SQL (Structured Query Language) – the standard query language for relational DBMS – remains the main tool for accessing structured data: as of 2025, relational databases occupy more than 60 % of the DBMS market (Zhong et al., 2017).

At the same time, SQL queries pose a significant barrier for a significant portion of business users. According to Defog.ai analysts, analytics department employees spend 30 to 50% of their work time formulating and debugging data queries, with each non-trivial query potentially requiring the involvement of a qualified developer or data engineer (Yu et al., 2018). This reduces operational efficiency and creates a bottleneck in the decision-making chain.

Large Language Models (LLMs)—neural network architectures based on attention mechanisms (Transformers) trained on trillions of text tokens—open up a fundamentally new opportunity: the automatic transformation of natural language queries into syntactically and semantically correct SQL queries. This task, commonly referred to as Text- to -SQL, is one of the most practical applications of LLMs in the field of data science.

to-SQL problem is not new: the first attempts to solve it were made back in the 1970s using rule-based systems (LUNAR, BASEBALL). However, it was the advent of large pre-trained The introduction of transformer models (BERT, GPT, T5) in 2018–2020, and then super-large LLMs (GPT-4, LLaMA, Claude) in 2022–2024, provided a qualitative leap in accuracy: from 60–70 % on the Spider benchmark to 93 % for the specialized SQLCoder-70B model (Li et al., 2023).

The relevance of this work stems from the gap between the high potential of modern LLMs and the complexity of their practical implementation in corporate analytical systems. A comparative study of three approaches to solving the Text- to -SQL problem—fine-tuning, RAG, and agent-based systems—is particularly important, as each has its own advantages, limitations, and requirements for effective application. Furthermore, the domestic scientific literature lacks comprehensive studies combining theoretical analysis, practical implementation, and economic evaluation of these technologies.

The aim of the work is to research and practically test methods for using large language models to automate data analysis in SQL, to comparatively evaluate three main technologies (paradigms) – Fine- tuning, RAG, agent systems – and to determine the conditions for their effective use in a corporate environment.

To achieve the goal, the following research objectives were defined:

- to -SQL problems using LLM, and to identify their comparative advantages and limitations.
- LLM fine-tuning technologies (Full Fine- tuning, LoRA, QLoRA, RLHF) as applied to the Text- to -SQL task; to explore the practical aspects of deploying the SQLCoder model.
- To review the RAG (Retrieval-Augmented Generation) technology and explore its implementation in the specialized Vanna.AI framework.
- Develop an agent-based text-to-SQL conversion system with an automatic verification and error correction mechanism based on the smolagents framework.

The object of the research is large language models and methods of their application to automate work with relational databases.

The subject of the research is methods, architectures and software frameworks that ensure the transformation of natural language queries into SQL queries (Text-to-SQL task) with the required level of accuracy and practical applicability.

The developed solutions (agent system on smolagents, Vanna.AI configuration with BigQuery / PostgreSQL, templates for local deployment of SQLCoder) can be directly applied in the analytical departments of organizations to reduce dependence on qualified SQL developers and speed up the acquisition of analytical data.

Materials and methods.

Task Text-to-SQL: production, benchmarks And methods solutions

to -SQL problem is formally defined as follows: given a relational database schema $S = \{T_1, T_2, \ldots, T_n\}$ (where each T_i includes a table name, a list of columns and their data types, primary and foreign keys) and a user question in natural language Q, it is necessary to generate an SQL query R, the execution of which on this schema returns the correct answer to the question Q (Lewis et. al., 2020).

The complexity of the task is determined by several factors.

The first is the semantic gap: the same meaning can be expressed by many different natural language formulations, and the model must invariantly map them to the same SQL.

The second is the structural complexity of the schema: real corporate databases contain tens and hundreds of tables, thousands of columns, and their meaning is often encoded in abbreviated or uninformative names.

Third, understanding the business context is essential: the query “show top clients” requires knowledge of how a specific organization defines the concept of “top client” (by revenue, by purchase frequency, or other metrics).

At the time of writing, the BIRD (2023) benchmark is the most relevant, as it includes real databases with «polluted» data and requires the model to incorporate external knowledge (e.g., an understanding of industry terminology). Defog ‘s specialized SQL-Eval is crucial because it uses schemas guaranteed not to have been present in the training set of the models being tested, ensuring a fair assessment of generalization ability.

To objectively evaluate the quality of Text-to-SQL systems, several standard benchmarks have been developed (Table 1).

Table 1 – Main benchmarks for evaluating Text- to -SQL systems

Benchmark	Year		Peculiarities	Metrics
WikiSQL	2017	80 654	Simple queries, one table	Execution Accuracy
Spider	2018	10 181	200 databases, complex JOINs, cross- domain	Exact Match + Execution
BIRD	2023	12,751	Real databases, error annotations, external knowledge	Execution Accuracy + Valid Efficiency
SQL-Eval (Defog)	2023	200 (hold-out)	Schemes NOT from training, corporate patterns	% of correctly executed queries

The main quality metric is the accuracy of execution (Execution Accuracy (EX) is the proportion of queries whose results match the benchmark. Unlike Exact Match Accuracy (EM), EX takes into account semantically equivalent queries with different syntax, making it a more realistic estimate.

Currently, there are four main types (paradigms) of LLM application for solving the Text- to -SQL problem

1. Zero -shot and few -shot prompting. Basic LLM (GPT-4, Claude, Llama) takes a schema description and a user question as input and then generates SQL in a single pass without any preliminary model modifications. The method is simple to implement and requires no training costs, but is limited in accuracy: 60–82 % on SQL- Eval. The main drawback is the complete dependence on prompting skills and the model's inability to understand the specifics of a particular database.

Fine- tuning. The pre-trained base model undergoes an additional training step on a domain-specific dataset of (question → SQL) pairs specific to the target DBMS and business domain. This knowledge is embedded directly into the model's weights, ensuring high inference accuracy. An example is the SQLCoder family of models with an accuracy of up to 93 % on SQL- Eval (Brown et al., 2020). Limitations: the need for significant computing resources at the training stage and the need for retraining when the database schema changes.

3.RAG (Retrieval - Augmented Generation). Instead of changing the model weights, an external knowledge base is formed (a vector storage of DDL descriptions, SQL examples, and business documentation). Upon receiving a request, a semantic search for relevant context is performed, which is dynamically included in the LLM prompt. The method is flexible, allows for prompt knowledge base updates, and does not require retraining. It is implemented in the Vanna.AI framework.

Agent-based systems. LLM acts as the «brain» of the agent, with access to a set of tools: a database schema inspector, an SQL query executor, and a result verifier. The agent iteratively generates queries, executes them, analyzes errors, and corrects the queries until the correct result is obtained. This allows for automatic correction of errors unachievable with single-pass generation. Implemented using the smolagents framework (Vaswani et al., 2017).

A comparative analysis of the four approaches considered by the main parameters is presented in Table 2.

Table 1.2 shows that the choice of approach is determined by the characteristics of a specific project. For a stable database schema and high accuracy requirements, retraining is optimal; for a frequently changing schema and the need for corporate deployment, RAG is optimal; in the presence of complex multi-step analytical queries requiring verification, agent systems are optimal. In practice, the most powerful solutions combine several types: for example, Vanna.AI implements RAG on top of a pre-trained LLM (or retrained SQLCoder), and the agent system can use RAG to build the prompt.

Table 2 – Comparative analysis of approaches to solving the Text- to -SQL problem

Criterion	Fine- tuning	RAG	Agent-based approach	Zero shot (basic LLM)
Accuracy	High (93 % for SQLCoder-70B)	High with a good knowledge base	Very high (self-correction)	Average (65-82%)
Deployment cost	High (GPU, training)	Medium (vector storage)	Average (API calls)	Low
Data requirements	Labeled pairs (question→SQL)	DDL, documentation, SQL examples	Tools + database access	Not required
Knowledge updatable	Requires retraining	Knowledge base update	Dynamically	Not updating
Bug fixes	No (generation in 1 pass)	Partial (via RAG)	Yes (auto-verification)	No
Dependency on the database schema	Requires inclusion in prompt	Automatically from a vector database	On-the-fly circuit inspection	Manual transmission
Examples of systems	SQLCoder, DAIL-SQL	Vanna.AI, LlamaIndex	smolagents + SQL tool	GPT-4 (basic)
Optimal application	Stable circuit, high precision requirements	Corporate analytics, a frequently changing scheme	Complex multi-step queries	Prototypes, one-off requests

In addition to the basic types, a number of universal techniques for improving the quality of Text- to -SQL are used:

– Query decomposition (query decomposition): a complex question is broken down into subquestions , each of which is translated into a subquery or CTE (Common Table) Expression).

– Self -consistency: the model generates several SQL variants, executes them and selects a consistent result.

– Schematic coordination (schema linking): Before SQL is generated, explicit linking of the entities mentioned in the question to the tables and columns of the schema is performed.

– Using ensembles: results from multiple LLMs are aggregated to improve reliability.

Thus, the Text- to -SQL problem is a multi-parameter problem requiring the selection and combination of several technical solutions. In our paper, we sequentially explore three of the most practically relevant paradigms— pretraining, RAG, and agent-based systems—implementing and evaluating each.

Technology Retrieval-Augmented Generation (RAG): architecture And application.

Retrieval-Augmented Generation (RAG) is an architectural paradigm in which a pre-trained language model relies not only on the knowledge encoded in its weights to generate a response, but also on the current context dynamically retrieved from an external store. This type was described in detail in the work of Lewis et al. (2020) and has since become a standard engineering approach for building domain-specific systems based on LLM, including Text-to -SQL systems (Devlin et al., 2019). The primary motivation for using RAG in the Text-to -SQL task is the following: even the most powerful general-purpose LLMs lack a priori knowledge of the schema of a specific corporate database—table names, column values, filtering and aggregation business rules. Passing the entire schema to each query (as DDL) is inefficient for large databases (hundreds of tables), as it bloats the context window. RAG solves this problem: for each specific question, only the relevant schema fragments and SQL examples are extracted from the index and compactly included in the prompt.

The fundamental differences between RAG and Fine- tuning are shown in Table 3.

Table 3 – Comparative analysis of the RAG and Fine -tuning approaches for the Text-to -SQL task

Criterion	Fine- tuning	RAG
Principle	Knowledge is embedded into the model weights during the training phase	Knowledge is stored in an external index and retrieved dynamically
Domain Accuracy	Very high (93% for SQLCoder-70B)	High with a quality index (85-92% for Vanna.AI)
Knowledge update	Requires retraining – expensive and time-consuming	Adding/removing documents from the index is instant
Transparency	Low: The source of knowledge is hidden in the scales	High: The model clearly indicates which context it used
Launch cost	High: GPU, data engineering, annotation	Medium: vector database, embedding calculation
Hallucinations	Possible when going beyond the training	Reduced: The model is based on real-world context
Data privacy	The data is included in the training set	Data is stored in a controlled index and is not transferred to LLM.
Optimal application	Stable database design, high accuracy requirements	Frequently changing scheme, corporate analytics, chatbots

Table 3 shows that RAG is not a universal replacement for fine- tuning: when high accuracy requirements are met with a stable schema, specialized fine-tuning is still preferable. However, RAG offers significant advantages in terms of maintenance costs, transparency, and data privacy, making it the primary choice for enterprise analytics systems with frequently updated schemas.

The RAG system architecture includes two independent pipelines: indexing and query.

pipeline (preparatory stage, performed once):

1. Uploading documents: DDL table scripts, text documentation of business rules, examples of SQL queries with annotations.

2. Chunking: The source documents are cut into fragments, each of which will be independently represented by a vector.

3. Embedding computation: each chunk is transformed into a fixed-dimensional vector using the encoder model.

4. Saving to a vector database: vectors and source texts are saved for later retrieval.

pipeline (executed for each user question):

embedding calculation: the user's question is transformed into a vector of the same dimension.

2. Search for nearest neighbors: the vector DB contains the k most semantically similar pieces.

3. Ranking (optional): The found pieces are sorted by relevance using a reranker.

4. Prompt formation: the user's question is combined with the found pieces of context.

5. SQL generation: the prompt is passed to LLM, the model generates the SQL query.

6. Verification: SQL is executed in the database; if an error occurs, a retry cycle is performed (optional).

For prototyping Text- to -SQL systems, we recommend ChromaDB —an embedded DBMS that requires no dedicated server and is the default in Vanna.AI. For production systems with scalability and hybrid search requirements, Qdrant or pgvector are preferred (the latter is especially advantageous with an existing PostgreSQL infrastructure).

Embedding models. Embedding (vector representation) is the mapping of a text fragment to a point in a multidimensional space such that semantically similar texts are geometrically similar. The quality of embeddings directly determines search relevance. For the Text- to-SQL task, it is important that the model understands both the natural language of questions and the SQL syntax of DDL descriptions. It is recommended to use models from the text-embedding-3 series (OpenAI) for cloud solutions and multilingual-e5-large or BGE-M3 for on-premises deployments with Russian language support.

to -SQL tasks, the recommended slicing parameters are: chunk size – 512–1024 tokens; overlap – 10–15 % (64–128 tokens); strategy – «by DDL» for the schema, « recursive « for the documentation. Too little overlap leads to a loss of context at chunk boundaries; too much overlap leads to duplication and increased storage costs.

Chunking documents. The choice of slicing strategy critically impacts search quality: chunks that are too small lose context, while chunks that are too large introduce noise. For the Text-to-SQL task, slicing by logical schema units is optimal: one chunk = one table (CREATE TABLE + column comments). SQL query examples are stored in their entirety, along with the query annotation. The main slicing strategies are presented in Table 4.

Table 4 – Strategies for splitting documents into pieces in RAG systems

Strategy	Description	Pros	Cons
By symbols/tokens	The document is divided into fixed-length chunks with overlapping (overlap)	Fast, easy to implement	Can terminate sentences and SQL statements
By delimiters (Recursive)	First it is divided by paragraphs, then by sentences, then by words.	Maintains the integrity of sentences	Uneven pieces
By Markdown / structure	Separators are headings (#, ##); each section is a separate piece	Semantically clean blocks enriched with metadata	For structured documents only
Semantic slicing	Sentences are combined into pieces based on cosine similarity of embeddings	Maximum semantic integrity of the pieces	High computational costs
Sentence Window	Individual sentences are indexed, but a window of N adjacent sentences is passed to LLM	Precise search + rich context	Excessive context on wide window
By DDL/SQL (for Text-to-SQL)	Each CREATE TABLE is a separate chunk; SQL examples are stored in their entirety	Maximum relevance for the SQL generation task	Requires schema preprocessing

Vector Databases. The choice of a vector DBMS determines search performance, infrastructure cost, and filtering flexibility. A comparative analysis of the main solutions is presented in Table 5.

Table 5 – Comparison of vector databases for RAG systems

BD	License	Storage	Hybrid search	Metadata filtering	Peculiarities	Application
ChromaDB	Apache 2.0	Disk / memory	No	Yes	Embeddable, no-server mode; ideal for prototyping	Vanna.AI by default
Qdrant	Apache 2.0	Disk + cache	Yes (dense + sparse)	Yes (payload)	High performance, vector quantization	Production, corporate solutions
Weaviate	BSD-3	Disk	Yes (BM25 + vector)	Yes (GraphQL)	GraphQL API, built-in embedding models	Semantic search of documents
Pinecone	Propriet.	Cloud	Yes	Yes	Managed service, serverless pricing	Cloud production systems
pgvector	PostgreSQL		Partially	Yes (SQL WHERE)	PostgreSQL extension – no additional infrastructure	Integration into an existing PostgreSQL stack
FAISS	MIT	Memory	No	No	Meta AI library; maximum speed, no server	Research, Jupyter laptops

Search Enhancement Techniques. Basic semantic search is often insufficient for complex SQL scenarios. In Table 6, we systematize advanced search techniques and evaluate their applicability to Text- to -SQL.

-to -SQL tasks, the most effective combination is a hybrid search (dense + BM25) with Self- Query filtering by table name. Hybrid search ensures that relevant tables are found even with non-standard query formulations, while Self- Query allows for immediate exclusion of irrelevant tables when the query contains explicit references to specific entities.

Table 6 – Techniques for Improving RAG Search for Text- to -SQL Task

Technique	The essence	When to use	Relevance for Text- to -SQL
Semantic search (Dense)	Searching for nearest neighbors by cosine similarity of vectors	Always - Basic Method	High: Finds relevant DDL and SQL examples
BM25 / Full- text (Sparse)	Classic lexical search using TF-IDF	Queries with exact terms, table names	High: Exact column names, SQL keywords
Hybrid	Linear combination of Dense + Sparse results (RRF, α- fusion)	Production systems – the best balance	Very high: combines semantics and precise terms
HyDE	LLM generates a hypothetical answer; its embedding is used to search	Queries that are far from the document style	Medium: Useful when there is a discrepancy between the styles of the query and the DDL
MMR	Maximum edge-relevant diversity: avoids duplication of found pieces	Broad analytical questions	Medium: Useful when there are many similar SQL examples
Context compression	The found pieces are summed up by LLM before being passed to the generating model	Long documents, narrow context window	Low: DDLs are usually compact
Self- Query	LLM extracts structured filters (date, object type) from the question	Queries with explicit attributes	High: Filtering by table name, SQL dialect

RAG system quality assessment metrics. The quality of a RAG pipeline is assessed using a set of metrics covering both intermediate stages (search, ranking) and the final generation quality (Table 7).

Limitations of RAG technology include: search quality depends on the quality of the embedding model; degradation due to an incomplete or poorly structured index; difficulty setting up hybrid search and reranking; and the possibility of «losing» relevant information when the retrieval window is too narrow (top -k). For Text- to -SQL tasks, an additional risk is the situation where similar but logically different DDLs compete in search results, misleading the model regarding relationships between tables.

For automated evaluation, the RAGAS (Retrieval-Augmented Generation Assessment) framework is recommended. It allows for the calculation of all the listed metrics without the involvement of a human annotator, using the LLM itself as a «judge.» In the Text- to -SQL task, the Execution metric Accuracy is the final and most informative parameter, as it directly reflects the functional correctness of the generated queries.

Table 7 – RAG system quality assessment metrics for Text -to -SQL

Metrics	Definition	Interpretation for Text- to -SQL
Faithfulness	To what extent is the answer based on the context provided, rather than on the model's «fantasy»	Low fidelity → the model ignores DDL and generates SQL from memory, which leads to errors
Answer Relevance (response relevance)	To what extent does the generated SQL answer the question asked?	You can evaluate it automatically: execute SQL and compare the result with the reference one
Context Recall (completeness of context)	The percentage of relevant chunks (table DDL, SQL examples) that were found and included in the prompt	If the required table is not included in the context, the SQL will be incomplete.
Context Precision	The proportion of truly relevant pieces among all found	Noisy context (irrelevant tables) reduces SQL accuracy
Execution Accuracy (precision of execution)	The percentage of SQL queries whose execution result matches the benchmark	Main end-to-end metrics for Text-to-SQL RAG systems

Vanna.AI's Text- to -SQL Capabilities and Practical Applications

Vanna.AI is an open-source Python framework (MIT license), a specialized RAG implementation for the Text- to -SQL task. Unlike general-purpose RAG frameworks (LlamaIndex, LangChain), Vanna.AI was designed exclusively for SQL generation, which determines its architectural decisions and user interface (Hu et al., 2022).

Vanna.AI's key architectural solution is complete modularity: each of the three components (LLM backend, vector storage, and SQL engine) can be independently replaced without changing the rest of the code. This allows the framework to be adapted to any requirements—from cloud enterprise solutions to fully on-premises deployments without sharing data with third parties.

Supported integrations by category are listed in Table 8.

Table 8 – Supported Vanna.AI Integrations

Category	Supported options	Recommendation
LLM backend	OpenAI (GPT-4o, GPT-4), Anthropic (Claude), Google (Gemini), Ollama (Llama 3, Mistral), HuggingFace, any OpenAI - Interoperable API	GPT-4o – for maximum accuracy; Ollama + Llama 3 – for local private deployment
Vector storage	ChromaDB, Qdrant, Pinecone, Weaviate, pgvector, Vanna Cloud	ChromaDB – quick start; Qdrant / pgvector – production with hybrid search
DBMS / data source	PostgreSQL, MySQL, SQLite, BigQuery, Snowflake, DuckDB, MS SQL Server, ClickHouse and others via SQLAlchemy	Connecting via a standard connection string; multiple databases are supported simultaneously
User interface	Flask web app, Streamlit, Jupyter Notebook, Slack bot, REST API	Flask – ready-to-use chat; Streamlit – for quick prototyping
Training data format	DDL scripts (CREATE TABLE), SQL examples (question + SQL pairs), text documentation	Minimum: 5-10 DDL + 10-20 SQL pairs for basic work; 50+ pairs for production quality

Working with Vanna.AI is based on two consecutive phases: training and query.

The training phase does not involve changing the LLM weights—the term is used in the sense of «filling the vector index.» Three types of training data are loaded into the index:

1. DDL descriptions (CREATE TABLE with comments on columns and keys): provide the model with an understanding of the database structure.

2. Text documentation (business rules, glossaries): explains semantics that are not available from the schema (e.g., a category A customer is one who has made more than 10 orders).

3. SQL examples (question→SQL pairs): the most valuable data type; allows the model to learn from successful historical experience.

The corrected query was verified directly in BigQuery and returned correct results, confirming the reliability of the Auto Fix mechanism for correcting common dialect-specific SQL errors.

Feedback and learning mechanism based on successful queries. After each successful SQL query, Vanna.AI prompts you to save the (query → SQL) pair to a vector index. This enables continuous learning: the more successful queries accumulate, the more accurate the system becomes for a specific schema and business domain. Automatic saving can be enabled in the settings (auto_train = True), eliminating the need for manual confirmation.

New Question » to manage the session.

A comparison of Vanna.AI with alternative Python frameworks for Text -to -SQL is given in Table 9.

Table 9 – Comparison of Vanna.AI with alternative Text- to -SQL tools

Criterion	Vanna.AI	LlamaIndex + SQL	LangChain SQL Agent	Defog (cloud)
Open- source	Yes (MIT)	Yes (MIT)	Yes (MIT)	No (SaaS)
SQL Specialization	Full	Partial	Partial	Full
Built-in UI	Yes (Flask, Streamlit)	No	No	Yes (cloud)
Automatic visualization	Yes (Plotly)	No	No	Yes
Feedback mechanism	Yes (auto-save correct pairs)	No	No	Yes
SQL Auto-Correction	Auto Fix button	No	Through an agent	Yes
On-premises deployment	Yes (Ollama + ChromaDB)	Yes	Yes	No
Difficulty of setup	Low (3 lines of code)	Average	High	Very low (SaaS)

Table 9 shows that Vanna.AI occupies the niche of the most ready-to-use open-source solution: with minimal setup complexity, it provides a built-in UI, automatic visualization, and a feedback mechanism—features that are missing from the universal LlamaIndex and LangChain frameworks.

Despite significant practical advantages, a number of limitations of the framework must be taken into account:

– Quality depends on the training data: without a sufficient number of correct SQL examples (less than 10-15 pairs), the accuracy of the system will be comparable to the basic LLM without RAG.

– No guarantees of correctness: Vanna.AI does not formally verify the generated SQL – verification occurs only through actual execution in the database.

– Complex analytical queries with multi-level subqueries, recursive CTEs, or specific window functions may require multiple Auto Fix iterations or manual intervention.

– Cloud Privacy: When using Vanna In cloud or cloud LLM (GPT-4, Claude API), database schema fragments and user questions are transmitted to third-party servers. For sensitive data, local deployment is required (Ollama + ChromaDB).

Conclusion.

RAG technology is an architecturally elegant solution to the Text- to -SQL problem: it combines the power of pre-trained LLMs with a dynamically updated knowledge base of the specific database schema. Comparative analysis shows that RAG outperforms Fine-tuning in flexibility, transparency, and maintenance costs, while achieving slightly lower maximum accuracy.

The Vanna.AI framework is the most mature open-source implementation of this approach: its modular architecture ensures compatibility with any LLM, vector DBMS, and SQL database; a built-in Flask interface, automatic Plotly visualization, and an automatic error correction mechanism make the system ready for industrial use. Practical testing on a dataset Orders (BigQuery) demonstrated correct SQL generation for RFM analysis, YoY comparisons, and ABC-XYZ segmentation tasks. The identified limitations include the dependence of quality on the volume of training data and the need for local deployment under strict privacy requirements, which prompted the study of additional self-correction mechanisms, namely agent-based systems.

Agent-based text-to-SQL conversion system with automatic error correction

An agent system in the context of LLM application is an architecture in which the language model acts not simply as a text generator, but as a “thinking” orchestrator, capable of planning a sequence of actions, calling external tools, interpreting their results, and iteratively adjusting its behavior to achieve a set goal (Dettmers et al., 2023). In the Text- to -SQL task, the agent is able not only to generate SQL, but also to immediately check its correctness through actual execution in the database – and, if necessary, correct errors independently.

To implement the agent system in our work, we use the smolagents framework developed by HuggingFace (Yao et. al., 2023).

The framework features a minimalist design: the agent works through Python code generation rather than through JSON tool call functions, which provides maximum flexibility – the agent can combine the results of several tools in arbitrary expressions.

The fundamental advantage of the agent-based approach over a standard single-pass pipeline is the following: a standard pipeline lacks a verification mechanism—an SQL query may be syntactically correct and execute without errors, but return semantically incorrect data (for example, calculating revenue without accounting for returns). An agent can detect and correct such situations through critical analysis of the results. A comparative analysis of the two approaches is presented in Table 10.

Table 10 – Comparison of the standard Text- to -SQL pipeline and the agent system

Criterion	Standard Text- to -SQL pipeline	Agent system
Generation architecture	Single-pass (query → SQL → result)	Iterative (query → SQL → execution → verification → adjustment)
SQL Error Handling	No: The error is returned to the user.	Automatic: The agent analyzes the error message and generates a corrected request
Verification of the correctness of the result	No: SQL may execute without errors but return invalid data.	Partial: the agent analyzes the result for meaningfulness (number of rows, data types)
Working with multiple tables	Requires a full description of all JOINs in the prompt	The agent independently inspects the schema and builds JOINs through tools
Decomposition of complex issues	No	Yes: The agent breaks down a complex question into substeps
Request cost (tokens)	Low (1 LLM challenge)	High (2-5+ LLM calls per iteration)
Delayed response	Minimum (seconds)	Increased (10-60 sec with several iterations)
Optimal application	Simple queries, high load, real time	Complex analytical queries, batch processing, intolerance to errors

An important feature is automatic error handling in scenarios with incorrect SQL. If, in step 2 of the previous version of the agent (with a less powerful model), a query was generated with an SQLite syntax error —ORDER BY after LIMIT—the agent didn't stop, but read the error message and corrected the query in step 3, removing the LIMIT clause before ORDER BY. This is what distinguishes the agent-based approach: error handling is built into the generation cycle, rather than delegated to the user.

When deployed locally, LLM runs through Ollama, a tool that provides an OpenAI -compatible API for local models. Integration with smolagents is accomplished through the LiteLLMModel adapter:

Basic security recommendations for agent systems. An agent with access to the SQL engine is potentially dangerous: LLM can generate a destructive query (DROP TABLE, DELETE without a WHERE clause). The following precautions must be observed:

1. Use a connection with read-only rights (READ ONLY) for the user on whose behalf the agent runs.

2. Restrict the types of SQL statements allowed (SELECT only) at the tool level – add validation before execution.

3. Set a query execution timeout (statement_timeout) to prevent freezing on heavy queries.

4. Use the max_steps parameter to limit the number of agent iterations and prevent infinite loops.

Thus, the agent-based system based on smolagents provides a qualitatively new level of Text- to -SQL reliability: automatic result verification, iterative self-correction, and the ability to decompose complex queries—features unachievable in single-pass pipelines. The tradeoff—increased cost and iteration latency—is acceptable for analytical tasks that do not require real-time responses.

To clearly display the logic of the agent's operation – including the self-correction cycle – we will construct a block diagram (Figure 1).

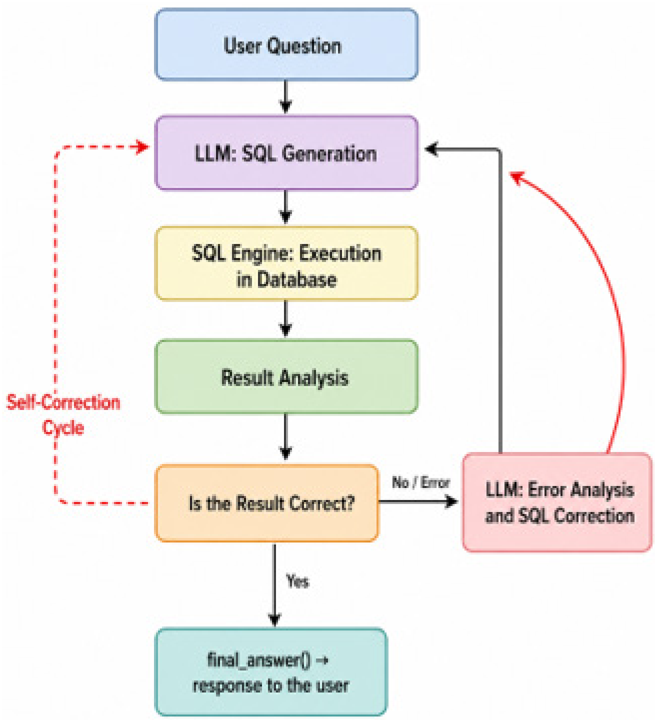

Fig. 1. Workflow Cycle of the Text-to-SQL Agent System

Local agent system deployment configurations. In scenarios where data privacy is important, the agent system can be deployed entirely locally, without forwarding any requests to external servers. Table 11 summarizes the available configurations, from minimal to production.

Table 11 – Configurations for local deployment of the Text- to -SQL agent system

Configuration	LLM	Vector. Storage	Min. requirements	Application
Minimum	Ollama + Llama 3.2 3B	ChromaDB (in-memory)	8GB RAM, no GPU	Demos, prototypes, testing
Basic	Ollama + Llama 3.1 8B	ChromaDB (persistent)	16 GB RAM / 8 GB VRAM	Small teams, private data
Recommended	Ollama + Llama 3.3 70B (4-bit)	Qdrant (persistent)	64 GB RAM / 40 GB VRAM	Corporate analytics service
Maximum	SQLCoder-70B (float16)	Qdrant + pgvector	80+ GB of VRAM (2× A100)	High-load production

Conclusion. The smolagents -based agent system implements a fundamentally different approach to the Text -to -SQL task compared to a single- pass pipeline: iterative SQL generation and verification automatically correct both syntax errors and semantically incorrect queries. A practical implementation with two tables (receipts, waiters) confirmed the functionality of the self-correction mechanism: when an SQLite error occurred, the agent automatically corrected the query without user intervention.

Discussion.

The results demonstrate that the Text- to -SQL problem cannot be effectively solved using a single, universal approach. The analysis confirmed that each of the technologies considered—fine- tuning, RAG, and agent-based systems—has its own area of rational application. The article has already demonstrated that fine- tuning ensures high accuracy with a stable database schema, RAG increases flexibility and knowledge refreshability, and agent-based systems enable iterative validation and automatic error correction of SQL queries.

The most significant result of the study is the justification for a combined approach that uses large language models in conjunction with search-augmented generation and agent-based self-correction. This approach partially overcomes the main limitations of basic LLMs: dependence on prompt quality, lack of knowledge of the specific database schema, and the risk of generating syntactically correct but semantically incorrect SQL queries. Unlike single-pass generation, agent-based architecture includes query execution, result analysis, and error correction, increasing the system's reliability when working with complex analytical queries.

A comparison of approaches shows that fine- tuning is feasible in a stable subject area and with labeled query-SQL pairs. However, in a corporate environment, database schemas frequently change, with new tables, attributes, and business rules added. Under such conditions, retraining the model becomes costly. Therefore, RAG is a more flexible solution: knowledge of the database structure, DDL descriptions, sample SQL queries, and business documentation are stored in an external index and can be quickly updated without changing the model's weights. This is especially important for organizations that require constant updating of analytical data.

However, RAG also has limitations. The quality of the response directly depends on the completeness and quality of the vector index, the document partitioning strategy, the choice of embedding model, and the top -k extraction parameter. If the relevant table or the required SQL example is not included in the context, the model may generate an incomplete or erroneous query. Furthermore, if similar tables or ambiguous column names are present, there is a risk of selecting irrelevant context. Therefore, the effectiveness of a RAG system is determined not only by the quality of the LLM but also by the quality of the knowledge base preparation.

The agent-based approach discussed in this article complements RAG with a self-checking mechanism. The agent-based system not only generates an SQL query but also executes it, analyzes errors, and generates a corrected version of the query. The article notes that the smolagents -based agent system differs from a standard pipeline in that it uses an iterative process: query → SQL → execution → verification → correction. This is especially important for multi-step analytical tasks that require working with multiple tables, JOIN operations, grouping, filtering, and qualifying conditions.

However, the use of agent-based systems comes with increased computational costs and response latency. While standard generation requires a single LLM call, an agent-based cycle can involve multiple iterations, increasing token consumption and processing time. Therefore, the agent-based approach is most appropriate not for simple one-off queries, but for complex analytical scenarios where result reliability is more important than minimal response latency.

Security requires special attention. Since the agent accesses the SQL engine, there is a risk of generating unwanted or potentially dangerous queries. Therefore, the article rightly suggests security measures: using read-only permissions, limiting valid SQL statements to the SELECT command, setting execution timeouts, and limiting the number of agent iterations. These measures are mandatory when implementing Text- to -SQL systems in a corporate environment.

Thus, the study's results confirm that the most promising approach lies not in the isolated use of fine- tuning, RAG, or agent-based systems, but in their integration into a single hybrid architecture. In such an architecture, fine- tuning can provide the model's basic linguistic and SQL competence, RAG provides the relevant context of a specific database, and the agent-based mechanism verifies and corrects the result. This approach improves the adaptability, reliability, and practical applicability of intelligent Text- to -SQL systems in corporate analytics.

Conclusion.

This paper explores and tests methods for using large language models to automate SQL data analysis. All stated objectives were achieved and all tasks outlined in the introduction were solved.

A systemic analysis of the Text- to -SQL problem is conducted: a formal formulation is provided, the evolution of benchmarks (WikiSQL, Spider, BIRD, SQL-Eval) is considered, and the comparative advantages of four solution types are identified. The main conclusion: there is no universally best approach – the choice is determined by the specific project requirements for accuracy, flexibility, cost, and data privacy. The LLM fine-tuning technology is studied in detail as applied to Text -to -SQL; it is shown that the specialized SQLCoder-70B model, with an accuracy of 93 % on SQL- Eval, outperforms GPT-4 (82 %) and Claude 2 (65 %) when working with unfamiliar schemas, demonstrating the fundamental advantage of domain-specific fine- tuning over general LLMs.

RAG technology was explored as an alternative to fine-tuning for dynamic corporate environments. All components of the RAG pipeline were analyzed: document slicing strategies, vector databases (ChromaDB, Qdrant, pgvector), embedding models, and search enhancement techniques (hybrid search, Self-Query, MMR). Practical testing of the Vanna.AI framework on a real dataset was conducted. Orders in BigQuery confirmed the approach's effectiveness for typical business analytics tasks: RFM analysis, YoY comparisons, and ABC-XYZ segmentation. The built-in Auto Fix mechanism demonstrated the system's ability to correct SQL errors without user intervention. It was established that the performance of the RAG system is critically dependent on the volume and quality of the training data in the vector index.

An agent-based text-to-SQL conversion system based on the smolagents framework was developed. A comparative analysis of the agent-based approach and a standard pipeline showed that the agent outperforms single-pass generation in terms of reliability (iterative verification), the ability to handle complex multi-table queries, and automatic error correction, albeit at the expense of higher token costs and response latency.

Based on the conducted research, the following practical recommendations for choosing a paradigm were formulated:

- Fine- tuning (SQLCoder): if you have a stable database schema, high accuracy requirements, and are ready to invest in GPU infrastructure and data labeling.
- RAG (Vanna.AI): with a frequently changing scheme, the need for a corporate chatbot for analysts, a lack of resources for additional training, and a priority for a quick start.
- Agent system (smolagents): for complex multi-step analytical queries requiring decomposition and verification, in batch mode or when response latency is acceptable.
- Combined approach: for maximum quality – RAG on top of pre-trained SQLCoder as LLM backend, with agent-based verification loop.

Overall, the practical significance is determined by the fact that the developed solutions—Vanna.AI configurations, the agent system on smolagents , and local deployment templates—can be directly applied in the analytical departments of organizations without significant adaptation.

The following are promising areas for further research:

- integration of RAG and agent-based approaches into a single pipeline with automatic strategy selection depending on the complexity of the request;
- development of specialized Russian-language training datasets for the Text- to -SQL task;
- study of the application of multi-agent systems, where different agents specialize in different types of SQL queries;
- Evaluation of the impact of DDL documentation quality (completeness of column comments) on the accuracy of RAG systems in real corporate schemes.

REFERENCES

Brown T.B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. - Curran Associates. Vol. 33. Pp. 1877–1901. https://doi.org/10.48550/arXiv.2005.14165 [In Eng.].

Devlin J., Chang M.-W., Lee K., Toutanova K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT // Association for Computational Linguistics. Vol. 1. Pp. 4171–4186. https://doi.org/10.48550/arXiv.1810.04805 [In Eng.].

Dettmers T., Pagnoni A., Holtzman A., Zettlemoyer L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. Advances in Neural Information Processing Systems. - Curran Associates. Vol. 36. https://doi.org/10.48550/arXiv.2305.14314 [In Eng.].

Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L., Chen W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations. — OpenReview. https://doi.org/10.48550/arXiv.2106.09685 [In Eng.].

Li J., Hui B., Qu G., Yang J., Li B., Wang B., Qin B., Cao R., Geng R., Huo N., Zhou X., Ma C., Li G., Chang K.C.C., Huang F., Cheng R., Li Y. (2023). Can LLM Already Serve as A Database Interface? A Big Bench for Large-

Scale Database Grounded Text-to-SQLs. arXiv preprint // Cornell University. arXiv:2305.03111. https://doi.org/10.48550/arXiv.2305.03111 [In Eng.].

Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W.T., Rocktäschel T., Riedel S., Kiela D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems // Curran Associates. Vol. 33. Pp. 9459–9474. https://doi.org/10.48550/arXiv.2005.11401 [In Eng.].

Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W.T., Rocktäschel T., Riedel S., Kiela D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. - Curran Associates. Vol. 33. Pp. 9459–9474. - https://doi.org/10.48550/arXiv.2005.11401 [In Eng.].

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems. Curran Associates. Vol. 30. Pp. 5998–6008. https://doi.org/10.48550/arXiv.1706.03762 [In Eng.].

Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Ma J., Li I., Yao Q., Roman S., Zhang Z., Radev D.R. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing // Association for Computational Linguistics. Pp. 3911–3921. https://doi.org/10.18653/v1/D18-1425 [In Eng.].

Yao S., Zhao J., Yu D., Du N., Shafran I., Narasimhan K., Cao Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models // International Conference on Learning Representations. — OpenReview. https://doi.org/10.48550/arXiv.2210.03629 [In Eng.].

Zhong V., Xiong, C., Socher R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv preprint // Cornell University. arXiv:1709.00103. https://doi.org/10.48550/arXiv.1709.00103 [In Eng.].

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Собственник:
АО «Международный университет информационных технологий» (Казахстан, Алматы)

Главный редактор:
Колесникова Катерина Викторовна

Ответственный редактор:
Мрзабаева Раушан Жалиевна

Компьютерная верстка:
Калабай Замзагуль Ертугановна

Сайт журнала: https://journal.iitu.edu.kz
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)

Подписано в печать 30.06.2026.
050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09).

Издание АО "Международного университета информационных технологий"