УДК 004.891.3

Морозов Д.А., магистр, Самарский национальный исследовательский университет имени академика С. П. Королёва, г. Самара, Россия

УСЛОВИЯ ИЗБЫТОЧНОСТИ АГЕНТНЫХ LLM-АРХИТЕКТУР ПРИ ОРГАНИЗАЦИИ ДОСТУПА К КОРПОРАТИВНЫМ АНАЛИТИЧЕСКИМ ДАННЫМ

Аннотация. Рассматривается выбор архитектуры доступа большой языковой модели к корпоративным данным. Сопоставлены три класса решений: генерация SQL-запросов в реальном времени, агентные системы дополненной генерации с планированием и рефлексией и одношаговая схема с предварительной офлайн-подготовкой показателей. Показано, что надёжность генерации запросов на корпоративных схемах данных существенно ниже, чем на академических наборах: доля успешно решённых задач падает с 91 % до 21 %. Сформулирован формальный критерий избыточности агентной надстройки: если ответ восстановим из результата однократного извлечения, планирование, повторное извлечение и рефлексия не добавляют информации, а лишь увеличивают число обращений к модели. Приведена ресурсная оценка классов архитектур: по опубликованным измерениям агентный контур над генерацией языка запросов даёт примерно пятнадцатикратный рост расхода токенов и шестикратный рост задержки. Определены три условия, при выполнении которых агентная надстройка избыточна, и очерчена область, где она остаётся оправданной.

Ключевые слова: большие языковые модели; агентные системы; RAG; Text-to-SQL; архитектура систем; корпоративная аналитика; поддержка принятия решений.

Abstract. The paper addresses the choice of architecture for large language model access to enterprise data. Three classes of solutions are compared: real-time SQL generation, agentic retrieval-augmented systems with planning and reflection, and a single-pass scheme with offline pre-computation of indicators. Query generation reliability on enterprise schemas is shown to be substantially lower than

on academic datasets, with the task success rate dropping from 91 % to 21 %. A formal redundancy criterion for the agentic layer is formulated: if the answer is recoverable from the result of a single retrieval, then planning, repeated retrieval and reflection add no information and merely increase the number of model calls. A resource assessment of the architecture classes is provided: according to published measurements, an agentic loop over query-language generation yields an approximately fifteenfold increase in token consumption and a sixfold increase in latency. Three conditions under which the agentic layer is redundant are identified, and the area where it remains justified is delineated.

Keywords: large language models; agentic systems; RAG; Text-to-SQL; system architecture; enterprise analytics; decision support.

Введение

Конкурентоспособность предприятия в значительной мере определяется скоростью получения аналитической информации. В логистике, где ежедневно обрабатываются большие объёмы сезонно изменчивых данных о перевозках, складах и маршрутах, это ограничение проявляется особенно остро. Существенным барьером остаётся затруднённый доступ сотрудников без ИТ-квалификации к корпоративным данным: значительная часть отчётности формируется вручную, а сведения разрознены по учётным системам и локальным таблицам.

Опора на электронные таблицы как на инструмент оперативной аналитики создаёт самостоятельный источник риска. Обобщение полевых аудитов показывает, что ошибки обнаруживаются не менее чем в 86 % проверенных рабочих таблиц, при этом разработчики систематически переоценивают их корректность [1]. Разрозненные таблицы порождают расхождение показателей между подразделениями, а платформы бизнес-аналитики ориентированы на заранее заданные сценарии и требуют лицензирования, сопровождения процессов загрузки и привлечения аналитиков.

Технология дополненной генерации (retrieval-augmented generation, RAG) сочетает языковую модель с поиском по внутренним данным, опирая ответ на проверяемые источники [2, 3]. Однако конкретная архитектура такого доступа выбирается сегодня скорее по инерции, чем по анализу задачи: распространённой практикой стало применение агентных схем с планированием, вызовом инструментов и рефлексией к задачам произвольной сложности. Цель настоящей работы — сформулировать критерий, позволяющий определить, оправдана ли агентная надстройка для конкретного класса запросов, и оценить издержки её неоправданного применения.

1. Классы архитектур доступа к структурированным данным

Задачу обращения к структурированным данным на естественном языке традиционно решают методом трансляции запроса в SQL. Применение RAG для подгрузки описаний схем и примеров вида «вопрос — запрос» повышает точность генерации, однако подход наследует ограничения языковых моделей: непрозрачность вывода и склонность к правдоподобным, но неверными построениям. Исполнение сгенерированного запроса над рабочей базой дополнительно повышает операционные риски.

Развитием классической схемы RAG стали продвинутая и модульная архитектуры [2], а с 2024–2025 гг. — агентная (agentic) RAG, в которой автономный агент управляет извлечением как частью цепочки рассуждений [4]. Её типовые компоненты — планировщик, выполняющий декомпозицию запроса; исполнитель с вызовом инструментов, включая SQL-агенты; модуль рассуждения над промежуточными результатами; механизм рефлексии с повторным запуском извлечения, восходящий к схеме Self-RAG [5]; а также память. Концептуальным прообразом служит парадигма чередования рассуждения и действия ReAct [6].

Существенно, что агентные архитектуры не являются избыточными сами по себе. Для многошаговых (multi-hop) задач, требующих синтеза сведений из нескольких источников и разрешения межшаговых зависимостей,

они дают прирост качества, недостижимый одношаговой схемой [4]. Вопрос, рассматриваемый в настоящей работе, состоит не в состоятельности агентного подхода, а в его соразмерности классу решаемых задач.

2. Надёжность генерации запросов на корпоративных схемах

Оценка применимости генерации SQL требует разграничения академических и промышленных условий. На корпусе Spider [7], содержащем базы данных с очищенными схемами, методы на основе современных языковых моделей достигают точности исполнения 91,2 %. Набор BIRD [8], построенный на 95 реальных базах данных из 37 предметных областей и требующий знания содержимого данных, понизил планку: экспертная точность человека составляет 92,96 %, тогда как ведущие системы длительное время оставались в диапазоне 73–75 %.

Наиболее показательны результаты набора Spider 2.0 [9], воспроизводящего корпоративные условия: базы данных содержат свыше тысячи столбцов, размещены в различных СУБД, а решение задачи требует изучения метаданных, документации диалекта и кодовой базы проекта, а также формирования нескольких запросов, нередко превышающих сотню строк. Агентный каркас на базе модели o1-preview решает лишь 21,3 % таких задач против 91,2 % на Spider 1.0 и 73,0 % на BIRD [9]. Отмечается также падение качества на неанглоязычных формулировках запросов, что существенно для российской практики.

Дополнительную осторожность диктует характер самой метрики оценки. Показано, что точность исполнения даёт значительное число ложноположительных и ложноотрицательных срабатываний, а её согласованность с экспертной оценкой без специальных корректировок соответствует каппе Коэна на уровне 0,62 [10]. Публикуемые значения точности, таким образом, отражают качество работы систем лишь приближённо.

Принципиальным для управленческого контура является характер отказа. Ошибка системы генерации запросов не проявляется как сообщение о невозможности выполнить задание: пользователь получает синтаксически корректный ответ с правдоподобным, но неверным числовым значением. В отличие от отказа, который наблюдаем немедленно, такая ошибка обнаруживается постфактум — уже после того, как на её основании принято решение.

3. Формальная постановка и критерий избыточности

Введём обозначения. Пусть B — корпоративная реляционная база данных, S — фиксированный набор аналитических запросов, применяемых на этапе подготовки данных, T = S(B) — результат их выполнения. Обозначим через τ отображение, преобразующее табличные записи в множество аналитических документов D = τ(T), где каждый документ снабжён текстовым представлением и метаданными. Совокупность документов с их индексными представлениями образует базу знаний, размещаемую в векторном хранилище.

Пусть Q — пользовательский запрос на естественном языке, а R — функция извлечения, возвращающая упорядоченное подмножество документов $R(Q) \subseteq D$ мощности k. Ответ системы формируется языковой моделью L как функция запроса и извлечённого контекста:

$$A = L(\, Q, R(Q)\,). \qquad (1)$$

Существенно, что в момент обслуживания запроса аналитические вычисления не производятся: отображение $\tau \circ S$ применяется однократно на офлайн-стадии по цепочке $B \to S \to T \to \tau \to D$, тогда как онлайн-стадия сводится к $Q \to R(Q) \to L \to A$, то есть к одному извлечению и одному вызову модели.

Агентный контур обобщает соотношение (1) до итеративной схемы, в которой запрос декомпозируется на подзапросы, извлечение выполняется

многократно, а результат каждого шага подвергается оценке с возможным повторным запуском:

$$A = L(\, Q, R(Q_1), R(Q_2), \ldots, R(Q_m)\,), \text{ где } \{Q_i\} = \mathrm{Plan}(Q). \qquad (2)$$

Назначение схемы (2) состоит в разрешении межшаговых зависимостей: ситуаций, в которых формулировка последующего извлечения зависит от результата предыдущего. Отсюда следует критерий избыточности агентной надстройки. Если для рассматриваемого класса запросов ответ восстановим из результата однократного извлечения, то есть выполняется условие

$$\forall Q \in \mathcal{Q} : A \text{ определён на } R(Q), \qquad (3)$$

то планирование, повторное извлечение и рефлексия не добавляют информации, поскольку множество доступных модели сведений не расширяется, а лишь дробится на части. Схема (2) в этом случае вырождается в схему (1), сохраняя, однако, все накладные расходы итеративного контура.

Класс запросов, рассматриваемый в настоящей работе, — фактологические обращения к заранее подготовленным показателям — условию (3) удовлетворяет. Ответ на запрос вида «ожидаемый объём груза по филиалу за период» или «сравнение загрузки двух направлений» полностью определяется одним подмножеством документов, поскольку все агрегаты, коэффициенты и статусы вычислены на офлайн-стадии и присутствуют в документах в готовом виде. Иными словами, работа, которую в агентных системах выполняют декомпозиция и повторное извлечение, здесь выполнена однократно и детерминированно — на этапе подготовки данных.

4. Ресурсная оценка классов архитектур

Оценим издержки онлайн-стадии. Обозначим через n_R число раундов извлечения, n_L — число вызовов языковой модели, n_S — число запросов к СУБД, выполняемых в момент обращения пользователя. Предлагаемая схема характеризуется значениями n_R = 1, n_L = 1, n_S = 0. Классическая схема RAG также использует одно извлечение и один вызов модели, но не

располагает предрассчитанной аналитикой. Трансляция в SQL требует генерации и исполнения запроса ($n_S \geq 1$). Агентные архитектуры выполняют несколько раундов извлечения и вызовов модели.

Величина накладных расходов агентного контура установлена независимыми измерениями. Для многошаговых вопросно-ответных задач представительные агентные методы требуют времени вывода, превышающего одношаговую схему в 16–22 раза, причём около 90 % задержки приходится на порождение рассуждений и подзапросов; усреднённая по методам оценка составляет порядка пятнадцатикратного роста задержки [11]. Особенно показательны измерения для агентной генерации языка запросов — задачи, структурно тождественной трансляции в SQL: введение агентного цикла с исполнением и самопроверкой увеличивает среднюю задержку с 4,8 до 32,4 с, а расход токенов — с 1,8 до 27,2 тыс. на запрос, то есть примерно в пятнадцать раз [12]. Существенно, что рост обусловлен преимущественно итерациями вывода модели, а не обращениями к внешнему интерфейсу. Сопоставление классов архитектур приведено в табл. 1.

Таблица 1 — Сравнительная оценка онлайн-стадии по классам архитектур

Характеристика	**Text-to-SQL**	**Агентная RAG**	**Предлагаемая схема**
Извлечений на запрос (n_R)	0–1	несколько (≈2–3)	1
Вызовов модели (n_L)	≥ 1	несколько	1
Запросов к СУБД (n_S)	≥ 1	≥ 0 (через агент)	0
Этапы рассуждения	нет либо один	планирование, рефлексия	нет
Относительный расход токенов	средний	кратный (до ≈15×) [11, 12]	низкий
Относительная задержка	зависит от СУБД	кратная (≈15×) [11, 12]	низкая
Риск ошибок исполнения	ошибки генерации SQL	накопление по шагам	отсутствует
Воспроизводимость	средняя	низкая — средняя	высокая

Из табл. 1 следует, что предлагаемая схема достигает минимума по всем трём параметрам, тогда как трансляция в SQL добавляет исполнение запроса, а агентные архитектуры — кратные извлечения и вызовы модели. Приводимые для агентных систем диапазоны носят ориентировочный характер и служат для сопоставления классов архитектур, а не для прогноза затрат конкретной реализации.

Отдельно следует отметить эффект, не сводимый к затратам. Детерминированность онлайн-стадии определяет воспроизводимость ответа: при неизменных D и Q результат извлечения R(Q) воспроизводится точно, и вариативность ограничена стадией генерации, где она регулируется параметрами декодирования. В агентном контуре недетерминированность планирования распространяется на состав извлекаемого контекста, вследствие чего один и тот же вопрос может получать ответы, опирающиеся на разные подмножества данных. Для управленческой отчётности это самостоятельный недостаток, независимый от стоимости.

5. Условия применимости

Условие (3) выполняется не всегда, поэтому предлагаемая схема не является универсальной заменой ни агентных архитектур, ни трансляции в SQL. Её применимость определяется тремя требованиями к предметной области.

Первое: множество востребованных аналитических срезов конечно и выявляемо на этапе проектирования. Это позволяет заранее определить набор запросов S и материализовать результаты. Второе: данные обновляются по регламенту, а не непрерывно; вычисление ответа в момент обращения не даёт выигрыша в актуальности, поскольку возвращается то же значение, что получено при регламентном расчёте. Третье: допустима задержка актуализации, равная периоду загрузки. Все три требования выполняются для прогнозных и планово-отчётных показателей — прогноза объёмов, плана поступления, коэффициентов загрузки, — но нарушаются для операционных данных реального времени.

Обратные условия очерчивают область, где агентная надстройка оправдана. Это задачи с межшаговыми зависимостями, когда формулировка последующего извлечения определяется результатом предыдущего; задачи межисточникового синтеза, требующие согласования сведений из принципиально различных хранилищ; задачи разрешения противоречий, где необходима оценка достоверности конкурирующих источников. Сопоставление приведено в табл. 2.

Таблица 2 — Соответствие класса задач и архитектуры

Класс задачи	Условие (3)	Рациональная архитектура
Фактологический запрос к предрассчитанному показателю	выполняется	одношаговая схема
Сравнение показателей между объектами и периодами	выполняется	одношаговая схема
Произвольный аналитический срез, не предусмотренный набором S	не выполняется	трансляция в SQL с контролем
Многошаговый вопрос с межшаговой зависимостью	не выполняется	агентная схема
Синтез из разнородных источников, разрешение противоречий	не выполняется	агентная схема

Практическим следствием является не выбор одной архитектуры, а разделение контура обслуживания. Одношаговая схема с предындексацией покрывает регулярные и наиболее массовые сценарии; трансляция в SQL сохраняется для узкого круга аналитиков с обязательным просмотром сформированного запроса; агентные механизмы применяются выборочно, для задач, где условие (3) заведомо нарушено. Такое разделение выносит вероятностный компонент из контура принятия массовых решений и оставляет его там, где ошибка обнаруживается специалистом.

6. Обсуждение

Рассмотренный критерий позволяет переформулировать распространённую постановку вопроса. Обсуждение архитектур доступа к данным обычно ведётся в терминах сравнения качества: какая схема даёт более точные ответы. Предлагаемая формулировка смещает вопрос: следует определить не то, какая архитектура лучше вообще, а то, содержит ли рассматриваемый класс запросов зависимости, ради разрешения которых итеративный контур и создавался. При отрицательном ответе сопоставление по качеству лишено предмета, поскольку обе схемы оперируют одним и тем же множеством доступных сведений.

Следует оговорить, что предлагаемая схема не свободна от собственных отказов. Систематизация практических внедрений RAG выделяет несколько независимых точек сбоя: отсутствие нужного документа в индексе, вытеснение релевантного фрагмента конкурирующими, потерю содержимого на этапе преобразования данных в текст и игнорирование моделью присутствующего в контексте фрагмента [13]. Различие с ошибками генерации запросов состоит не в их отсутствии, а в локализации: перечисленные отказы возникают на этапе подготовки данных и потому поддаются автоматическому контролю до обращения пользователя, тогда как ошибка сгенерированного запроса проявляется в момент ответа.

Ограничением работы является характер ресурсных оценок: приведённые кратности заимствованы из измерений на смежных задачах и не заменяют прямого сравнительного эксперимента в конкретном контуре. Такой эксперимент требует сопоставимой реализации обеих схем на одной модели и едином наборе вопросов. Вопросы построения отображения τ — преобразования табличных данных в документное представление, пригодное для извлечения, — составляют самостоятельную техническую задачу и в настоящей работе не рассматриваются: приведённый критерий формулируется в терминах свойств класса запросов и от способа построения τ не зависит.

Заключение

Сформулирован формальный критерий избыточности агентной надстройки при организации доступа языковой модели к корпоративным данным: если ответ восстановим из результата однократного извлечения, планирование, повторное извлечение и рефлексия не расширяют множество доступных модели сведений и потому не улучшают ответ, сохраняя при этом накладные расходы итеративного контура.

Показано, что для фактологических запросов к регламентно обновляемым показателям критерий выполняется, а издержки неоправданного

применения агентной схемы существенны: по опубликованным измерениям на структурно тождественной задаче генерации языка запросов агентный контур увеличивает расход токенов примерно в пятнадцать раз и задержку — приблизительно всемеро. Дополнительным недостатком является утрата воспроизводимости, обусловленная недетерминированностью планирования.

Определены три условия применимости одношаговой схемы — конечность множества востребованных срезов, регламентный характер обновления данных и допустимость задержки актуализации — и очерчена область, в которой агентные архитектуры сохраняют преимущество. Практическим выводом является разделение контура обслуживания по классам запросов, а не выбор единственной архитектуры.

Список литературы

1. Panko R.R. What We Know About Spreadsheet Errors // Journal of End User Computing. 1998. Vol. 10, No. 2. P. 15–21.

2. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. 2024. arXiv:2312.10997.

3. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 9459–9474.

4. Singh A., Ehtesham A., Kumar S., Khoei T.T. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. 2025. arXiv:2501.09136.

5. Asai A., Wu Z., Wang Y., Sil A., Hajishirzi H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection // Proceedings of the 12th International Conference on Learning Representations (ICLR). 2024.

6. Yao S., Zhao J., Yu D. et al. ReAct: Synergizing Reasoning and Acting in Language Models // Proceedings of the 11th International Conference on Learning Representations (ICLR). 2023.

7. Yu T., Zhang R., Yang K. et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task // Proceedings of EMNLP. 2018. P. 3911–3921.

8. Li J., Hui B., Qu G. et al. Can LLM Already Serve as a Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs // Advances in Neural Information Processing Systems. 2023. arXiv:2305.03111.

9. Lei F., Chen J., Ye Y. et al. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows // Proceedings of the 13th International Conference on Learning Representations (ICLR). 2025. arXiv:2411.07763.

10. Kim H., Jeon T., Choi S. et al. FLEX: Expert-level False-Less EXecution Metric for Reliable Text-to-SQL Benchmark. 2024. arXiv:2409.19014.

11. LatentRAG: Latent Reasoning and Retrieval for Efficient Agentic RAG. 2026. arXiv:2605.06285.

12. Agentic Jackal: Live Execution and Semantic Value Grounding for Text-to-JQL. 2026. arXiv:2604.09470.

13. Barnett S., Kurniawan S., Thudumu S. et al. Seven Failure Points When Engineering a Retrieval Augmented Generation System. 2024. arXiv:2401.05856.