SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Retrieval-Augmented Generation for Natural-Language Access to Enterprise Operational Data

Ashwin Krishnappa Kumar

Senior Manager of Data Engineering and Analytics, Tesla, USA. ORCID: 0009-0007-7020-4227

Abstract
Enterprise data warehouses hold answers to countless operational questions, but for most employees those answers remain locked behind SQL syntax and schema knowledge they were never trained to acquire. Fixed dashboards address recurring questions well, but every question outside a dashboard's anticipated scope routes back to a data team, creating delays that discourage exploratory questioning altogether. Retrieval-augmented generation offers a categorically different access model: language models paired with a retrieval step grounded in an organization's actual schema and data can answer novel operational questions in natural language, closing a gap that ungrounded models cannot close at all. This article examines the production-engineering decisions, retrieval architecture, context management, execution guardrails, governance enforcement, and continuous evaluation discipline that determine whether such a system is trustworthy enough for operational decision-making and argues that the performance ceiling of a conversational data agent is set as much by the underlying warehouse's architecture as by the language model itself. The analysis synthesizes recent empirical evidence on retrieval-augmented text-to-SQL generation with established governance and evaluation frameworks and finds that production readiness in this domain is fundamentally a systems-engineering achievement rather than a model-selection decision.

Keywords: Retrieval-Augmented Generation, Enterprise Data Access, Natural-Language Query, Vector Embeddings, Text-to-SQL, Conversational AI Governance

1. Introduction

Enterprise operational data accumulates across production execution systems, quality platforms, and equipment telemetry streams, but the ability to ask a question of that data has historically required either specialized SQL fluency or a pre-built dashboard anticipating that exact question. Fixed dashboards work well for recurring questions such as daily throughput or weekly scrap cost, since a data team can build a visualization once and let it serve every future instance of the same query. Operational environments, however, constantly generate novel questions that fall outside any dashboard's anticipated scope: why yield dropped on a specific line last Tuesday or how one facility's labor efficiency compares to another's over a quarter with a specific product mix excluded. Answering these ad hoc questions requires writing new SQL queries, a skill most operational staff neither have nor should be expected to acquire, and the resulting bottleneck, non-technical staff submitting requests and waiting on a data team's queue, does not merely slow individual decisions; it discourages exploratory questioning altogether once employees learn that anything beyond the standard dashboard is not worth the wait.

Large language models paired with retrieval-augmented generation offer a different model of access. Retrieval-augmented generation was proposed as a general mechanism for grounding a language model's output in context retrieved from an external source at query time rather than relying solely on knowledge encoded during training [1], and the retrieval step itself is most commonly implemented today using dense vector representations retrieved via approximate nearest-neighbor search, an approach shown to substantially outperform sparse lexical matching on open-domain question answering tasks [2]. Applied to an enterprise warehouse, the same mechanism allows any employee to ask an operational question in natural language and receive an answer traceable to the organization's own schema and records, rather than to whichever question a dashboard's designer anticipated in advance. The gap the approach closes is not incremental. Recent evaluation of large language models on enterprise-style SQL generation tasks found exact-match accuracy at 0% across all tested tasks when no retrieval augmentation was applied, even using state-of-the-art models [3]; execution accuracy climbed to as high as 82.2% once a retrieval-augmented framework was introduced [3].

This gap is a categorical difference, not a matter of degree, and it is the starting point for treating retrieval-augmented generation as infrastructure rather than as an experimental convenience.

Closing that gap in a benchmark setting, however, is a different achievement from closing it reliably in a production enterprise environment where a wrong answer can drive a real operational decision. The benchmark literature that establishes the accuracy figures cited above is itself built on standardized cross-domain evaluation datasets designed to test generalization across many independent database schemas rather than performance on a single, memorized one [4], and the gap between accuracy on such a benchmark and reliability inside one organization's specific warehouse, with its schema quirks, naming conventions, and query patterns, is precisely where most of the engineering effort in a real deployment goes. This article examines the engineering decisions, spanning schema retrieval, conversational context management, pre-execution guardrails, access governance, and continuous evaluation, that determine whether a retrieval-augmented conversational agent is trustworthy enough for that environment and argues throughout that the ceiling on what such a system can reliably achieve is set jointly by the language model and by the architecture of the warehouse it queries.

2. Related Work

2.1 The Limits of Traditional Dashboards and SQL Literacy

The structural constraint of dashboard-based business intelligence is that a dashboard's designer must anticipate a question before a user can ask it. For frequently recurring questions this model is efficient, but literature on the adoption of self-service business intelligence has repeatedly documented that non-technical users continue to depend on IT and analyst intermediaries for anything beyond the pre-built layer, undermining the self-service promise the tooling was meant to deliver [5]. This dependency is not simply a training gap; reviews of the literature on self-service business intelligence describe it as a persistent structural pattern across implementations, in which data democratization efforts repeatedly stall at the same point: users can explore within a designed scope but cannot extend that scope themselves [5]. The result is the same bottleneck described above, a data team's request queue functioning as the actual constraint on how quickly an organization can answer its questions about itself.

The pattern documented in this literature is worth taking seriously precisely because it recurs across implementations that vary considerably in tooling, industry, and organizational maturity. If the constraint were simply a matter of buying a better dashboard product or training more employees in basic query-building, the literature would show that better-resourced organizations had solved the problem; instead, the reviewed evidence suggests the constraint is closer to structural, following from the dashboard model's own design assumption that the space of relevant questions can be enumerated in advance [5]. That assumption is precisely what a retrieval-augmented conversational interface does not require, since it answers whatever question is actually asked rather than whichever question was anticipated.

2.2 Retrieval-Augmented Generation as a Bridge

The general retrieval-augmented mechanism introduced above [1], most commonly implemented through dense vector retrieval [2], was not designed with enterprise schemas specifically in mind, but it addresses a gap those schemas make acute: table names, column semantics, and business-specific abbreviations are not part of any language model's general training data, so the model has no way to produce a correct query without some mechanism supplying that missing context at query time.

The magnitude of that gap, and of retrieval's effect on it, has been measured directly. Beyond the near-total failure of ungrounded generation described in Section 1, comparative work evaluating retrieval-augmented schema selection against fixed top-k retrieval found that a dynamically clustered retrieval approach improved SQL generation execution accuracy from a baseline of 0.70 to 0.78 using the same underlying language model, with schema-retrieval F1 improving from 0.79 to 0.88 over the same comparison [6]. That the retrieval strategy itself, independent of any change to the generating model, produced this improvement is direct evidence that the choice of retrieval strategy is a significant factor in output quality, not an implementation detail.

This finding also reframes what "improving the model" should mean in a production context. An organization facing disappointing accuracy from a conversational data agent has at least two distinct levers available: upgrading the underlying language model, which is often the more visible and more heavily marketed option, or re-engineering the retrieval strategy feeding that model, which the evidence above suggests can move accuracy by a comparable margin at a fraction of the cost and disruption of a model migration. Treating these as substitutable levers, rather than assuming the language model itself is always the binding constraint, is one of the more actionable implications of the retrieval-versus-generation literature for a team operating under a fixed budget.

3. Architecting RAG for Enterprise Data

3.1 Vector Embeddings and Semantic Schema Retrieval

The retrieval half of a production RAG system depends on representing schema elements, documentation, and example queries as vector embeddings and storing them in a vector index searchable by semantic similarity rather than exact match [2]. This allows a system to correctly retrieve relevant schema context even when a user's phrasing does not match the underlying column or table names, which is the norm rather than the exception in natural conversational queries. As already established in Section 2.2, this retrieval step is not a fixed, one-size-fits-all component: the comparison between static top-k selection and adaptive, dynamically clustered retrieval produced a measurable accuracy gain from retrieval design alone [6], and researchers evaluating a unified framework for grounding language models directly against database contents have argued that treating retrieval and generation as two cleanly separable stages understates how much the interaction between them determines end-to-end reliability [7]. The practical implication for an enterprise deployment is that retrieval scope and retrieval strategy deserve the same design attention typically reserved for model selection, since the evidence indicates they can move accuracy by a similar order of magnitude.

That framework-level argument has a direct bearing on how an enterprise team should think about the boundary between the database and the language model sitting above it. Rather than treating the database purely as a passive store that returns rows to whatever query the model happens to generate, a system designed with the database's own capabilities in mind, its indexes, its materialized views, its query optimizer, can shift some of the reasoning burden away from the language model and onto infrastructure that is far more reliable at that specific job [7]. This is a modest but important reframing: the conversational layer's job is not to compensate for everything the underlying data platform does not already do well, but to sit atop a platform that has been designed, as far as possible, to make the generation task easier in the first place.

3.2 Conversational Context and Multi-Turn Follow-Up

A single-turn question-and-answer model is insufficient for real operational use, where users naturally ask follow-up questions that depend on a prior exchange, such as asking to break a just-returned figure down by shift after first asking for a daily total. One practical approach carries the prior turns of a conversation forward directly within the language model's context window, letting the model resolve references such as "that" or "the same period" using its language understanding rather than a custom slot-filling system. This approach's simplicity is also its limitation: research examining how language models use long input contexts has found that model performance on information located in the middle of a long context degrades measurably relative to information located near the beginning or end, a pattern that persists across model families and context lengths [8]. For a conversational data agent, this means that simply extending the context window as a conversation grows is not a neutral operation; beyond some point, carrying the full conversation forward risks degrading the model's ability to correctly use the specific turn a follow-up question actually depends on, which is a different and more subtle failure mode than running out of context space outright.

This distinction matters operationally because the two failure modes call for different mitigations. Running out of context space is a clear, detectable error: the system can catch it and truncate or summarize before the request fails outright. Degraded use of mid-context information is a silent failure: the request completes, an answer is returned, and nothing in the system's own telemetry flags that the answer relied on a misread of an earlier turn. Systems built this way need an explicit policy for when to summarize or truncate earlier turns, not only to stay within a hard context limit but also to keep the turns that a follow-up question is likely to depend on positioned where the evidence indicates a model handles them most reliably. Without such a policy, the failure mode is not a hard error but a quiet degradation in the correctness of ordinary follow-up questions that no error log will ever surface.

3.3 Retrieval Against Warehouse Architecture

The retrieval and generation techniques described above are not backend-agnostic; the design of the underlying warehouse materially shapes how difficult the text-to-SQL problem actually is for a given question. Enterprise deployments typically limit retrieval to a curated subset of an organization's full schema, because exposing every table in a large analytical warehouse to the retrieval step would both reduce retrieval precision and increase the attack surface for the guardrail failures discussed in Section 4.1; a sharded, columnar architecture chosen in part for its compression and query-latency characteristics is a common foundation for that warehouse layer [9]. That warehouse's underlying storage format choices are not incidental to the conversational layer sitting above them: Empirical evaluation of columnar storage formats has found that format and compression-scheme mismatches can impose measurable scan-time overhead independent of query complexity [10], meaning a warehouse whose storage layer is poorly matched to its query patterns

imposes a performance ceiling that no amount of retrieval or prompt engineering built on top of it can fully recover. A generated query that is syntactically and semantically correct can still return slowly, or time out under the guardrails described in Section 4.1, purely because the storage layer beneath it was not designed for the access pattern the conversational agent's users actually generate.

Schema design choices carry a further, more direct effect on generation correctness. It might seem, at first inspection, that heavily denormalized, wide tables would straightforwardly simplify the model's task by collapsing what would otherwise be a web of foreign-key joins across many narrow, normalized tables into a small number of self-contained tables. The actual evidence is more qualified than that intuition suggests. A systematic evaluation of database normalization effects on SQL generation across eight large language models found that denormalized schemas did outperform normalized schemas on simple retrieval-style queries, including in zero-shot settings with smaller models, but that normalized schemas following standard second- and third-normal-form design performed better on aggregation queries specifically because denormalization's data duplication introduced its own error mode: aggregation over a denormalized table can silently over-count or under-count values that were duplicated across the collapsed rows, an error a model has no way to detect from the schema alone [11]. The same study found that normalized schemas' more common failure mode, incorrect join prediction, was substantially mitigated by including a small number of few-shot examples in the prompt, while denormalization's aggregation-correctness risk was not similarly resolved by prompting alone [11].

The practical implication is that schema design for a conversational agent is not a matter of denormalizing everything in sight; it is a matter of matching schema shape to the query types the system is actually expected to answer and treating the resulting normalized-versus-denormalized tradeoff as query-type-dependent rather than settled in either direction. An enterprise whose conversational agent is expected to field mostly simple, filtered lookups, like a specific machine's status on a specific shift, benefits from the denormalized design the companion warehouse analysis describes [9]; an enterprise whose users lean more heavily on aggregation-style questions, comparing efficiency across facilities over a quarter, should weigh that same denormalization against the aggregation-correctness risk the normalization literature identifies [11] and may find that a hybrid schema, denormalized for the tables feeding the most common lookup questions and normalized where aggregation-heavy reporting dominates, better matches its actual query distribution than a uniform choice in either direction.

3.4 Where Retrieval Ends and Execution Begins

A further architectural choice, distinct from schema shape itself, is where the boundary sits between retrieving the right schema context and executing the resulting query. Treating text-to-SQL generation as a standalone task, correct once the model produces a syntactically valid query, understates what a production system actually needs, since some operational questions cannot be answered by any single SQL query at all: a question that requires the model to reason over intermediate results, apply a threshold, and then issue a follow-up query conditioned on that threshold falls outside what one-shot query generation was designed to do. Framing table-augmented generation as a unified problem spanning retrieval, query synthesis, and downstream reasoning over the result set, rather than as three independently optimized stages, has been argued to close a category of failures that neither better retrieval nor better SQL generation alone can fix, because the failure originates in the boundary between the stages rather than within any single one [8]. For an enterprise conversational agent, this means the architecture question is not only which tables to retrieve or which schema shape to use, but whether the pipeline treats query execution as the final step or as one intermediate step among several that a genuinely complex operational question may require.

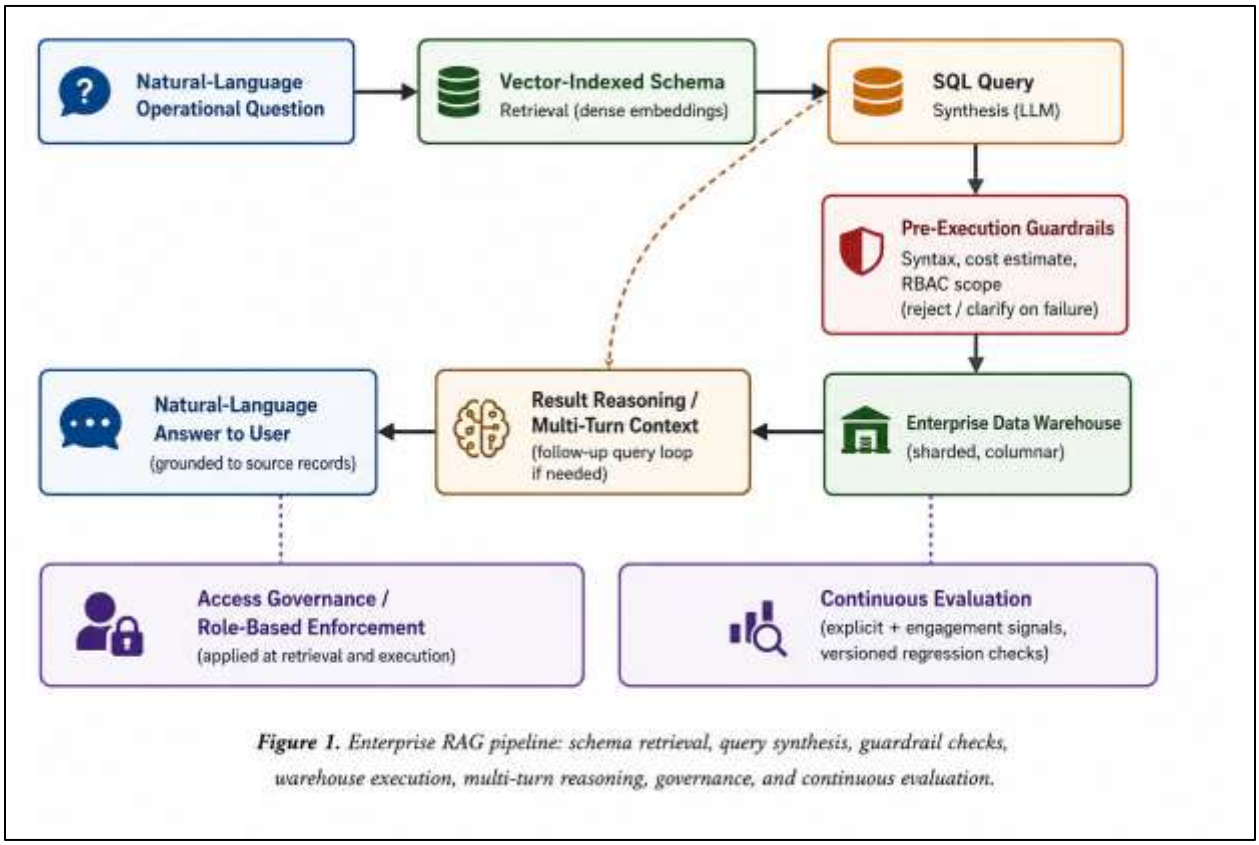

Figure 1. Enterprise RAG pipeline: schema retrieval, query synthesis, guardrail checks, warehouse execution, multi-turn reasoning, governance, and continuous evaluation.

4. Results and Discussion

4.1 Guardrails Before Execution: Timeouts and Query Validation

As the language model generates executable SQL directly, a production system cannot treat that output as safe to run without independent validation. Cost models for query optimization, long treated skeptically in the database systems literature as too unreliable for practical use, have been shown to be considerably more useful for predicting execution cost and catching pathological queries than that reputation suggests, provided the model is properly calibrated against the actual workload it will see [12]. Running a generated query through this kind of plan-validation step before execution, paired with a database-level timeout that terminates any query exceeding a bounded runtime regardless of what the plan estimate predicted, forms a lightweight but layered safety net: the plan-validation step catches structurally malformed or unexpectedly expensive queries before they run, and the timeout catches the residual case where a query's actual runtime diverges from its estimated cost. Neither guardrail depends on the language model behaving correctly, which is a deliberate and important design property: a safety mechanism that trusted the same model whose output it exists to check would not be a safety mechanism at all.

The calibration point deserves emphasis, since it is particularly tempting to treat cost-model skepticism as a settled conclusion rather than a workload-dependent finding. The evidence that optimizer cost models are more usable than their reputation suggests was specifically conditioned on calibrating the model against the workload it would actually see [12]; a plan-validation guardrail deployed against a generic, uncalibrated cost model inherits exactly the unreliability the earlier skepticism was responding to. For a conversational agent whose generated queries may follow patterns quite different from the hand-written queries a warehouse's optimizer was originally tuned against, this means the guardrail itself requires periodic recalibration as the population of agent-generated queries evolves, not a one-time setup step treated as permanently sufficient.

Table 1. Guardrail mechanisms and the failure modes each addresses.

Guardrail Mechanism	Failure Mode Addressed	Source

Calibrated cost-model plan validation	Structurally malformed or pathologically expensive generated queries	[12]
Database-level execution timeout	Queries whose actual runtime diverges from estimated cost	[12]
Execution-time role-based access binding	Access-control bypass via a conversational path rather than the dashboard interface	[13]
Explicit + implicit feedback monitoring	Silent quality regressions not caught by a one-time accuracy benchmark	[14],[15]
Versioned, benchmark-based regression evaluation	A change that improves one capability while silently degrading another	[11],[6],[3]

4.2 Enforcing Role-Based Access Within AI Responses

A conversational interface to enterprise data introduces a governance requirement that traditional dashboards handle more simply: because the model can, in principle, generate a query against any table it has retrieval access to, access control has to be enforced at the query execution layer rather than only at the application layer. Role-based access control was formalized specifically to bind a user's permissions to a defined role rather than to an application's front-end logic, on the reasoning that access decisions enforced only at the interface can be bypassed by any path that reaches the underlying resource directly [13]. A conversational agent is exactly this kind of alternate path: if the agent's own database credentials are not bound to the requesting user's existing role-based permissions, the same access rules that govern dashboard visibility can be silently circumvented by rephrasing a question rather than by any deliberate attack.

Binding agent credentials to the requesting user's role at execution time, so the same rules governing dashboard access govern what the generated query is permitted to return, closes that path without requiring the language model itself to reliably refuse an improperly scoped request, which experience with prompt-based restrictions elsewhere suggests it cannot be counted on to do consistently. This design choice also has a useful side effect for the retrieval architecture described in Section 3.1: because access control is enforced at execution rather than at retrieval, the retrieval scope itself does not need to be individually recomputed per user role, simplifying the retrieval layer considerably at the cost of occasionally retrieving schema context for a table the user's eventual query will be denied against. That tradeoff, simpler retrieval against a small amount of wasted retrieval effort on denied paths, is generally favorable, since the execution-layer check is the one that actually has to be airtight, while an imperfect retrieval-layer optimization only affects efficiency, not correctness of the access decision itself.

4.3 Measuring Quality: Explicit and Implicit Feedback Signals

Evaluating whether a conversational data agent is actually working well in production requires more than a one-time accuracy benchmark; it requires an ongoing feedback mechanism, and the choice of feedback signal itself has real consequences for what a monitoring system can and cannot see. Research comparing explicit user ratings against behavioral engagement signals in a field deployment of a machine-learning-based recommender system found that the two signal types frequently diverged, with engagement-based measures sometimes indicating satisfaction that explicit ratings did not corroborate, and vice versa [14]. This divergence matters for a conversational data agent because explicit feedback is known to undercount dissatisfaction broadly: most users who receive a poor answer simply move on rather than taking the extra step of flagging it.

A complementary behavioral signal, drawn from research on how query reformulation predicts search satisfaction in web search, treats a user re-issuing a similar or rephrased version of the same question within a short window as a strong implicit indicator that the prior answer did not satisfy their need [15]. Treating this kind of repeated, semantically similar querying as a proxy for dissatisfaction, rather than requiring an explicit flag, substantially widens the population of signal available for quality monitoring beyond what an opt-in feedback button alone would ever capture. The divergence documented between explicit and engagement-based signals [14] suggests a further design implication beyond simply adding a second signal: a monitoring system that only reconciles disagreements between explicit and implicit signals after the fact will miss cases where both signals agree but are jointly wrong, for instance when a user is satisfied with an answer that is nonetheless factually incorrect in a way neither rating behavior nor reformulation behavior would surface. Neither feedback channel is a substitute for the offline evaluation discipline described in Section 4.6; both are necessary complements to it, not replacements.

4.4 Rule-Guided Visualization Selection

Generating a correct answer is only half the task; presenting it in a useful visual form is what makes the answer immediately actionable. Machine-learning approaches to visualization recommendation have demonstrated that chart-type selection can be modeled as a predictable function of a small number of data characteristics,

including cardinality, data type, and the statistical distribution of the values being visualized [16]. Rather than leaving chart-type selection entirely to a language model's case-by-case judgment, production systems can apply this same principle through explicit examples that map metric characteristics to preferred chart types: whole-number counts to bar charts, percentage-based metrics to line graphs that emphasize trend over time, and percentile-based metrics to box plots that communicate distribution and spread in a way a single bar or line cannot.

When a metric does not clearly match one of these patterns, defaulting to a bar chart as the safest general-purpose choice keeps the failure mode a mild one rather than a misleading visualization. This example-driven approach combines a language model's flexibility for a metric it has not seen a specific rule for with the consistency of a rules-based system for the common cases that make up the majority of queries, and it preserves the user's ability to override the default by explicitly specifying a chart type in the question itself. The underlying finding that chart selection is a learnable function of data characteristics rather than a matter of pure aesthetic judgment [16] is also what makes an example-driven, rather than purely model-judged, approach defensible in the first place: if chart-type appropriateness were not a reasonably predictable function of the data itself, a fixed set of examples would not generalize nearly as well across the range of metrics an enterprise conversational agent actually encounters.

Table 2. Metric characteristics mapped to recommended chart/reporting types [16]

Metric Characteristic	Recommended Chart / Reporting Type
Whole-number counts	Bar chart
Percentage-based metrics (trend over time)	Line chart
Percentile-based / distributional metrics	Box plot
No clear match to the above patterns	Bar chart (safe general-purpose default)

4.5 Deployment Architecture: Streaming, Cost, and Latency at Scale

Every stage of the pipeline described above, embedding the incoming question, retrieving relevant schema context, generating a query, validating it, executing it, and generating a natural-language summary, adds latency and, for any language-model call, direct compute cost. Recent survey work on large language model inference engines documents that batching, key-value caching, and speculative decoding techniques can each independently reduce per-request latency and cost at scale, but that no single technique dominates across all deployment patterns, meaning the right combination depends on request volume, concurrency, and the acceptable latency ceiling for a given application [17]. For a conversational data agent operating at a meaningful production scale, with many daily active users each triggering multiple language-model calls per exchange, the cumulative cost of these calls becomes a real operational expense that has to be actively managed rather than treated as unbounded.

Practical mitigations available at the application layer, independent of the inference-engine-level techniques surveyed above, include caching embeddings for frequently retrieved schema fragments, caching full responses for identical or near-identical repeated questions within a short time window, and constraining retrieval scope so that each request's prompt stays as small as it can while still containing the context an accurate answer requires; because prompt size directly drives both latency and per-request cost, disciplined scope management functions as a cost-control lever, not merely a quality safeguard. These application-layer mitigations and the inference-engine-level techniques surveyed above are complementary rather than competing: an organization can adopt caching and scope discipline regardless of which inference-serving stack it runs underneath, and the gains from each layer are largely additive rather than redundant, since one governs how much work the pipeline has to do per request and the other governs how efficiently the underlying model executes whatever work remains.

4.6 Continuous Evaluation and the Warehouse-Architecture Ceiling

As the underlying language model, retrieval index, and prompt structure all continue to evolve after initial launch, a production system needs a repeatable way to confirm that a change intended to improve one capability has not silently degraded another. This discipline echoes what has already been established at the storage layer in Section 3.3: Just as a warehouse's storage format and schema design set a ceiling on what the conversational layer above it can achieve regardless of how well that layer is engineered, a conversational system's evaluation practice sets a ceiling on how confidently an organization can make further changes to prompts, retrieval configuration, or the underlying model without risking an undetected regression.

The parallel between these two ceilings runs deeper than an analogy. In both cases, the organizations most likely to discover the ceiling the hard way are the ones that invested heavily in one layer while treating the other as settled. A team that carefully tunes retrieval strategy and prompt design but never revisits its warehouse's schema and storage choices will eventually find that further conversational-layer improvements produce diminishing returns, not because the improvements are poorly engineered but because the storage layer beneath them cannot deliver the query performance those improvements assume [10], [9]. Symmetrically, a team that gets the warehouse architecture right but never establishes versioned, benchmark-based evaluation of its prompts and retrieval configuration will eventually ship a regression it cannot detect until users notice, at which point the feedback signals described in Section 4.3 are already reporting a problem the team has no systematic way to trace back to its actual cause. The two ceilings compound: an enterprise that has matched its warehouse architecture to its query patterns, per the design tradeoffs discussed in Section 3.3, and paired that with disciplined, versioned evaluation of every subsequent change to the conversational layer is positioned to capture the continuing performance gains that the empirical literature shows are actively being made in this space [3], [6], [11], while an enterprise that gets either layer wrong will find that improvements at the other layer arrive against a ceiling it cannot see until something breaks in production.

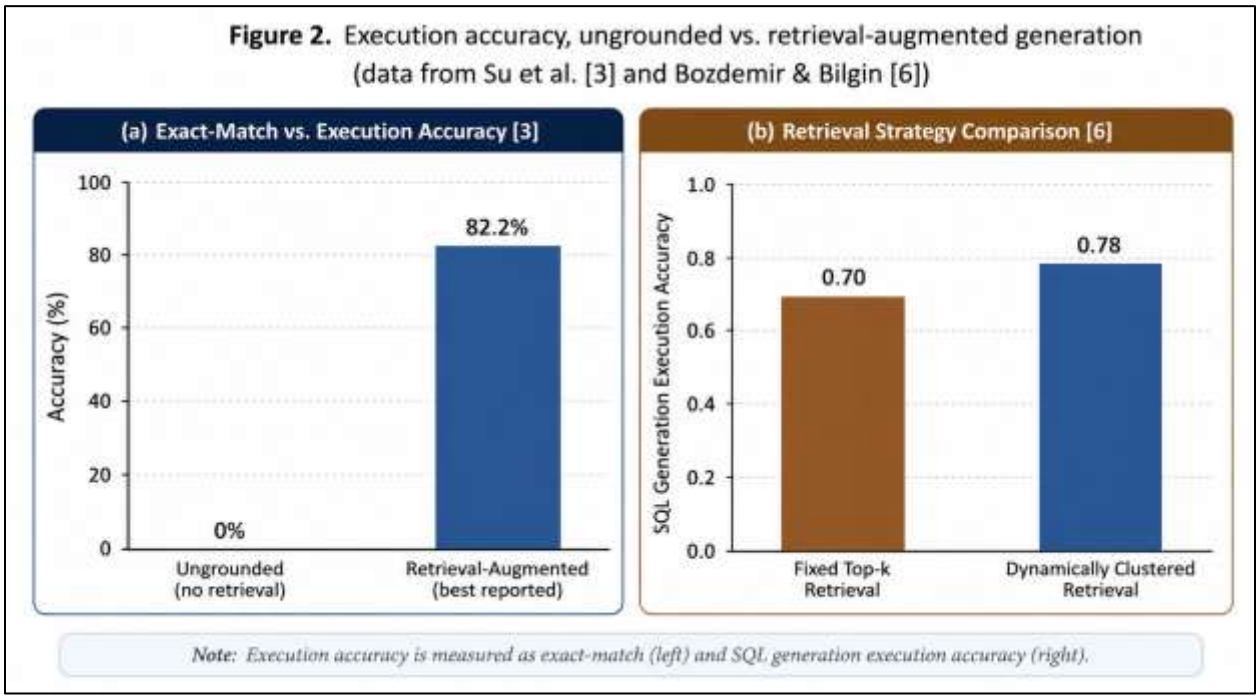

Figure 2. Execution accuracy, ungrounded vs. retrieval-augmented generation (Su et al. [3]; Bozdemir & Bilgin [6]).

This same principle of avoiding single-layer optimization has an organizational parallel outside the immediate technical stack. Case-study literature on model-based analytics-as-a-service frameworks in manufacturing settings has found that the recurring limiting factor for realized value from a data platform is less the sophistication of any individual analytical technique and more the integration discipline that connects disparate systems into one coherent, governed platform [18]. A conversational data agent is, in this sense, one more analytical service that addresses that same integration challenge, and its long-run value depends on the same platform-level governance and evaluation discipline that determines whether any other analytical service built on the same warehouse succeeds or stalls.

5. Conclusion

Retrieval-augmented generation closes a gap that no amount of model scale alone can close: the near-total failure of ungrounded language models to generate enterprise-style queries with execution accuracy, which recent evaluations show can reach the low eighties once retrieval is properly designed. This article has argued that the distance between that benchmark result and a production system that enterprise employees can actually trust is closed by systems engineering, not by model selection: retrieval strategy and scope,

conversational context management, pre-execution guardrails, access governance bound to existing role-based permissions, and continuous, versioned evaluation each contribute independently to whether a conversational data agent is trustworthy enough for operational use. The evidence further indicates that this conversational layer cannot be designed in isolation from the warehouse architecture beneath it; storage format, schema normalization choices, and query-pattern alignment set a performance ceiling that no amount of retrieval or prompt engineering can fully overcome.

This analysis has limits. It synthesizes published empirical evaluations and governance frameworks rather than an original, instrumented production deployment measured directly by the author; the specific accuracy figures cited here reflect the benchmarks, models, and schema conditions described in their original sources, which may not transfer uniformly to every enterprise environment. The cross-domain benchmarks underlying the headline accuracy figures are themselves built to test generalization across many independent schemas, which is a different and in some ways harder test than performance on one organization's specific, long-lived warehouse, so the figures cited here should be read as an evidenced floor for what careful engineering can achieve, not a guarantee that any specific deployment will reach them without its calibration effort.

Future work applying this production-readiness framework to an original, instrumented enterprise deployment and studies that directly measure how much of an organization's ad hoc question volume shifts from data-team queues to self-service once a well-governed conversational agent is deployed would extend the evidence base beyond the comparative synthesis presented here. A further direction, suggested directly by the warehouse-architecture ceiling discussed in Section 4.6, would examine empirically how much of a conversational data agent's realized accuracy in a given deployment is attributable to the language model and retrieval configuration rather than the underlying warehouse's own schema and storage design, a decomposition that the case-study and benchmark literature synthesized here does not yet resolve.

Author Contributions

Conceptualization, Methodology, Investigation, Writing – Original Draft, Writing – Review & Editing, Visualization: A.K. The author has read and agreed to the published version of the manuscript.

Funding

The author declares that no external funding was received for this work.

Conflicts of Interest

The author declares no conflict of interest.

Statement of Human and Animal Rights

This study did not involve human participants, animal subjects, or identifiable personal data requiring ethics board review.

Informed Consent

Not applicable.

Data Availability Statement

This article synthesizes previously published data and findings, cited throughout. No new datasets were generated or analyzed for this article.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), 2020. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[2] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih, "Dense Passage Retrieval for Open-Domain Question Answering," Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. Available: https://aclanthology.org/2020.emnlp-main.550/

[3] Xiaodong Su, Yang Gu, Peng Wang, Wei Gu, Lincheng Qi, and Jingwei He, "A Robust Natural Language Text-to-SQL Generation Framework with Dynamic Strategies Based on LLMs," Scientific Reports, 2026. Available: https://doi.org/10.1038/s41598-026-39128-9

[4] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev, "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 (EMNLP), 2018. Available: https://doi.org/10.18653/v1/D18-1425

[5] Ramalakshmi Murugan and Swetha Sekhar, "A Literature Review on Data Democratization, Self-Service Business Intelligence, and Data Literacy," International Journal of Business Analytics, 12(1), 2025. Available: https://doi.org/10.4018/IJBAN.398838

[6] Mehmet Bozdemir and Metin Bilgin, "Schema Retrieval with Embeddings and Vector Stores Using Retrieval-Augmented Generation and LLM-Based SQL Query Generation," Applied Sciences, 16(2), 2026. Available: https://doi.org/10.3390/app16020586

[7] Asim Biswal et al., "Text2SQL Is Not Enough: Unifying AI and Databases with TAG," Conference on Innovative Data Systems Research (CIDR), 2025. Available: https://vldb.org/cidrdb/papers/2025/p11-biswal.pdf

[8] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang, "Lost in the Middle: How Language Models Use Long Contexts," Transactions of the Association for Computational Linguistics, 12, 2024. Available: https://doi.org/10.1162/tacl_a_00638

[9] Robert Schulze, Tom Schreiber, Ilya Yatsishin, Ryadh Dahimene, and Alexey Milovidov, "ClickHouse: Lightning Fast Analytics for Everyone," Proceedings of the VLDB Endowment, 17(12), 2024. Available: https://doi.org/10.14778/3685800.3685802

[10] Xinyu Zeng, Yulong Hui, Jiahong Shen, Andrew Pavlo, Wes McKinney, and Huanchen Zhang, "An Empirical Evaluation of Columnar Storage Formats," Proceedings of the VLDB Endowment, 17(2), 2023. Available: https://doi.org/10.14778/3626292.3626298

[11] Ryosuke Kohita, "Exploring Database Normalization Effects on SQL Generation," Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM), 2025. Available: https://doi.org/10.1145/3746252.3761583

[12] Wentao Wu, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Hakan Hacıgümüş, and Jeffrey F. Naughton, "Predicting Query Execution Time: Are Optimizer Cost Models Really Unusable?," 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013. Available: https://ieeexplore.ieee.org/document/6544899/

[13] Ravi S. Sandhu and Edward J. Coyne, "Role-Based Access Control Models," IEEE Computer, 29(2), 1996. Available: https://doi.org/10.1109/2.485845

[14] Qian Zhao, F. Maxwell Harper, Gediminas Adomavicius, and Joseph A. Konstan, "Explicit or Implicit Feedback? Engagement or Satisfaction? A Field Experiment on Machine-Learning-Based Recommender Systems," Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC), 2018. Available: https://doi.org/10.1145/3167132.3167275

[15] Ahmed Hassan Awadallah, Xiaolin Shi, Nick Craswell, and Bill Ramsey, "Beyond Clicks: Query Reformulation as a Predictor of Search Satisfaction," Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM), 2013. Available: https://doi.org/10.1145/2505515.2505682

[16] Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo, "VizML: A Machine Learning Approach to Visualization Recommendation," Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019. Available: https://doi.org/10.1145/3290605.3300358

[17] Sihyeong Park, Sungryeol Jeon, Chaelyn Lee, Seokhun Jeon, Byung-Soo Kim, and Jemin Lee, "A Survey on Inference Engines for Large Language Models: Perspectives on Optimization and Efficiency," ACM Transactions on Intelligent Systems and Technology, 2025. Available: https://doi.org/10.1145/3803798

[18] Angelo Corallo, Anna Maria Crespino, Mariangela Lazoi, and Marianna Lezzi, "Model-Based Big Data Analytics-as-a-Service Framework in Smart Manufacturing: A Case Study," Robotics and Computer-Integrated Manufacturing, 76, 2022. Available: https://doi.org/10.1016/j.rcim.2022.102331