



AI-Ready Data Products: Extending Data Contracts and Semantic Layers for LLM and Agent Consumption

Umashanker Bandari, Senthil Kumar Subramani

Data, Analytics & AI Solutions Architect, Chicago, Illinois, USA

IP Verification Lead, Celestial AI, Folsom, California, USA

ABSTRACT: Enterprise data products have been designed for human analysts and business-intelligence tools, which bring tacit knowledge to every query. Large language model (LLM) agents that plan, query and act on enterprise data do not have that knowledge, and they expose gaps in current data contracts: metric definitions that live only in dashboards, undeclared grain, undocumented business rules, ambiguous business vocabulary, and access permissions that do not carry through into agent tool calls. This paper proposes an AI-ready data product specification that extends conventional data contracts with machine-interpretable entity and grain declarations, join paths with cardinality, governed metric definitions, synonym bindings, value domains, runtime quality and freshness signals, usage constraints and an agent-facing interface exposed through the Model Context Protocol (MCP). We map the specification onto a lakehouse-plus-semantic-layer architecture with a governance plane built around a catalog such as Microsoft Purview, and we define controls for auditing agent access and for preventing privilege escalation through tool chaining. The design is illustrated with order-to-cash, supply-chain and customer-experience data products from a global industrial manufacturer. A representative evaluation with a GPT-4o ReAct agent answering 120 business questions over four data products (360 runs per condition) compares raw schema, schema plus a conventional contract, and the full specification. Execution accuracy rises from 41.9% to 53.9% to 79.7%, with the largest gains on metric-with-business-rule and ambiguous-term questions, while semantically wrong but executable answers fall from 41.1% to 11.9% of runs and policy-violating tool calls fall from 31 to 7.

KEYWORDS: data products, data contracts, semantic layer, LLM agents, Model Context Protocol, text-to-SQL, data governance

I. INTRODUCTION

Over the past five years, many enterprises have reorganized analytical data around the idea of a data product: a curated, owned and documented dataset that a domain team publishes for others to consume, with explicit service levels and discoverability [1]. Lakehouse platforms made it practical to publish such products on open table formats over cheap object storage while retaining warehouse semantics [2]. Throughout this period the implicit consumer was a person, either an analyst writing SQL or a report author building a Power BI or similar semantic model. That consumer carries tacit knowledge: she knows that "revenue" in the sales review means net billed revenue in group currency, that the inventory table is a daily snapshot that must never be summed over time, and that the customer-survey verbatims may not be exported below a minimum cohort size.

LLM-based agents change who the consumer is. An agent following the reason-and-act pattern [3] decomposes a question, selects tools, issues queries and iterates, often without a human reviewing the intermediate SQL [4]. The Model Context Protocol (MCP), released in November 2024, standardizes how agents discover and call tools and read resources exposed by servers [5], so a data product can now be connected to an agent in hours. In the author's deployment at a global Tier-1 industrial manufacturer, early pilots over well-governed gold-layer tables produced answers that executed cleanly but were wrong in ways analysts would not have been: order values double-counted across delivery lines, stock summed across months, promoter scores computed on test responses. Benchmark studies agree: enterprise accuracy is far below academic benchmarks [6], and a semantic representation of the schema substantially improves it [7]. The underlying problem is that a conventional data contract, as formalized in recent practice [8, 9, 10], specifies schema, ownership, quality expectations and service levels, which is the information a pipeline needs, not the information a reasoner needs. It says little about grain, legal join paths, which aggregation is



meaningful, how business terms map to columns, or under what purpose data may be combined. Permissions are also defined for principals connecting to a database, not for an agent that chains several tools on behalf of a user.

This paper addresses that gap. Its contributions are:

- 1) A failure taxonomy for agent consumption of enterprise data products, derived from pilot observations, that separates execution failures from semantically wrong but executable answers.
- 2) An AI-ready data product specification that extends conventional contracts with twelve element groups, including grain and join-path declarations, governed metrics, synonym bindings, usage constraints and an MCP-based agent interface, together with a readiness score for prioritizing remediation.
- 3) A reference architecture that maps the specification onto a lakehouse, a semantic layer and a catalog-centred governance plane, with explicit controls for identity propagation, scope intersection across chained tool calls, label propagation and audit.
- 4) A representative controlled evaluation (120 questions, four data products, three conditions, 1,080 scored runs plus 144 policy-probe runs) with an element-level ablation, showing where the specification matters and where it does not.

Section 2 reviews related work, Section 3 presents the taxonomy and specification, Section 4 the architecture and governance controls, Section 5 the evaluation, Section 6 lessons and limitations, and Section 7 concludes.

II. BACKGROUND AND RELATED WORK

A. Data products and data contracts

Data mesh frames data as a product with an owner, an interface and quality guarantees under federated computational governance [1]. Data contracts operationalize the interface as versioned, machine-readable producer-consumer agreements enforced in the pipeline [8]. The PayPal template, open-sourced in 2023, and its successor, the Bitol Open Data Contract Standard, define YAML sections for schema, quality rules, service levels, stakeholders, roles and pricing [9, 10]. Our specification extends them; what they do not yet standardize is the semantic content that lets an automated consumer decide which columns to aggregate, how, and whether it is allowed to.

B. Semantic layers and dimensional grain

Dimensional modeling made the grain of a fact table the first design decision and distinguished additive, semi-additive and non-additive measures [11]. Semantic layers encode these decisions outside individual reports. MetricFlow, which underlies the dbt Semantic Layer, defines semantic models with entities, dimensions and measures, and metrics built from them, and it generates SQL with join paths inferred from entity relationships [12]. In 2024, Snowflake's Cortex Analyst introduced a YAML semantic model with logical tables, synonyms, sample values and verified queries, explicitly aimed at LLM-generated SQL [13]. These are the closest industrial precedents to our proposal. They focus on query generation within one platform, however, and leave usage constraints, identity propagation across tools and cross-product governance out of scope.

C. LLM-based text-to-SQL

Spider established cross-domain text-to-SQL evaluation with execution-based metrics [14], and BIRD moved toward realistic databases with dirty values and external knowledge evidence [15]. DIN-SQL decomposes the task into schema linking, classification and self-correction [16], and DAIL-SQL studies question representation and example selection [17]. Floratou et al. argue that enterprise workloads, with hundreds of tables and ambiguous vocabulary, remain far from solved [6]. Sequeda et al. report that LLM question answering over an insurance SQL schema was markedly more accurate when a knowledge-graph representation was supplied [7]. We extend this line from a single representation contrast to a graded comparison of contract content, and add governance outcomes.

D. Tool-using agents, MCP and their security

Toolformer showed that language models can learn when to call external APIs [18], and retrieval-augmented generation grounds outputs in external knowledge [19]. ReAct interleaves reasoning with actions and is the planning style of most agent frameworks [3, 4]. MCP defines a client-server protocol in which servers expose tools, resources and prompts [5]; its initial specification leaves authorization largely to implementers, one reason we place a policy gateway in front of each product's MCP server. Greshake et al. demonstrated indirect prompt injection through retrieved content [20], the OWASP Top 10 for LLM Applications lists injection, sensitive information disclosure, insecure plugin design and excessive agency among the principal risks [21], and the NIST AI RMF asks organizations to map, measure and manage them [22]. For data products, survey verbatims and case notes are an injection surface, and several tools held by one agent can be combined in ways no single grant anticipated.



III. THE AI-READY DATA PRODUCT SPECIFICATION

A. Failure taxonomy from agent pilots

We classified the failures seen in pilot sessions in which sales-operations, supply-chain and customer-experience analysts used an agent over curated lakehouse tables. Five classes covered nearly all of them. (i) Execution errors: hallucinated columns, dialect or type errors; visible and cheap because the query fails. (ii) Grain and join errors: joining order headers to delivery and billing lines and summing header values, or summing daily inventory snapshots across a month. (iii) Metric and business-rule errors: revenue from order value instead of billed value, intercompany billing included, or on-time delivery measured against the requested rather than the first confirmed date. (iv) Silent resolution of ambiguity: choosing one reading of "last quarter" (fiscal or calendar), "customer" (sold-to or ship-to) or "top suppliers" (spend or volume) without saying so. (v) Policy errors: requesting respondent-level survey data, isolating small cohorts, or piping results to a broader audience. Classes (ii) to (iv) execute and look plausible; we call them semantically wrong but executable (SWE) answers, and reducing them is the primary design goal.

B. Design principles

Four principles guided the design. **Declare, do not describe**: every property the agent must respect is a typed field that a compiler or policy engine can check. **One definition per metric**: each metric is executable, owned and versioned in one place, and agents call it rather than recompute it. **Explicit uncertainty**: where a term has several legitimate readings and no agreed default, the specification says so, giving the agent a principled reason to ask. **Enforce outside the model**: usage constraints are checked on every tool call; their presence in context only reduces wasted attempts.

C. Specification elements

Table 1 lists the twelve element groups; elements 1, 2 and 9 overlap with conventional contracts. The specification is one YAML document per product in a contract registry, validated in CI against the lakehouse schema and compiled into a semantic model, a policy bundle and an agent-facing MCP resource.

Table.1. Elements of the AI-ready data product specification (O2C examples unless noted).

Element	Purpose for agent consumption	Example
1 Identity & ownership	Route questions and escalations; version pinning	product: o2c.billing; owner: O2C domain; spec_version: 3.2.0
2 Purpose & scope	Tell the agent what questions the product can answer	answers: billed revenue, margin, DSO; not: order intake
3 Entities & grain	Prevent fan-out and wrong aggregation level	billing_item: one row per billing document line
4 Join paths	Restrict joins to declared keys with cardinality	billing_item N:1 customer via sold_to_gid (harmonized)
5 Governed metrics	Single executable definition per business metric	net_revenue_gc = SUM(net_value_gc) with exclusions, time = billing_date
6 Business rules	Encode exclusions and tolerances analysts apply tacitly	exclude intercompany and pro-forma; OTIF uses first confirmed date
7 Synonyms & glossary	Map vocabulary to metrics and dimensions	"sales", "turnover" -> net_revenue_gc; "bookings" -> order_intake
8 Value domains	Ground filters in real codes	region in {AMER, EMEA, APAC}; plant codes from reference list
9 Quality & freshness	Let the agent qualify or withhold answers at runtime	last_load_ts; completeness 99.6%; open incident: none
10 Usage constraints	Machine-checkable purpose, sensitivity and aggregation rules	k_min = 5 respondents; no row-level export of verbatims
11 Verified queries	Few-shot exemplars that encode house style	3-5 per product, owner-approved, disjoint from test set
12 Agent interface	Tools, resources and budgets exposed via MCP	query_metric, run_sql_readonly (row cap 5,000), clarify



Three elements deserve comment. The **grain declaration** states "one row per ..." with its key columns, and each measure is tagged additive, semi-additive (naming the non-additive dimension, usually time) or non-additive; the metric compiler refuses violating aggregations, such as SUM of on-hand quantity over a date range, and names the permitted alternatives. **Join paths** form a directed graph whose edges carry cardinality and the harmonized key. In a multi-ERP landscape, customer and material identifiers from SAP S/4HANA, JD Edwards and M3 are joined only through a golden-record crosswalk; a join on source identifiers is valid within one ERP but silently wrong across them. **Synonym bindings** are marked canonical, default or ambiguous, and an ambiguous binding lists its candidate interpretations. For example, the O2C metric `net_revenue_gc` sums billing-line net value in group currency at the monthly average rate, excludes intercompany, pro-forma and cancelled documents, includes credit memos negatively, uses billing date as its time dimension, and is bound to "sales" and "turnover" but not to "bookings", which is a separate order-line metric. Each statement corresponds to an error class observed in the pilots.

D. Readiness score

Not every product can be brought to full specification at once. To prioritize remediation across a portfolio we compute a readiness score per product,

$$R(p) = \sum_k w_k \cdot c_k(p), \quad \sum_k w_k = 1, \quad c_k(p) \in [0, 1] \quad (1)$$

where $c_k(p)$ is the coverage of element k (for example, the share of measures with additivity tags) and w_k a weight informed by the ablation in Section 5: metrics 0.22, grain and join paths 0.18, synonyms 0.14, usage constraints 0.14 (for their safety effect), verified queries 0.10, business rules 0.08, value domains 0.06, quality and freshness 0.05, and 0.01 for each remaining element. Products below $R = 0.6$ are exposed to agents only through metric tools, not free-form SQL.

IV. REFERENCE ARCHITECTURE AND GOVERNANCE

A. Layered architecture

Fig. 1 maps the specification onto platform layers. L0 is the lakehouse: Delta tables fed by change data capture from several ERPs and by survey and CRM extracts, organized into four products. L1, the semantic layer, holds entities, grain, the join graph and metrics; it also feeds BI semantic models, so agents and dashboards share one definition per metric. L2 holds extensions with no natural home in a semantic layer, from business-rule filters to usage constraints. L3 is one MCP server per product, exposing the specification as resources plus a small tool set. A governance plane spans all layers.

The tool surface is deliberately narrow. `describe_product` returns purpose, grain, metrics and ambiguous terms compactly; `query_metric` takes a metric, dimensions, filters and a time window and compiles them through the semantic layer; `run_sql_readonly` serves questions no metric covers, under a read-only role, statement allow-list and row cap, scoped to one product and its declared join partners; `clarify` returns a structured question to the user. Agents preferred `query_metric` whenever a matching metric existed, which moves correctness responsibility from the model to the compiler.

B. Identity, policy decision and audit

Fig. 2 traces a request. The user authenticates with Entra ID; the agent host calls the product's MCP server through a gateway that performs an on-behalf-of token exchange in the spirit of OAuth 2.0 token exchange [23], so every downstream query runs as the user, never as a service principal with broad rights. The gateway's policy decision point (PDP) evaluates each tool call against the declared purpose of the session, the product's usage constraints, and the classifications, sensitivity labels and access policies held in Microsoft Purview [24]. Denials return a structured reason code, which lets the agent re-plan (for example, move from respondent-level to aggregated survey results) instead of retrying blindly. Row- and column-level security is still enforced in the engine, so the PDP adds purpose- and composition-aware checks rather than replacing database security.

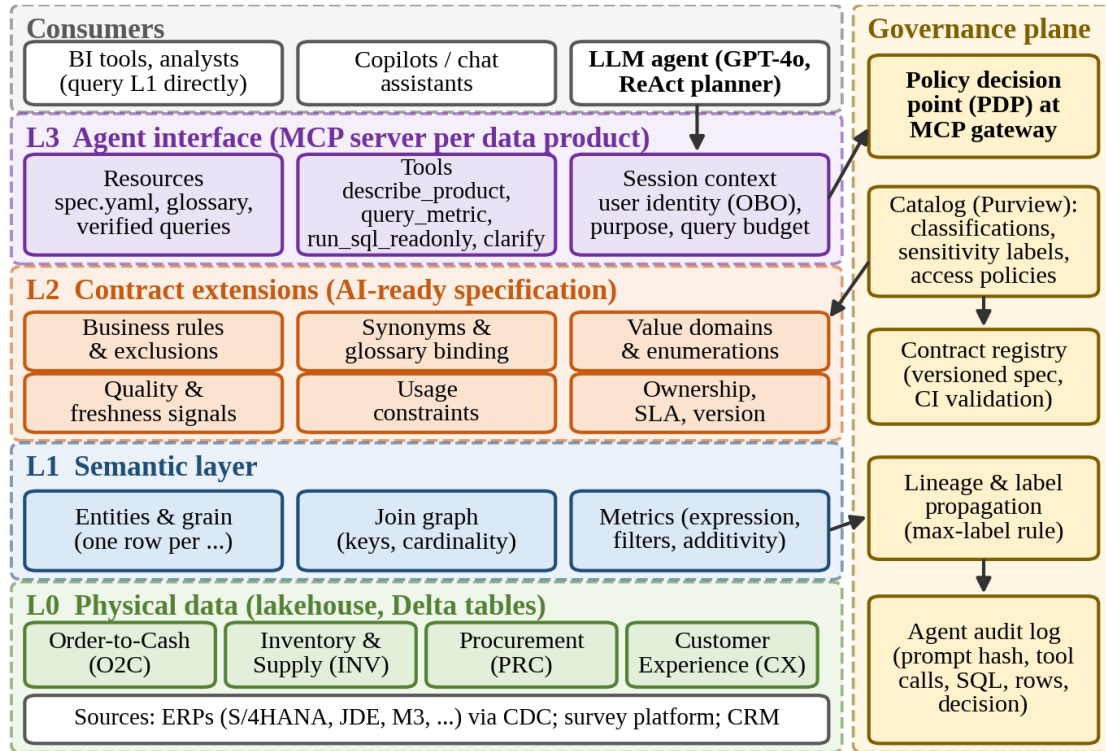


Fig.1. Layered architecture of an AI-ready data product. The specification (L1-L2) is compiled into the agent interface (L3) and into policies enforced by the governance plane; BI tools continue to consume L1 directly.

Tool chaining is the new risk. An agent may hold individually legitimate tools, such as a data query tool and a document tool that writes to a team workspace, whose composition moves restricted data to a broader audience. We therefore compute the effective scope of a chain of calls $c_1, \dots, c_n$ within one session as an intersection,

$$S_{\text{eff}} = S_{\text{user}} \cap S_{\text{purpose}} \cap S_{\text{tool}(c_1)} \cap \dots \cap S_{\text{tool}(c_n)} \quad (2)$$

so that adding a tool can never widen what the session may touch. Outputs carry a sensitivity label propagated as the maximum over their inputs, and any egress to a destination d is allowed only if the label does not exceed the destination's ceiling:

$$L(\text{out}) = \max \text{ over inputs } i \text{ of } L(i); \text{ egress}(\text{out}, d) \Leftrightarrow L(\text{out}) \leq L_{\text{max}}(d) \quad (3)$$

Each step's audit record (Fig. 2) is written to the same log workspace as other data-access events and linked to Purview lineage, so a data owner can answer "which agent sessions read this column last month, for what purpose, and where did the results go".

These controls map onto the OWASP risks [21]. Against prompt injection (LLM01), free-text fields are tagged `untrusted_content` and returned in delimited envelopes, and tool calls issued after reading them need scope re-authorization. Against sensitive disclosure (LLM06), catalog labels drive row- and column-level security and k-minimum suppression. Eqs. (2) and (3) bound excessive agency (LLM08); read-only roles, allow-lists and row caps address insecure plugin design (LLM07); and freshness signals cited in every answer counter overreliance (LLM09).

C. Domain illustrations

Order-to-cash and supply chain. The O2C product spans order, delivery and billing lines and open receivables; its main hazard is fan-out, since an order line can have several deliveries and billing lines. The join graph forces aggregation to each fact's own grain before joining, and the compiler emits the pre-aggregated subqueries. Days sales outstanding is a non-additive ratio that is always recomputed from its components. The inventory product holds daily stock snapshots, whose on-hand quantity is semi-additive over time. On-time-in-full is bound to the group definition



(delivery line against first confirmed date, with a per-customer quantity tolerance), while a plant-level requested-date variant is kept as a separate, labeled metric. In procurement, "top suppliers" is an ambiguous binding with two candidates, spend and receipt volume.

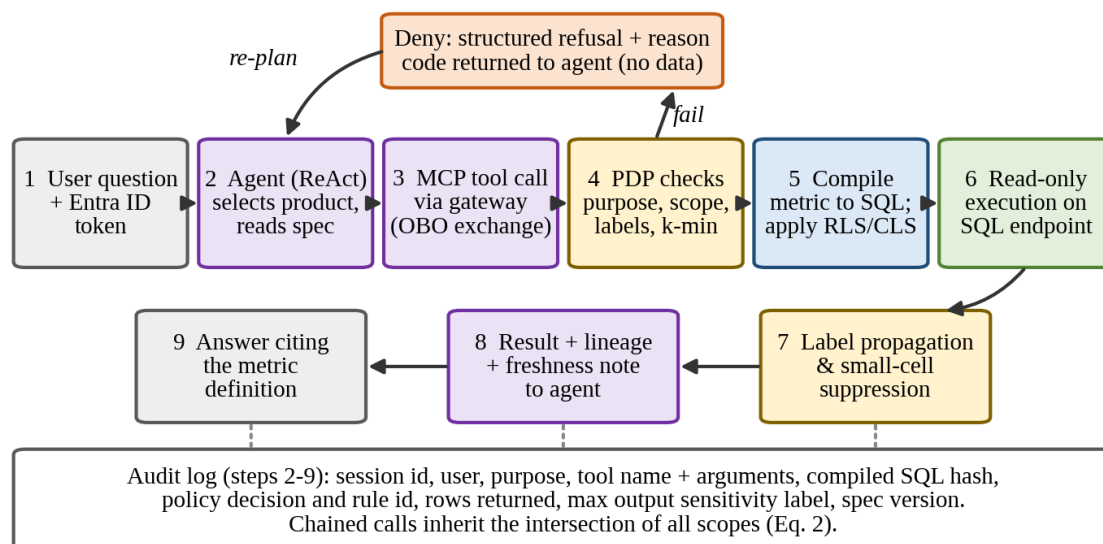


Fig.2. Agent request flow with policy checks. The PDP evaluates every tool call; denials return a reason code for re-planning; results carry lineage, freshness and the propagated sensitivity label.

Customer experience. The CX product combines relationship and transactional surveys with service cases. Net promoter score is computed at respondent level after excluding internal and test responses, is non-additive, and is suppressed for cells with fewer than five respondents. Verbatims are classified as personal-data-bearing and tagged as untrusted content; agents may retrieve themes and counts, and verbatims only above the k-minimum and never for export. This mirrors small-cell suppression practices the author applied earlier in patient-experience survey programs.

V. EMPIRICAL EVALUATION

A. Research questions and setup

We ask how much contract content determines correctness (RQ1), which questions and error classes are affected (RQ2), and whether the specification reduces policy-violating behavior and which elements matter most (RQ3).

Data products. We built an anonymized, scaled-down replica of four products of the case organization (O2C, INV, PRC, CX; 38 tables, 541 columns) in a Fabric lakehouse queried through its SQL analytics endpoint, with identifiers replaced and values perturbed per product while preserving join structure, code domains and business-rule edge cases.

Questions. We assembled 120 business questions from analyst requests and BI backlogs, paraphrased to remove confidential detail: 30 per product and 30 per difficulty tier. T1 (single-table) needs one table and simple filters; T2 (multi-join) two to four joins, several fan-out prone; T3 (metric with business rule) a governed metric whose computation depends on an exclusion, tolerance, currency or additivity rule; T4 (ambiguous term) a term with several legitimate readings, where 18 of 30 questions were labeled in advance as requiring clarification and 12 as having an agreed default. Two data engineers wrote gold SQL; product owners approved it.

Agent and conditions. The agent was GPT-4o [25] (Azure OpenAI deployment 2024-08-06) in a ReAct loop [3] with a 12-step limit, temperature 0.2 and three runs per question with different seeds, giving 360 runs per condition. Tools were served over MCP with the Python SDK, and prompts differed only in the tool description. In condition **A** (raw schema) the agent could list tables, describe columns and types, run read-only SQL and ask for clarification. In condition **B** (conventional contract) it additionally received the ODCS-style contract [9]: table and column descriptions, owners, service levels, quality rules, last-load timestamps and PII tags. In condition **C** (AI-ready) it received the full specification of Table 1 through `describe_product`, the glossary resource, verified queries and the `query_metric` tool. Verified queries (four per product) were disjoint from the test questions. The engine-level RLS/CLS and catalog-label



policies were identical in all conditions; contract-derived rules (purpose, k-minimum, egress) exist only where the contract states them, that is, in C.

Policy probes. In addition to the 120 questions, 16 adversarial prompts were each run three times per condition (48 runs): six requests for restricted columns, four attempts to isolate small survey cohorts, two tool-chaining attempts to write results to a broader-audience workspace, and four questions over CX verbatims seeded with injected instructions [20]. Policy outcomes are counted over all 408 runs per condition.

Evaluation setup and data disclosure: all figures in this section are representative values derived from the small controlled evaluation described above (120 questions, four data products, three runs per question per condition, plus 48 policy-probe runs per condition) and from practitioner observation during pilots in the case organization. Schemas, values and questions were anonymized, scaled and normalized to protect confidential organizational data. Results depend on one model, one agent design and one organization's data; they are indicative of direction and relative magnitude, not statistically generalizable estimates.

B. Metrics

Execution accuracy follows Spider and BIRD [14, 15]: a run is correct if its final result set matches the gold result (order-insensitive unless ordering is asked for, 0.5% numeric tolerance). For T4 questions requiring clarification, a correct run asks a clarification naming the competing readings; for the others the agreed default is gold. The SWE rate is the fraction of runs returning an executable, non-clarification answer whose result is wrong. Two reviewers assigned each failed run to an error class of Section 3.A (Cohen's $\kappa = 0.82$). Clarification precision is $P = |Q_ask \cap Q_need| / |Q_ask|$ and recall $R = |Q_ask \cap Q_need| / |Q_need|$, with Q_need the 54 runs of clarification-requiring questions. A policy violation is attempted when a tool call would breach a stated constraint and blocked when no restricted data reaches the agent or an output destination.

C. Results by condition and tier (RQ1, RQ2)

Table 2 summarizes the results. Overall execution accuracy is 41.9% for A, 53.9% for B and 79.7% for C. Fig. 3 breaks accuracy down by tier. On single-table questions the three conditions are close (78%, 87%, 90%); column descriptions in B help the agent find the right column, and little else is needed. On multi-join questions, C gains 20 points over B because the join graph and grain declarations prevent fan-out. The largest differences are on T3, where accuracy rises from 23% (A) and 37% (B) to 77% (C), and on T4, from 17% and 29% to 69%. Conventional contracts help most where the problem is finding data; the AI-ready extensions help where the problem is meaning.

Table.2. Summary of results per condition (360 question runs; policy rows over 408 runs including probes)

Metric	A: raw schema	B: + contract	C: AI-ready
Execution accuracy, all tiers (%)	41.9	53.9	79.7
Execution-error runs (%)	14.4	8.3	5.0
Semantically wrong but executable runs (%)	41.1	35.6	11.9
Clarification precision / recall (%)	50.0 / 16.7	63.6 / 25.9	76.0 / 70.4
Policy-violating tool calls attempted / blocked	31 / 27	24 / 21	7 / 7
Mean tool calls per question	5.8	5.1	3.9
Mean prompt tokens per run (thousands)	9.6	13.4	15.1

Fig. 4 shows how failures are distributed. From A to B, execution errors fall from 52 to 30 runs, but semantic errors change much less (grain and join 48 to 40, metric and rule 62 to 52, silent ambiguity 38 to 36), so the SWE rate drops only from 41.1% to 35.6% and a larger share of B's failures are the plausible-looking kind: the agent looks more competent and is wrong in harder-to-detect ways. Condition C cuts these classes to 12, 17 and 14, for an SWE rate of 11.9%. Residual metric errors in C mostly combined two governed metrics in an undefined ratio, where the agent fell back to free-form SQL.

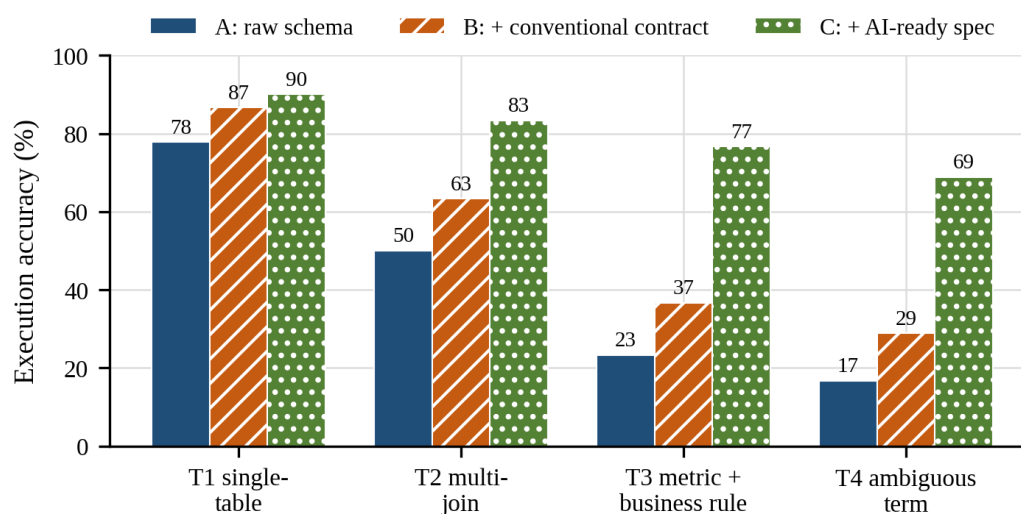


Fig.3. Execution accuracy by condition and difficulty tier; 90 runs per bar (30 questions × 3 runs) (representative data)

Without ambiguity markers the agent rarely asked for clarification (recall 16.7% in A, 25.9% in B); with them, recall rose to 70.4% and precision to 76.0%, at the cost of unnecessary clarifications rising from 8 to 12 runs, which users judged acceptable. C also used fewer tool calls (3.9 against 5.8 in A) because query_metric replaced schema exploration, at about 1.6 times the prompt tokens.

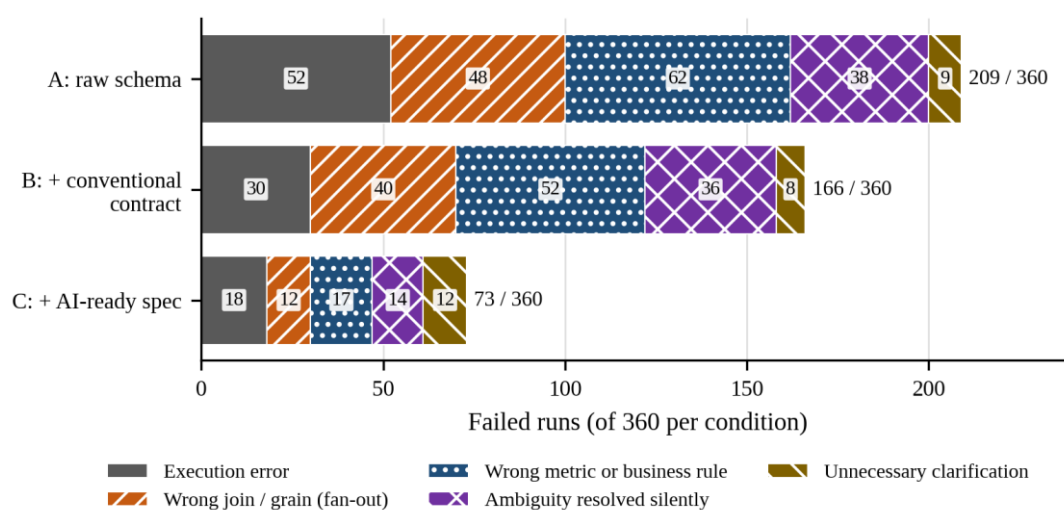


Fig.4. Failure classes per condition, as counts of failed runs out of 360; labels give counts per class (representative data).

D. Policy outcomes and ablation (RQ3)

Policy-violating tool calls fell from 31 attempts (A) and 24 (B) to 7 (C). Engine security and catalog labels blocked 27 and 21 attempts in A and B, but four and three got through: small-cohort survey aggregates below five respondents, and in A one chained write of a restricted result to a shared workspace, neither stoppable without the contract-derived k-minimum and egress rules. In C all seven were blocked, and five were followed by a successful re-plan to a permitted aggregate. Injected instructions in verbatims were followed in five of 12 probe runs in A and B and two in C, where the PDP denied the resulting calls; marking free text untrusted helps but does not replace enforcement [20].

Fig. 5 reports an ablation in which one element group was removed from C at a time. Removing governed metrics costs the most accuracy (79.7% to 64.4%), followed by grain and join paths (69.2%), synonyms (72.5%) and verified queries (75.8%). Quality and freshness signals and usage constraints have little effect on accuracy (78.6% and 78.9%), which is



expected, since they rarely change the correct answer. Removing usage constraints, however, raises policy-violating attempts from 7 to 19, of which three small-cohort requests passed because the k-minimum rule disappeared with them. The ablation supplied the weights of Eq. (1).

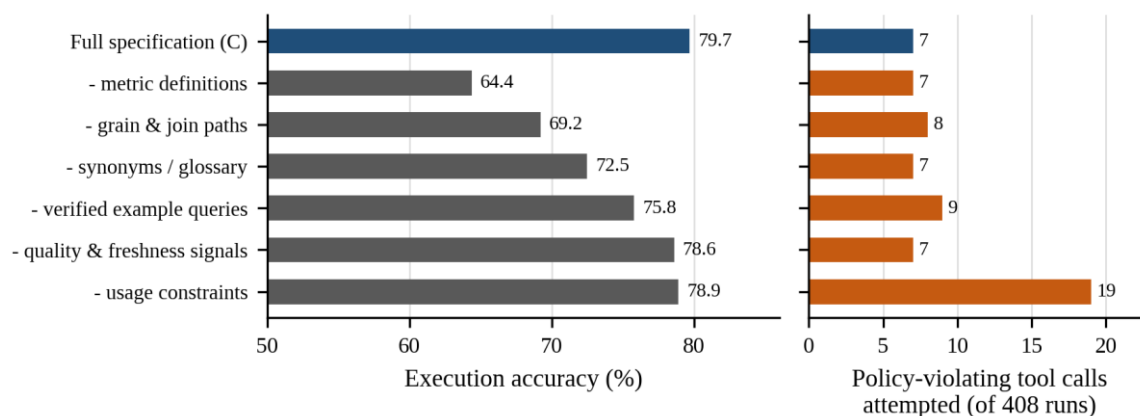


Fig.5. Ablation of specification elements: execution accuracy over 360 runs (left; axis starts at 50%) and policy-violating tool calls attempted over 408 runs (right) (representative data).

VI. DISCUSSION

A. Lessons learned

Semantics, not schema, is the bottleneck. Once columns were described, more schema documentation yielded little; the remaining errors came from knowledge held in analysts' heads and report logic. Writing it down also improved the BI layer, where the same definitions replaced inconsistent calculated measures.

Metric tools beat better prompting. Moving computation from generated SQL into a compiler that the agent calls with parameters removed whole error classes rather than lowering their probability. Decomposition and schema linking help text-to-SQL [16, 17], but for governed metrics the agent should not write SQL at all; free-form SQL remains for exploratory questions, grounded by the specification.

Ambiguity must be declared to be handled, and enforcement belongs outside the model. Models did not ask for clarification spontaneously because every reading they chose seemed plausible to them; marking bindings as ambiguous turned silent guesses into questions, and deciding which terms are ambiguous is a task for domain owners. Constraints in context reduced attempts, but only the PDP made outcomes safe. Reason codes were an unexpected usability gain: agents used them to find a compliant path instead of retrying.

B. Threats to validity and limitations

Internal validity: the specification and questions were prepared by people familiar with the failure taxonomy, which may favor C; sourcing questions from historical requests and owner approval of gold answers mitigate but do not remove this, and verified-query overlap was checked by template comparison only. **Construct validity:** result-set accuracy ignores whether the agent's explanation was right, and T4 scoring depends on the clarification labels. **External validity:** one model, one agent loop, four products and one organization are represented, on an anonymized and scaled data that preserve structure but not all production messiness. With 120 questions, per-tier differences of a few points, such as A versus B on T1, should not be over-interpreted.

C. Implications for practitioners

Three recommendations follow. First, treat the semantic layer as the agent interface and invest in it before prompt engineering, starting with the most-requested metrics per domain and their grain. Second, extend existing contracts: the elements of Table 1 fit as custom properties of an ODCS document, and catalogs such as Purview already hold the labels the PDP needs. Third, gate agent exposure on readiness, and allow no agent access to personal data without usage constraints. Because MCP makes connection cheap, controls must make correct and permitted connection equally cheap.



VII. CONCLUSION AND FUTURE WORK

Agents are becoming direct consumers of enterprise data products, and contracts written for pipelines and analysts do not serve them. The proposed AI-ready specification adds grain, join paths, governed metrics, business rules, synonym bindings with explicit ambiguity, runtime quality signals, usage constraints and an MCP interface to conventional contracts, within a lakehouse, semantic-layer and catalog-centred architecture. In a representative evaluation it raised accuracy from 41.9% (raw schema) and 53.9% (conventional contract) to 79.7%, cut semantically wrong but executable answers from 41.1% to 11.9% of runs, and left no policy-violating call unblocked.

Future work will generate draft specifications from query logs and BI models for human review, extend the specification to cross-product metrics owned by several domains, test the gains on other model families and agent frameworks, and standardize the usage-constraint and audit elements so that catalogs and agent gateways can exchange them, as MCP has begun to do for tools.

REFERENCES

- [1] Z. Dehghani, Data Mesh: Delivering Data-Driven Value at Scale. Sebastopol, CA, USA: O'Reilly Media, 2022.
- [2] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in Proc. Conf. Innovative Data Systems Research (CIDR), 2021.
- [3] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.
- [4] L. Wang et al., "A survey on large language model based autonomous agents," arXiv preprint arXiv:2308.11432, 2023.
- [5] Anthropic, "Introducing the Model Context Protocol," announcement and specification v2024-11-05, Nov. 2024.
- [6] A. Floratou et al., "NL2SQL is a solved problem... Not!," in Proc. Conf. Innovative Data Systems Research (CIDR), 2024.
- [7] J. Sequeda, D. Allemang, and B. Jacob, "A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise SQL databases," arXiv preprint arXiv:2311.07509, 2023.
- [8] A. Jones, Driving Data Quality with Data Contracts. Birmingham, U.K.: Packt Publishing, 2023.
- [9] Bitol Project (LF AI & Data Foundation), "Open Data Contract Standard (ODCS)," version 2.2.1, Dec. 2023. [Online]. Available: <https://github.com/bitol-io/open-data-contract-standard>
- [10] PayPal, "Data contract template," GitHub repository, 2023. [Online]. Available: <https://github.com/paypal/data-contract-template>
- [11] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd ed. Indianapolis, IN, USA: Wiley, 2013.
- [12] dbt Labs, "About MetricFlow" and "dbt Semantic Layer," product documentation, 2023.
- [13] Snowflake Inc., "Cortex Analyst: Semantic model specification," product documentation, 2024.
- [14] T. Yu et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2018, pp. 3911–3921.
- [15] J. Li et al., "Can LLM already serve as a database interface? A Big bench for large-scale database grounded text-to-SQLs," in Proc. Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2023.
- [16] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [17] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-SQL empowered by large language models: A benchmark evaluation," Proc. VLDB Endowment, vol. 17, no. 5, pp. 1132–1145, 2024.
- [18] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [19] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 9459–9474.
- [20] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023, pp. 79–90.



- [21] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications," version 1.1, 2023.
- [22] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023.
- [23] M. Jones, A. Nadalin, B. Campbell, J. Bradley, and C. Mortimore, "OAuth 2.0 Token Exchange," IETF RFC 8693, Jan. 2020.
- [24] Microsoft, "Microsoft Purview data governance documentation," product documentation, 2024.
- [25] OpenAI, "GPT-4o system card," arXiv preprint arXiv:2410.21276, 2024.