

The Role of Semantic Models in Modern Business Intelligence

Elavarasu Ramasamy

Sr. Director – BI Analytics

California, USA

elavarasu.ramasamys@gmail.com

Abstract

With the modern BI stack, many people are able to access shared warehouse data, but each dashboard will typically interpret the same business metric differently. Semantic models have been suggested as the solution, represented as a layer of entities, joins, dimensions and metrics between the storage and the consumers. We test this by use of a reproducible experiment over a synthetic retail warehouse (100,000–600,000 order lines). We simulate 200 analysts per metric, each re-implementing four common KPIs with plausible deviations in their definitions and measure latency and authoring size, (i) implement a minimal semantic layer with aggregate-aware query routing, (ii) measure latency and authoring size. With our assumptions, ad-hoc SQL reproduced the canonical value for 27–42% of analysts, based on the metric, with a mean order absolute error of up to 17.0% (avg order value). The speed-up of additive queries was up to 10132× while correctly disallowing aggregation of a non-additive measure. There was a 3.5× reduction in semantic requests compared to like-for-like SQL. Synthetic data and assumed behaviour yields results—read these as a controlled demonstration and not as field evidence.

1. Introduction

Self-service analytics has helped to put business data in the hands of users, managers, analysts and business decision makers, reducing the need for technical departments to do so. There is another basic semantic consistency problem, though, because of this analytical decentralisation. Different users and/or different analytical tools could assign specific numbers to the same business metric so the numbers themselves can differ even for the same underlying data. An organization can have different definitions for a revenue in a spreadsheet, on a business intelligence dashboard, with a SQL query, and in a data-science notebook. There are several ways in which one analyst can calculate revenue based on gross sales, while another would exclude returns and discounts, and another would only recognise revenue. While each calculation might be logical given local logic, the company still has several different interpretations of the same metric. This inconformance may result in miscommunication in management meetings, more complex performance measurement and eventually into losing trust in analytical systems over time. It's more than just a data access problem, it's a shared meaning problem. The meanings of key business terms, such as revenue, profit, customer, order, active user, churn or conversion rate, should be uniformly interpreted by the organisations. One way to overcome this is to create a business abstraction layer that abstracts the physical structure that stores the data, but exposes the business concepts. Without the need for every user and application to know about the underlying tables, joins, database structure and calculation rules, the semantic layer provides a consistent representation of the business entities and metrics that can be reused across analytical environments. This is nothing new. The software programs that preceded business intelligence and analytical data management tried to address the same issue. Kimball and Ross (2013) proposed a dimensional modelling approach that makes the analytical structure easier to decipher as they structured it around facts and dimensions that are of interest to the business users. Likewise, OLAP (Online Analytical Processing) systems have brought multidimensional representations that enabled users to analyse measures on different dimensions like “time”, “geography”, “product” and “customer”. The principles of OLAP systems were presented by Codd et al. (1993) and a more detailed overview of data warehousing and OLAP technologies was presented by Chaudhuri and Dayal (1997). Traditional BI platforms also brought in terms

like universes that enabled business users to interact with business concepts instead of directly with complex databases.

This can thus be considered to be an evolution of the old (separation of concerns) principle, namely the separation of concerns in the semantic-layer, headless BI, and metrics-layer architectures. These architectural designs are designed to enable the same semantic definitions to be shared as a shared service, that can be used by several downstream applications or tools. It includes such as LookML, of dbt's MetricFlow, Cube and Power BI tabular models. A metric can be centrally defined, and shared into dashboards, reports, analytical notebooks, APIs, as well as other data consumption contexts. This is particularly important in the context of the growth of LLM-powered analytical agents. But, typically, traditional users can detect if the generated query is not the one they need to perform for business and be able to correct it by hand. The LLM-based agent can on the other hand create syntactically proper SQL, pick the wrong table, employ the proper aggregation, or miss a basic business rule. Thus, it may be legal to run though it may give back the wrong result. A well-trained semantic layer can provide an explicit description of metrics, dimensions, relationships, filters, aggregation rules and business terms to the agent. In this regard, the semantic layer isn't only a governance tool, it could be a knowledge interface between the business meaning and queries generated by the machines. The research question then becomes more complex than it is whether using semantic layers is effective to facilitate data governance. It asks the question: Does centralised semantic definitions result in measurable improvements in analytical work itself? Users might only have to write joins, filters, calculations, and transformation logic once, and then use it again and again, which might save them time. The performance of query execution can be further improved by re-usable semantic definition that can optimize repeated computation in query systems and analytical systems. Given these potential scenarios, there are good reasons to take more than just management and consistency into account, and there are good reasons to take a look at what it means to the analytical accuracy, query performance and user effort.

2. Background: what a semantic model contains

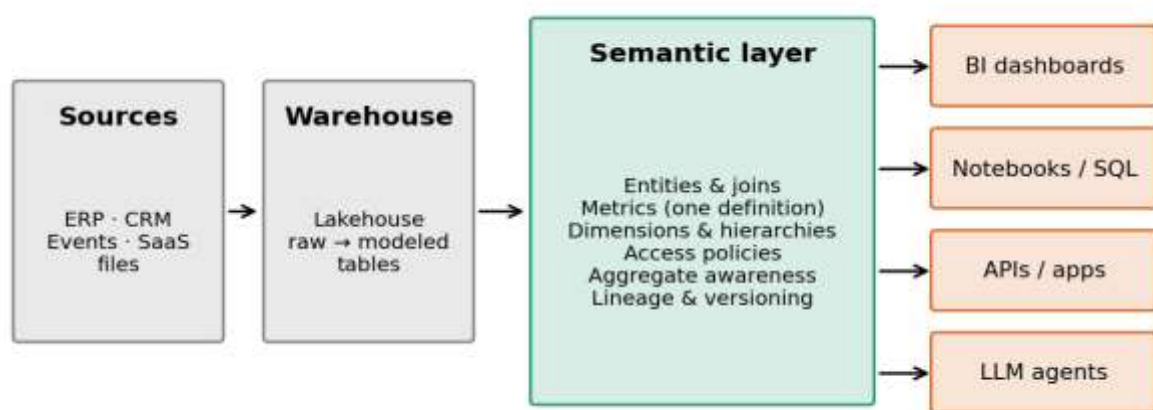


Figure 1. Position of the semantic layer in a modern BI architecture. Consumers request metrics by name; the layer compiles governed SQL against the warehouse.

Typical restrictions in the scope of a semantic model include declaration of (1) entities and join paths, (2) dimensions and hierarchies, (3) metrics and a single canonical expression and a single filter, (4) row and column level access policies, and (5) access hints at a physical level, such as pre-aggregations. Note that there is a difference between sum and distinct count, a sum can be rolled up from a finer grain, but generally it can't.

3. Method

3.1 Data

Created a star-schema warehouse using sqlite: 5,000 customers in 4 regions, 400 products in 5 product categories, and a fact table called order lines that contains order lines for the month, status, quantity, price, discount, tax, return flag. 8% of orders are cancelled and about 7% of lines completed are returned. The seed is set and can be reproduced as results vary about 2 seconds depending on machine.

3.2 Experiment 1 – metric consistency

For each of the four KPIs, we have given an expression, canonical and realistic deviations (Table 1). All the deviations are applied by each simulated analyst one at a time, with the given probability. For both variants, all generated variants are executed on the 200k row warehouse and then checked to see if they match the canonical variant. All variants are performed on 200k row warehouse and verified with the canonical value. The probabilities are based on our assumption and we test the sensitivity of these assumptions by changing the probabilities by $0.5\times$ and $2\times$.

Table 1. Canonical definitions and simulated analyst deviations.

Metric	Canonical definition	Simulated deviations (probability)
Net revenue	$\Sigma(\text{qty} \cdot \text{price} - \text{discount})$; completed, non-returned lines; excl. tax	include cancelled (.25), include tax (.15), ignore discount (.20), include returns (.30)
Avg order value	Net revenue $\div$ distinct orders	as above + divide by line count (.30)
Active customers (Q4)	Distinct customers with a completed, non-returned line in months 10–12	include cancelled (.25), include returned (.30), last 2 months only (.35)
Return rate	Returned value $\div$ completed value	count-based (.40), cancelled in denominator (.25)

3.3 Experiment 2 – performance

We developed a very simple semantic compiler. When it is presented with metrics and dimensions it outputs SQL with only pre-aggregates (month $\times$ region $\times$ category) when all requested measures are additive. We compare against direct SQL joining the fact table to dimensions, at 100k, 300k and 600k rows. The specification for the model is:

```
metrics:
  net_revenue:
    expr: SUM(qty * price - disc)
    filter: status = 'completed' AND ret = 0
    additive: true
orders:
```

```

expr: COUNT(DISTINCT oid)
additive: false # cannot be rolled up from a pre-aggregate
dimensions: [region (cust), category (prod), month (f)]

```

3.4 Experiment 3 – authoring size

We use character counts of semantic requests as a crude measure of effort when comparing to the SQL that would be required to answer the same six questions. This does not consider comprehension cost or tooling support.

4. Results

4.1 Metric divergence (RQ1)

Table 2.: 200 simulated analysts per metric of metric consistency. With a semantic layer: 100% match, 0% error, 0% disagreement by construction

Metric	Canonical value	Match (%)	MAPE (%)	Max error (%)	Disagreement (%)
Net revenue	3.857e+07	38.5	7.26	26.3	80.2
Avg order value	475.8	27.0	17.03	52.6	87.9
Active customers (Q4)	4,925	34.5	2.10	6.0	78.8
Return rate	0.07068	41.5	2.36	8.4	69.2

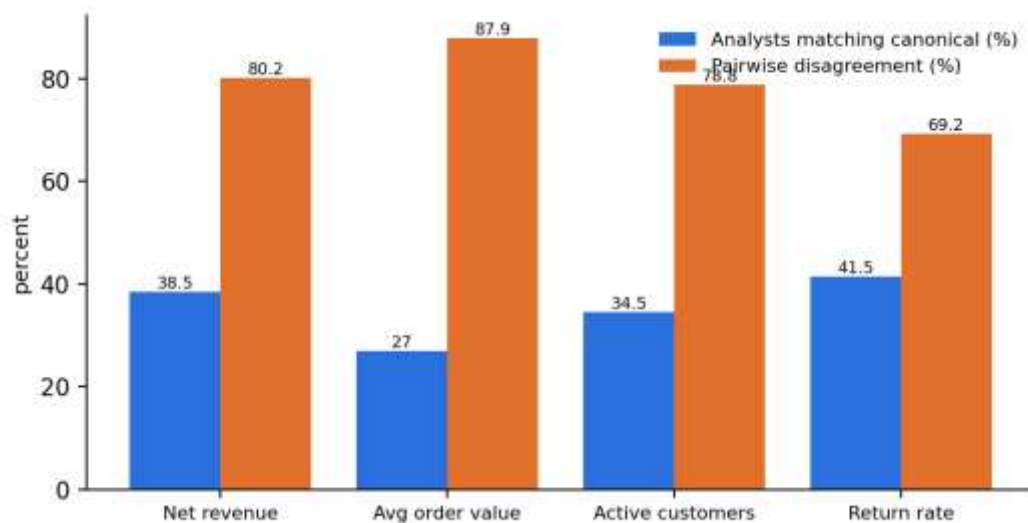


Figure 2. Share of simulated analysts reproducing the canonical metric and probability that two random analysts disagree.

If there isn't a consensus definition, a significant fraction of analysts provide non-canonical numbers, and for most metrics, two analysts selected at random disagree. The largest mean error value was obtained for the avg order value (17.0%). For net revenue, the deviations as shown in

Figure 3 are not noise around the truth: the 16 different variants have +34% offsets on both sides of the truth.

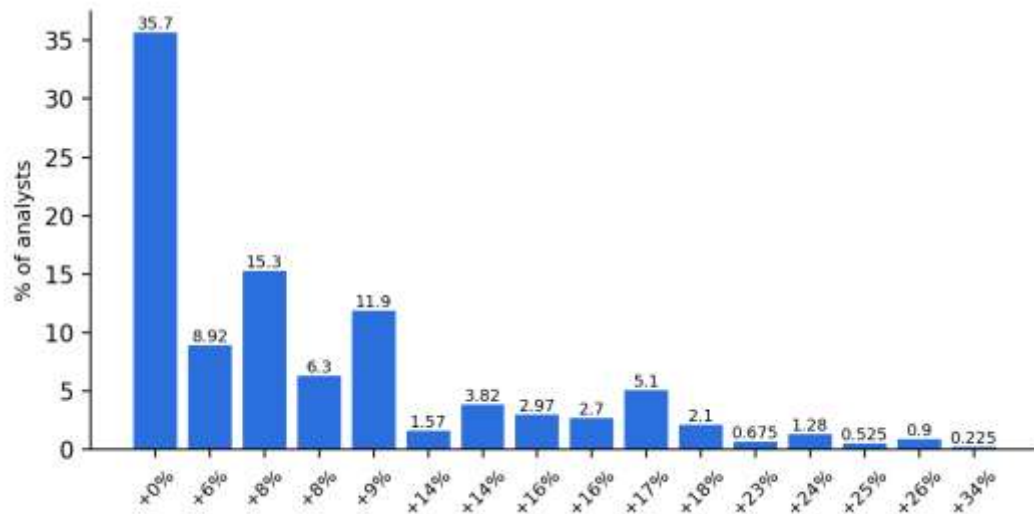


Figure 3. Expected distribution of net-revenue values reported by analysts, as % deviation from canonical (all 16 definition variants).

Table 3. Sensitivity of Experiment 1 to assumed analyst error probabilities.

Error-rate scale	Mean match (%)	Mean MAPE (%)	Mean disagreement (%)
0.5×	62.6	3.7	57.2
1×	35.4	7.2	79.0
2×	8.5	13.6	85.0

The qualitative conclusion is robust to the assumed error rates: even at half the baseline error probability, a substantial minority of analysts diverge.

4.2 Performance (RQ2)

Table 4. Median latency of 7 runs (SQLite, in-memory, single thread).

Rows	Query	Direct SQL (ms)	Semantic (ms)	Speed-up
100,000	Q1 revenue by month	34.45	0.04	856.4×
100,000	Q2 revenue by region × category	125.68	0.10	1292.4×
100,000	Q3 orders by region (non-additive)	67.30	64.07	1.1×
300,000	Q1 revenue by month	105.29	0.04	2490.5×
300,000	Q2 revenue by region × category	430.20	0.09	4916.5×

Rows	Query	Direct SQL (ms)	Semantic (ms)	Speed-up
300,000	Q3 orders by region (non-additive)	208.95	207.85	1.0×
600,000	Q1 revenue by month	227.50	0.04	5208.3×
600,000	Q2 revenue by region × category	840.07	0.08	10132.2×
600,000	Q3 orders by region (non-additive)	417.78	392.50	1.1×

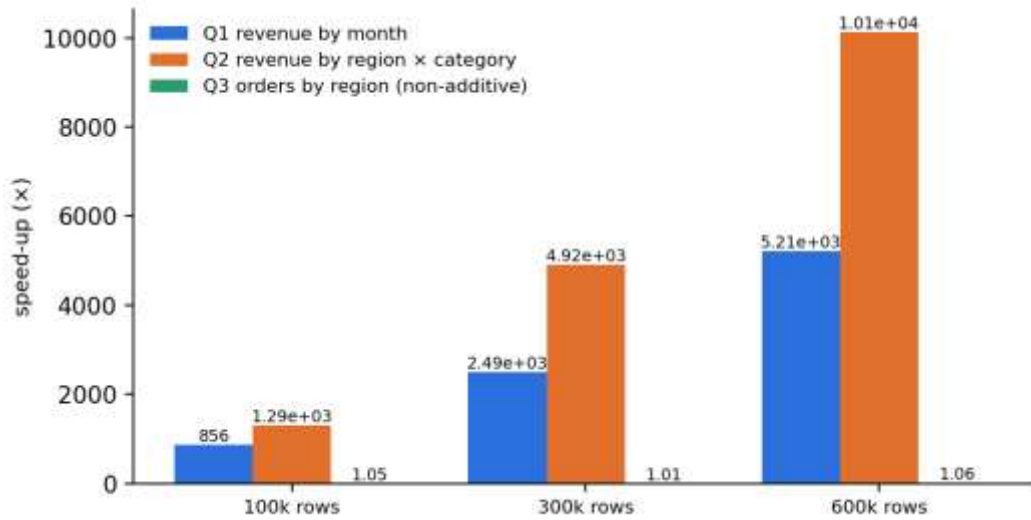


Figure 4. Speed-up of semantic-layer routing over direct SQL. Q3 falls back to the base table because distinct counts are non-additive.

At 600k rows, additive queries ran 5208× (Q1) and 10132× (Q2) faster via the pre-aggregate, while Q3 correctly kept the base-table plan ($\approx 1.1\times$). Compilation cost was 1 μ s per request, negligible against query time. Routing is only safe because additivity is declared in the model; a naive cache would have over-counted orders for Q3.

4.3 Authoring size (RQ3)

Table 5. Authoring size of six business questions.

Request	Semantic spec (chars)	Equivalent SQL (chars)	Ratio
metrics=net_revenue, dims=month	39	111	2.8×
metrics=net_revenue, dims=region	40	141	3.5×
metrics=net_revenue, dims=category	42	147	3.5×
metrics=net_revenue, dims=region, category	52	199	3.8×
metrics=net_revenue, orders, dims=region	50	170	3.4×

Request	Semantic spec (chars)	Equivalent SQL (chars)	Ratio
metrics=net_revenue, orders, dims=region, category	62	228	3.7×

Semantic requests were on average 3.5× shorter than the equivalent SQL, and the gap widens as joins and filters accumulate, because those live once in the model rather than in every query.

5. Discussion

The results demonstrate the usefulness of a semantic layer from a governance point of view and from a semantic consistency point of view and from a second and conditional performance of queries. The most easy to see is when a single definition of each key business measure is centrally managed. After defining a metric, such as a revenue, profit, number of customers, or number of active users, in the semantic model, other analytical consumers need not recreate the underlying calculation themselves. This translates to the consistency improvement being based on no particular software vendor or technology platform. Regardless of how the semantic model is implemented, whether it is dedicated metrics layer, BI semantic model or some other architectural approach, the main benefit is to aggregate all of the business definition in a single location and make the model reusable. This implies that semantic consistency is more of an architectural and governance characteristic than a product-specific characteristic. The results further prove to be more contingent concerning the advantages of semantic modelling. Not all queries are necessarily semanticised to be faster. If enough accurate information on the measures, dimensions, aggregation behaviour and data relationships exists in the semantic model, this analytical system can then select an appropriate physical data representation to achieve optimal performance. A pre-aggregated table might be helpful in a specific dimension if it is known that at that level a metric is additive and might serve as an alternative to scanning a detailed fact table multiple times. This can be beneficial to decrease the amount of data that is processed, and thereby improve the query execution time. This benefit can only be enjoyed if the load work is undertaken. For analytical questions that differ significantly from one another and are not for commonly used grains or require non-additive calculations, aggregate-aware routing might have no or minimal performance advantage. The conclusion that semantic layers automatically optimize the performance of the database in the results should therefore be made with a workload and physical data design that is suitable for the semantic metadata.

With the introduction of LLM-based analytical agents, the significance of semantic consistency is growing further. I think that the LLM would need to be able to read the names of tables, columns, documentation, examples and instructions in the context and be able to infer the meaning of the business concepts from their names, to be able to directly map to the raw tables. This leads to a situation similar to the simulated analysts in this study, who also were faced with definitional ambiguities. A query may be syntactically correct and execute without any errors, but it still may not be using the correct business definition. For example if the schema and context information provided to the agent is gross sales, net sales, recognised revenue, or another field, the agent may select the appropriate field when asking for information. One way to overcome this ambiguity is

to use a named-metric interface (NMI), where a client explicitly asks for the metrics, rather than interpreting business logic from the extremely raw tables that are being retrieved. Within this architecture, it is the agent's responsibility to request the name of a known metric (e.g. `net_revenue`) and the semantic layer would then consider which calculation, joins, filters and aggregation behaviour should be applied to the source data. This may be abstraction / control separation of requests to the natural language and the underlying physical data structures. This interpretation is a research implication in this particular study, however, which does not empirically test LLM-based agents. When a semantic layer is added, organisational and technical costs are applied to this. Just as software code is designed, curated, versioned, tested, documented and reviewed, so must be semantic models. Businesses may have different definitions of their businesses, as a result of accounting policies, management requirements, regulatory changes, product variations, or organisational changes. These changes should be cascaded down through the semantic model, and be made available to users. Version Control and Testing are key to ensure that a change in one metric will not have unintended consequences on downstream reports or analytical applications. Furthermore, if the semantic layer is not well maintained, it can be a hindrance, not a help. New measures might have to be long in order to approve, or even measure can remain for a long period of time, and the analysts will return to directly query or calculate locally. Also, semantic models don't necessarily reveal everything. Other than this structure or transformation, if it is not part of the governed model, it may be required to develop an exploratory analysis, ask new questions and make one-off investigations. The aim then is not to banish ad-hoc analysis, but to provide a secure and reliable governed environment for business ideas that are repetitive in nature and have value in decision making, and still allow enough flexibility for the exploratory development of them. Overall, the results support the separation of the concepts of semantic governance, analytical efficiency and query optimisation. The benefits of governance are fairly direct – they stem from the use of shared definitions. The benefits of efficiency come from those definitions that eliminate the need to repeat user logic. There are other requirements related to metadata quality, physical design, workload characteristics and aggregate availability that must be met in order to benefit the performance. This is important as otherwise the semantic layer will be a universal performance solution. Rather than that, it's a method of architecture which could increase consistency and decrease the analysis duplication and in some circumstances optimize the performance.

6. Threats to Validity

Results need to be taken with a grain of salt due to some limitations. For one, the data used in experiments are synthetic, and in this study, we have very small database schema. Synthetic data offer a controlled environment for the effects of semantic modelling to be isolated, but cannot fully mimic the complexity of production data environments. Organisational databases in the real world typically include many more data sources, a large number of fact tables, slowly changing dimensions, incomplete data, multiple id's, more complex business rules and legacy transformations. The properties may impact the meaning of the query and its processing. The ramifications of the effects observed in this study should not be assumed to be as applicable to large enterprise data warehouses or lakehouse environments. The other constraint concerns modeling of the analyst's behaviour. The non-semantic forms of analytical scenarios deviated from one another and the assumed probabilities were not directly observed by the professional analysts but set up for the experimental design. Analytical errors can not be independent in real

organisations. For instance, analysts using the same documentation, but different tools, might systematically use the same wrong definition, or analysts using the same tools, but different documentation, might make the same kind of errors. Similarly, some types of mistakes may be extremely common whereas others may occur only rarely. The assumed deviation probabilities are thus not meant to be a true-to-life sketch of the behaviour of real world analysts, but rather a controlled representation of semantic inconsistencies. Thirdly, data was collected in an in-memory SQLite environment, which can be very useful in evaluating different query strategies but is quite different to the current production data platforms. Distributed execution, columnar storage, caching, partitioning, indexing, query optimisers, materialised views, concurrency management and other features can have a significant impact on the performance of cloud warehouses and enterprise analytical systems. So the absolute times in the experiment are not necessarily comparable to other systems, like cloud-scale analytical warehouses. The results mainly show the relative performance of the query strategies under the experimental conditions. Another limitation is in terms of the treatment of pre-aggregation costs. The reported speed-ups are primarily for query execution; the costs of creation, maintenance, refresh and validation of the pre-aggregated structure are not accounted for in one-time costs and recurring costs. In a production system, an aggregate table causes additional data processing, storage demands. It's beneficial provided it's accessed on a regular basis and the update price isn't excessive. Furthermore, the gain in performance is partially because of the fact that it is a smaller dataset of pre-aggregated data (240 rows). This compact size can cause considerable performance differences if a detailed data set is used. Relative improvement can be achieved in a variety of ways in more extensive and complex production operations. Another limitation is the measure of authoring effort. I'm using number of analytical steps as a proxy for how much effort the end user has to go to, but if you write less SQL, or expressions, that doesn't mean that you will make it simpler for the end user to take in the effort. The first step in the analyst's job is to grasp the semantic model, become familiar with the terms of the model, find the right metrics, and know if a pre-defined metric is going to answer their analytical question. A new semantic layer could then entail a slight learning curve, but might mean that much less code is written after users become familiar with the layer. This present experiment does not measure training time, discoverability, user satisfaction and cognitive workload directly. Finally, the study does not include a longitudinal field evaluation. Metrics that are found in production but not found in a controlled experiment can be known as these. Examples include metric ownership conflicts, the definition of a business changing, metric version conflicts, incomplete documentation and user workarounds. The controlled evaluation that is introduced in the present paper would then be supported by a field study and include actual analysts, production warehouse logs, query histories and observed metric definitions. This study would provide information on if the decrease in divergence, query complexity and execution time established during the experiments would persist in realistic organisational environments.

7. Conclusion

In this study, the researchers have attempted to see how, when that shared semantic model is added to that analytical environment, in which the users would otherwise construct their business metrics without the presence of a common model, that would change the environment. The results of the controlled queries demonstrate that many of the differences in the definition of business metrics that can occur with ad-hoc analytical queries can be minimized by defining business metrics once and making them reusable. Consistency is the key advantage of the semantic layer: Multiple

analytical consumers can tie into the same business definition rather than making up a definition on their own that may be different from the company's. Besides, this work demonstrates how one can use semantic modelling to enhance query performance, under some circumstances. If the data has semantic metadata to describe how to aggregate data, and the analytical workload always needs certain grains of data, aggregate-aware routing can send queries to smaller, pre-computed views of the data. This may result in a lesser amount of data being processed, as well as quicker executions. This is a conditioned performance gain and is not an intrinsic one when considered as related to the semantic levels. It is highly dependent on the correctness of metadata, the existence of suitable aggregates, the workload and physical data architecture. Another discovery was that semantic abstraction can decrease the amount of analytical logic in the user's code. Users can use the pre-existing metrics and dimensions in the central location without being required to join, filter, calculate and apply business rules multiple times. This eliminates redundant logic, and can make some analytical queries more easily written and supported. However, a more simplistic authoring process does not imply that users need to be blind to the semantics available, and be able to use them as appropriate. This is even more important as organisations begin to deploy LLM-based analytical agents. For an agent working directly on top of the raw database schemas, there are numerous semantic ambiguities like those in human-constructed queries. An abstraction of a governed and named-metric interface could then be a helpful abstraction that an agent can refer to on understood concepts of the business—and not on the physical structure of the database. This effect was not observed in the current study but may well be investigated using LLM-based agents. In summary: The evidence suggests that the semantic layer is not a governance role in its own right, but an analytical platform that is shared. Its most powerful and direct impact is in the centralisation of business meaning, with the benefits of reduced user authored logic and increased performance from the merging of semantic metadata with physically optimal workload. Further studies are needed to confirm these findings with production scale data, real analyst activity, cloud data warehouses, and longitudinal log of queries and controlled experiments with LLM-based analytical agents. These studies would make it easier to determine if the gains reported in the controlled environment are still relevant when semantic-layer architectures are used in real-life enterprise environments.

References

1. Lei, F., et al. (2025). Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. *ICLR 2025* (oral). arXiv:2411.07763.
2. dbt Labs. (2026). Semantic layer vs. text-to-SQL: 2026 benchmark update. *dbt Developer Blog*. <https://docs.getdbt.com/blog/semantic-layer-vs-text-to-sql-2026> (vendor blog, not peer-reviewed)
3. Floratou, A., Psallidas, F., Zhao, F., Deep, S., et al. (2024). NL2SQL is a solved problem... Not! *CIDR 2024*.
4. Sequeda, J., Allemang, D., & Jacob, B. (2023). A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise SQL databases. arXiv:2311.07509.
5. Hong, Z., et al. (2024). Next-generation database interfaces: A survey of LLM-based text-to-SQL. arXiv:2406.08426.

6. Biswal, A., et al. (2025). Text2SQL is not enough: Unifying AI and databases with TAG. *CIDR 2025*. arXiv:2408.14717.
7. Li, B., Luo, Y., Chai, C., Li, G., & Tang, N. (2024). The dawn of natural language to SQL: Are we fully ready? *PVLDB*, 17(11).
8. Gao, D., et al. (2024). Text-to-SQL empowered by large language models: A benchmark evaluation. *PVLDB*, 17(5).
9. Li, J., et al. (2023). Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs (BIRD). *NeurIPS 2023, Datasets and Benchmarks Track*. arXiv:2305.03111.
10. Pourreza, M., & Rafiei, D. (2023). DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. *NeurIPS 2023*.
11. Dehghani, Z. (2022). *Data Mesh: Delivering data-driven value at scale*. O'Reilly.
12. Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering*. O'Reilly.
13. Abadi, D., et al. (2022). The Seattle report on database research. *ACM SIGMOD Record*, 51(3).
14. Hogan, A., et al. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4), Article 71.
15. Sequeda, J., & Lassila, O. (2021). *Designing and building enterprise knowledge graphs*. Morgan & Claypool.
16. Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021). Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. *CIDR 2021*.
17. Armbrust, M., et al. (2020). Delta Lake: High-performance ACID table storage over cloud object stores. *PVLDB*, 13(12).
18. Noy, N., et al. (2019). Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62(8).
19. Yu, T., et al. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. *EMNLP 2018*.
20. Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit* (3rd ed.). Wiley.
21. Halevy, A. Y. (2001). Answering queries using views: A survey. *The VLDB Journal*, 10(4), 270–294.
22. Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1), 65–74.
23. Gray, J., et al. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1), 29–53.
24. Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. *ACM SIGMOD 1996*.
25. Codd, E. F., Codd, S. B., & Salley, C. T. (1993). *Providing OLAP to user-analysts: An IT mandate*. Codd & Associates.