

Semantic Consistency in Distributed Analytics Systems: A Design Science Approach to Cross-Engine Query Equivalence

Muthupalaniappan Ramanathan

Independent Researcher, USA

Abstract

Modern analytics systems use multiple, heterogeneous query engines, balancing different requirements for performance, scalability, and workload specialization. While the engines share a similar syntax and support dialects of structured query language, they are not semantically equivalent. Heterogeneous execution environments introduce semantic inconsistencies due to differences in type systems, null semantics, data aggregations, and evaluation order. Semantic inconsistencies compromise analytic results in terms of trust, reproducibility, and governance. We apply the design science research methodology to define semantic consistency as an architectural invariant and the Canonical Semantic Representation framework to preserve analytic intent. The article consists of three layers: a parsing layer for extracting analytical intent, a semantic validation layer for enforcing deterministic policies, and engine-specific adapters for mapping the canonical representation to engine-executable queries. Our architecture closes an important gap in multi-engine analytics governance by turning semantic consistency from an implicit assumption into an explicit and enforceable system property. Domain experts who applied established design science evaluation frameworks were able to validate the framework against five proposed design principles with practical implications.

Keywords: Semantic Consistency, Distributed Analytics, Query Engines, Canonical Representation, Design Science Research

1. Introduction

In this context, commercial data platforms have transitioned from monolithic databases to poly-engine distributed systems that scale well and can specialize on different workloads. Production machine learning pipelines and analytics systems make use of billions of training dataset examples with hundreds of features and terabytes of data processed daily [1]. The resulting architectural heterogeneity enables organizations to tailor the computational characteristics of the processors to the needs of targeted workloads.

A second architectural trade-off facing multi-engine systems in both research and practice is how to coordinate the material equivalence of the same analysis logic in the different operational execution engines. When a business metric is calculated across different execution engines targeting different user populations or use-case contexts, differences in engines may produce semantic differences in the outputs [2]. This challenges analysis and governance by introducing failures of reproducibility that may be hard to both detect and interpret.

1.1 Problem Statement

Let Q be a query in a SQL-like query language and D be a dataset. For query engines $E_1, E_2, \dots, E_n$, a multi-engine architecture is based on the assumption that $E_i(Q, D) = E_j(Q, D)$ for all values of i and j , which is not true in general. Engines also differ in dialect syntax, type coercion, null handling, floating-point arithmetic, and optimizer transformations, which in practice do not usually result in catastrophic

divergences for ordinary business and consumer software, though they may have implications for financial reporting, regulatory compliance, and scientific reproducibility.

1.2 Research Gap

While query optimization across heterogeneous sites, and federated query processing has been studied, semantic heterogeneity has not received strong attention as an explicit architectural property. Earlier work (see [1]) has either assumed semantic equivalence or considered semantic mismatch as an implementation artifact rather than an architectural concern. The database community has studied data validity (schema enforcement and data cleaning) but production machine learning pipelines do not always have the appropriate scaffolding and have additional aspects to the problem [1]. There is no framework for formally representing analytic intent separate from engine semantics or for checking whether analytic intent is preserved during an engine translation.

1.3 Contribution

Our contributions are threefold. First, we formally define semantic consistency, analytical intent, and execution semantics as architectural concepts that lead to the definition of the theoretical basis of cross-engine equivalence in a distributed analytics architecture. Second, we propose a Canonical Semantic Representation (CSR) framework with three architectural layers for achieving analytical intent across heterogeneous execution environments. Additionally, the proposed framework is validated through a structured expert evaluation process according to established design science evaluation guidelines. The findings suggest that the framework adheres to the design science principles and is applicable to enterprise analytics architectures.

The rest of the paper is organized as follows. Section 2 surveys related work on distributed query processing and semantic interoperability. The following sections describe the methodological approach (Section 3), formalization of semantic consistency with its sources (Section 4), the suggested conceptual model (Section 5), and the expert-led evaluation (Section 6). The implications and limitations of the study are outlined in Section 7, and directions for future research are presented in Section 8.

2. Related Work

2.1 Distributed Query Processing

Distributed query processing has generated several principles relevant for processing queries using data from multiple sources. Federated database systems are seen as an answer to the issues of schema heterogeneity, query decomposition and the assumption of the semantic homogeneity of the relational model in the context of distributed, heterogeneous, and autonomous databases [3]. These systems allow each of their component databases to enter a federation where their information is made available to many types of applications while remaining under the control of their original system [3]. This approach to data federation has been extended to different data models by using adaptive query execution to address the structural differences among the relational, document, and graph models [4].

In most modern distributed query engines, the query plan is optimized through advanced algorithms based on cost models, data, and execution context, following the procedure of applying algebraic equivalences to the transformation of one query into another. The optimization is not preserved if the engines differ in arithmetic precision, null treatment or function evaluation semantics [3]. A central problem is assuring semantic coherence throughout the autonomous components in database management systems.

2.2 Semantic Interoperability

The majority of research into semantic interoperability has focused on enabling interchange at the schema level through the use of ontologies and mappings, which formalize (and therefore provide a common understanding of) the conceptualization (that is, the schema being modeled) [5]; but without addressing operational semantics, that is, behavioral differences in engine interpretation of identical queries.

Recent work on data contracts and schema registries is a step towards formalizing data semantics, since such contracts can provide structural compatibility. However, current infrastructure does not adequately address the difference between structural semantics and operational semantics (see [4]). Recognizing explicit and implicit data dependencies helps make sense of data context and is essential to building maintainable analytics pipelines [1].

2.3 Query Language Standardization

The SQL standard has been expanding more widely through its revisions, culminating in SQL:2016, which added row pattern matching, JSON functionality, and polymorphic table functions (see [6]). While SQL compliance has been generally increasing, support for standard features varies considerably across commercial and open-source engines. SQL compliance tests have indicated severe deviations in edge cases on null handling, type coercion, and aggregate function behavior [6]. The database implementation of rows of relational tables, and B-trees to access them was motivated by optimizations for business data processing. The same architecture was reapplied for applications with widely different properties [11]. These results indicate that syntactic standardization alone is not sufficient to achieve semantic coherence across heterogeneous execution environments.

2.4 Gap Analysis

The literature survey revealed three gaps this work seeks to fill: handling of semantic consistency as a first-class concern and a property distinct from correctness, defining an architectural model for maintaining observer intent across heterogeneous execution engines, and providing a validation framework for ensuring semantic consistency in multi-engine scenarios.

3. Research Methodology

3.1 Design Science Research Approach

We employ design science research methodology, as it is appropriate for the design and evaluation of artifacts to address problems in the information systems domain [7]. Design science research creates relevant constructs, models, methods, or instantiations to extend the boundaries of human and organizational capabilities in the information systems domain. Because design is intrinsically concerned with moving an existing state of affairs to some desirable state, for the research to have been successful, the design science research must do so by solving an unsolved problem in a novel and innovative way and/or solving an already solved problem more efficiently [8].

The design science research process consists of six activities, as described above [7]: problem identification and motivation is about defining the specific problem being solved and showing that a solution is needed; objectives of a solution are inferred from the problem definition and can be quantitative or qualitative; design and development is the construction of the artifactual solution and the constructs, models, methods, and instantiations used to construct it; demonstration is the proof of the utility of the artifact to solve the problem using experimentation, simulation, case study or proof; evaluation is observing or measuring the performance of the artifact; and communication is presenting the problem, artifact, and its usefulness to researchers and practitioners [7].

3.2 Design Objectives

Based on the analysis of the problem and the literature review, five design objectives or properties (DOs) are defined for the framework. DO1 (Intent Preservation): The framework shall capture the analytical intent irrespective of the syntax or semantics of the execution engine. DO2 (Engine Independence) requires that two semantic representations be equivalent regardless of engine replacement; DO3 (Deterministic Behavior) requires that the framework specifies deterministic semantics when engines exhibit inconsistent behavior. DO4 (Validation Capability): can the semantic equivalence be checked without an output comparison? DO5 (Practical Applicability): is it practical to implement the required analysis using the analytics infrastructure without modifying the engine?

3.3 Evaluation Approach

Framework evaluation is commonly assessed with the design science research evaluation methods. The Framework for Evaluation in Design Science Research (FEDS) claims that evaluation is an important part of design science research. Researchers are required to show that design artifacts are useful, of high quality, and effective, using well-documented ways of evaluating [9]. FEDS characterizes evaluation episodes in terms of two dimensions: functional purpose, formative or summative; and model of evaluation, artificial or naturalistic [9]. The Human Risk and Effectiveness evaluation framework was used in this study. According to the framework, formative evaluations should be used earlier in the evaluation process and summative evaluations later, when design risks are mainly social or user risks [9].

4. Semantic Consistency: Formalization and Divergence Sources

4.1 Formal Definitions

Formal definitions of the concepts are introduced to make the theoretical foundation of the proposed framework precise. The analytical intent of a query is defined as the expression of logical meaning that is independent of the syntactic form and the execution environment. It specifies what to compute, but it does not specify how to compute. Execution semantics describe how the engine interprets a query and computes its result. It covers things like type resolution, operator precedence, NULL values, and aggregations and can differ across engines. A distributed analytics system satisfies the definition of semantic consistency if for any query Q and any data set D , the results of all the engines that are supported are equivalent in the sense that $E_i(Q, D) \sim E_j(Q, D)$. By contrast, the equivalence relation must be defined explicitly at the level of the platform.

4.2 Equivalence Criteria

Depending on this classification of analyses, different semantic equivalences apply [10]. For example, two tuples can be compared to be semantically equivalent if they contain the same tuples in the same order. Order-independent equality allows repeated identical tuples in output, while numerical tolerance equality allows numerical values with some tolerance to be considered equal. Logical equivalence is satisfied when the logical predicates are satisfied, and the required type of equivalence condition depends on the specific application. Financial reporting, for example, may require exact equality, while exploratory analytics may allow for tolerable approximate equality.

4.3 Sources of Semantic Divergence

The characteristics and requirements of the physical and logical dimensions of data are very independent in nature, as is their combination in complex queries. The classic DBMS architecture was established for

the processing of business data but has been applied to data applications with a wide range of characteristics and requirements. This is not directly applicable to other database markets [11].

Different types and implicit coercions between them are implemented by query engines. Thus, a query that compares a string with a numeric field may be accepted with implicit coercion by one engine, cause a type mismatch error in another or return a different result depending on the coercion direction. The data structures of warehouse systems differ from those in an OLTP system. A warehouse system generally uses a bit-map index rather than a B-tree [11]. Some warehouse systems support arbitrary precision for decimal arithmetic, while others convert to floating-point numbers, still supporting some sort of precision. Although SQL null semantics follow three-valued logic in principle, implementations differ in their treatment of null in boolean expressions, its sorting order, handling in aggregate functions, and usage in arithmetic and equality comparisons [10]. Null semantics are fundamental to relational operations and propagate through query evaluation.

Distributed aggregation can bring about the non-associativity of floating-point addition, such as $(a + b) + c \neq a + (b + c)$ in floating-point arithmetic [12]. This is because parallel execution engines can choose to compute aggregates in various order based on how input keys are partitioned and executors allocated, and these can differ at precision boundaries. While individual differences are small, when summed over a large data set they exceed the threshold.

Query optimizers apply algebraic laws to rewrite a query into a more performant form. While these rewrites are logically equivalent, such equivalence only holds for an idealized version of relational algebra and does not necessarily hold for non-deterministic functions, floating-point arithmetic, and/or non-commutative operations. The difference between these two contexts, called training-serving skew in machine learning pipelines, can be modeled as different feature spaces, different value encodings, and different value distributions [1].

4.4 The Detectability Paradox

Some of these semantic divergences may be subtle: near-correct results are returned, the divergence is relatively small, and comparisons between engines to find divergences are rarely performed [11]. In many cases, a data processing validation failure may occur because of faulty programs for collecting and transforming the data, miscommunication of the teams that developed the different components of the data processing pipeline, or in production due to the deployment of binaries with different versions [1]. The divergence of the engine outputs, denoted Δ , is typically ignored if it is smaller than informal limits described for Δ . However, these cumulative deviations have been known to exceed tolerances, especially at points of high reproducibility.

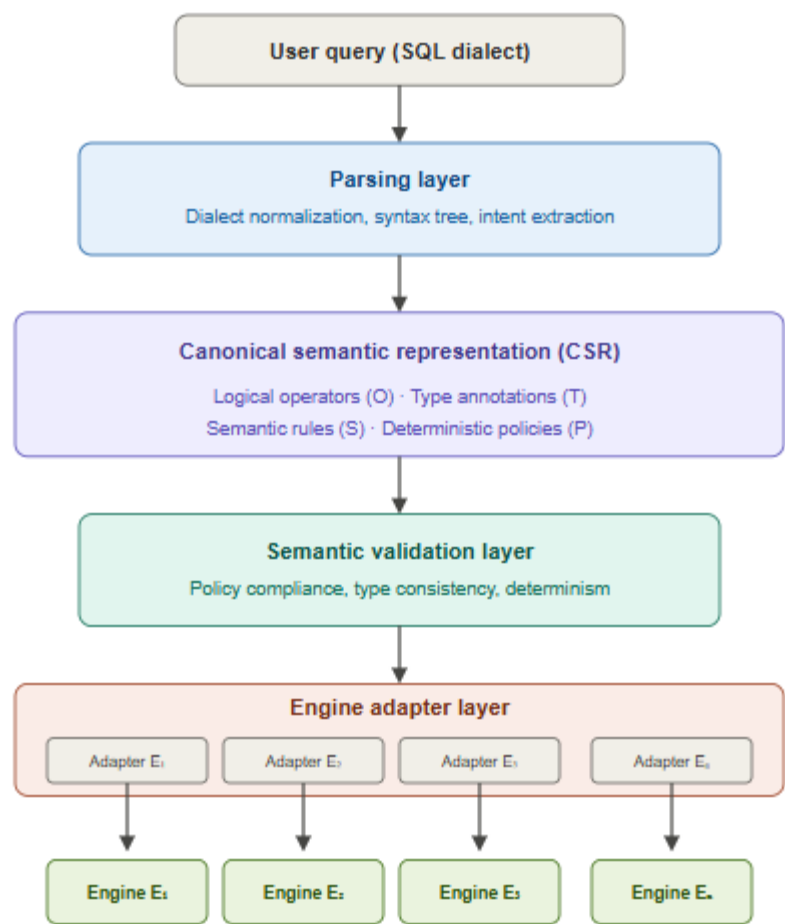
Divergence Category	Description	Impact on Query Results
Type systems and coercion	Query engines implement varying type systems with different implicit casting rules and numeric precision handling	A query comparing string to numeric field may succeed in one engine, fail in another, or produce different results depending on coercion direction
Null semantics	Variations in three-valued logic evaluation, null ordering in sort operations, and null participation in aggregate functions	Foundational variations propagate throughout query evaluation, affecting comparisons, aggregations, and result ordering
Floating-point aggregation	Non-associativity in floating-point operations where $(a + b) + c \neq a + (b + c)$; parallel engines evaluate in varying orders	Results differ at precision boundaries; cumulative effects across large datasets may exceed analytical tolerance thresholds
Optimizer transformations	Algebraic transformations applied by query optimizers to improve execution efficiency	May alter results when queries involve non-deterministic functions, floating-point arithmetic, or operations sensitive to evaluation order
Workload-specific optimization	Engines optimized for different workloads (OLTP vs. warehouse) use different indexing and processing strategies	Same query may produce different intermediate representations and execution paths, potentially affecting edge-case behaviors

Table 1: Sources of Semantic Divergence [1, 11]

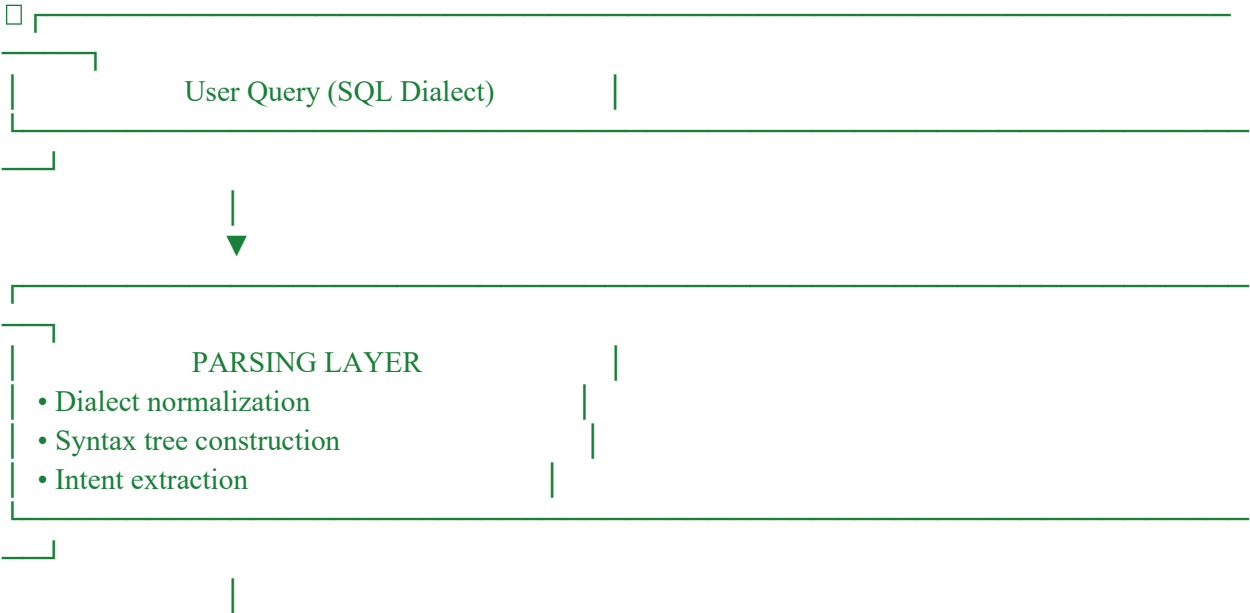
5. Canonical Semantic Representation Framework

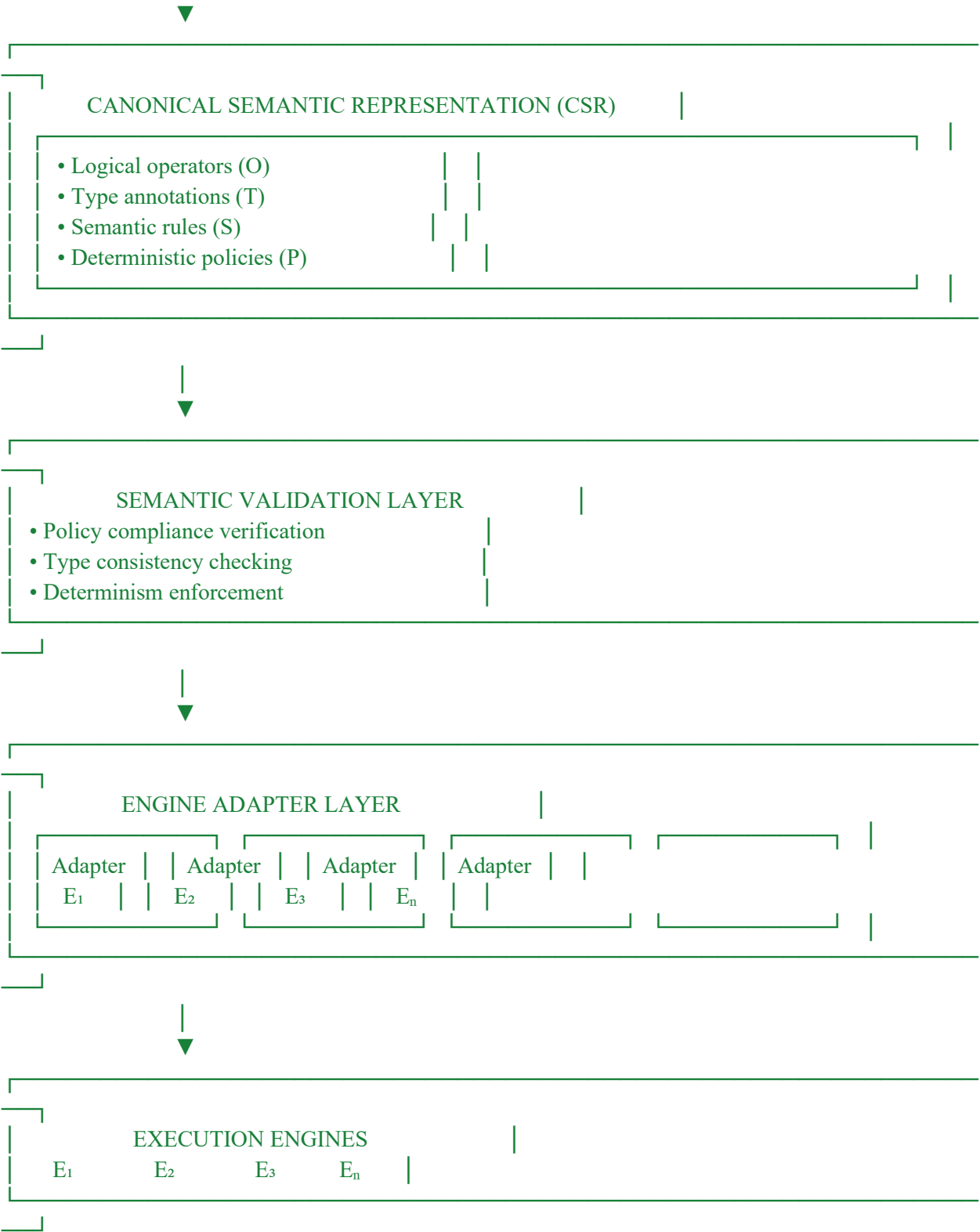
5.1 Architectural Overview

To ensure semantic consistency, we adopt a three-layer architecture that separates analytical intent from execution semantics. The architecture is shown in Figure 1 (layers are separated) .



[Figure 1: Canonical Semantic Representation Framework Architecture]





□

5.2 Canonical Semantic Representation

The Canonical Semantic Representation (CSR) specifies the intentions of the analysis. It can be shown as $CSR(Q) = G(O, T, S, P)$, where O are the logical relational operators in the DBMS, T the explicit mapping of types with their precision specifications, S the semantic rules for operator behavior, and P the deterministic operator policies. Thus, the CSR captures what the query computes, not how, allowing engine-specific optimizations while preserving the semantic meaning of the query.

In the CSR, a query is represented using a canonical operator algebra. The operator algebra is a relational algebra improved with aggregation, windowing, and set operations. Every operator is accompanied by a concrete type signature and semantic constraints. In contrast to engine-specific types, CSR uses a universal type system with explicit precision specifications [13]. Numeric types can specify precision and scale, temporal types can specify resolution, and string types can specify collation. These annotations provide a unified semantic understanding for all engines.

Semantic rules also specify differences between engines, such as how nulls are treated in comparisons and aggregates, the order of evaluation for nonassociative operators, the default value for optional parameters, or the treatment of boundaries in range operations. As a result of getting the primitives right at the lower levels of the engine rather than having to simulate that behavior using more conventional constructs, large performance differences can be achieved [11].

5.3 Engine Adapters

The translation function for each engine E_i is bidirectional, denoted T_i : $CSR(Q) \rightarrow Q_i$ where query Q_i is executable on E_i . The adapters also respect the semantic meaning of the canonical queries while translating them into the syntax of individual engines. Adapters may also insert operations to comply with semantic policies such as type-casting to preserve precision or sorts to determine a particular order of results.

The adapter architecture flips this relation: engines are no longer a semantic anchor; instead, they are the target of converting a canonical representation into a specific engine representation. This is similar to the distinction between an external representation and an internal representation, where the former has a semantic contract while the latter can rely on a common context [14].

5.4 Semantic Validation

The semantic property of the framework is that if $CSR(Q_1) = CSR(Q_2)$ then Q_1 and Q_2 are semantically equivalent. This property can be used to check that query refactorings are semantically preserving, that dialect translations are semantically preserving and to detect semantic drift of queries over time.

5.5 Deterministic Semantic Policies

For operators that induce nondeterministic behavior in the engine, deterministic policies are defined that guarantee semantic consistency [14]. Table 1 shows example policies for numeric precision, ordering of nulls, aggregation of nulls, aggregation of floating-point values, order of window functions, and timestamp precision.

Divergence Category	Policy Specification	Enforcement Mechanism
Numeric precision	Fixed decimal precision for financial calculations	Type annotation in CSR
Null sorting	Nulls sort last in ascending order and first in descending	Adapter-injected sort criteria
Aggregate null	Nulls excluded from all aggregates except	CSR semantic rule

handling	COUNT(*)	
Floating-point aggregation	Deterministic summation algorithm for reproducible results	Adapter-generated procedure
Window function ordering	Deterministic tiebreaker using row identifier	Adapter-injected ORDER BY clause
Timestamp precision	Microsecond resolution with explicit timezone	Type annotation in CSR

Table 2: Deterministic Semantic Policies [1, 11]

5.6 Theoretical Properties

A rigorous definition of a semantic consistency framework includes three properties: for every representation, transformation of the representation to engine-specific format and back to CSR should generate the same representation (Idempotent translation). Property 2 (Engine Independence) ensures that the CSR remains the same when substituting an engine, while Property 3 (Upgrade stability) guarantees that the semantics of the CSR do not change with the version of the engine, unless one changes the policy.

6. Evaluation

6.1 Evaluation Methodology

Framework evaluation follows a protocol of structured expert assessment, one of the standard methods for evaluating design science artifacts. The Framework for Evaluation in Design Science Research states that summative evaluations must create consistent interpretation across shared meanings and that formative evaluations must provide a basis for successful action [9]. The evaluation is guided by the steps of the FEDS process: explicating the goals of the evaluation, selecting the evaluation strategy, determining the properties to be evaluated, and designing single evaluation episodes [9].

Evaluation seeks to determine that with a high level of certainty, the artifact achieves the desired effect (efficacy) and works in the environment in which it will be deployed (effectiveness). Artificial evaluation increases scientific rigor through higher levels of repeatability and falsifiability, while naturalistic evaluation increases internal validity through observing real-world use contexts. The evaluation design needs to balance these considerations, and so it may start with formative artificial evaluations and end with summative naturalistic evaluations.

6.2 Evaluation Design

Experimental participants were recruited from professional networks of systems architects in multi-engine analytics platforms. Participants were systems architects with wide-ranging experience in distributed systems, including practical experience of implementing multi-engine analytics. Participants were provided with a framework specification document containing the framework's architectural model, formal definitions, policy examples, and implementation details. Participants reviewed the specification followed by the assessment tool to assess how well the framework achieved the five design objectives.

The evaluation tool consisted of sections with items for each design goal, including Likert scale items for agreement with statements about framework properties. From a functional perspective, the aim of this evaluation was summative, namely to judge how well the results meet expectations [9]. This procedure combined an empirically constructed evaluation tool with clinical judgment based on real-world experience.

6.3 Results

The results of this evaluation suggest that both the framework design and the objectives described are congruent. For instance, for Intent Preservation (DOI), there is a high agreement rating, which suggests that the CSR concept addresses the problem of representing analytical meaning independently of an

10.48047/jocaaa.2026.35.04.16

execution context. Engine Independence (DO2) is feasible and sound. Experts agree that all semantic representations remain invariant under engine substitution. Deterministic Behavior (DO3) is feasible and sound since experts confirm that proposed policies address the major sources of divergence between engines.

Experts gave a positive score to Validation Capability (DO4) and stated representation-level validation can reduce the need for output comparison. Practical Applicability (DO5) received the widest variation in score due to its implementation. Qualitative observations indicate that the costs of defining new adapters and maintaining the implementation are also important practical considerations due to the large surface area of SQL that must be mapped and, for example, the differing engine behavior across versions.

6.4 Qualitative Findings

A thematic analysis of the practitioners' feedback on the framework identified its strengths as formalizing semantic consistency as an architectural property, as well as handling semantic consistency as an invariant rather than an assumption customarily made implicitly, thus catering to practitioners who had witnessed cross-engine divergence occurring in production environments. The cons included costs of adapter implementation and maintenance, as well as making changes to adapt to the engine's new versions. Suggested extensions to the adapters included support for machine learning operations (MLOps) beyond SQL, integration with data lineage tools, and policy generation based on query history analysis.

This shows that the framework satisfies criterion 4 (artifact utility). There are still aspects that need to be finalized before common usage. However, the expert assessment also shows that the framework solves a real problem with a principled architectural solution [9].

Design Objective	Evaluation Criterion	Expert Assessment Finding	Implementation Consideration
DO1: Intent preservation	CSR captures analytical intent independent of execution engine syntax or semantics	Strong agreement: The CSR concept effectively addresses core challenge of representing analytical meaning	Requires comprehensive operator coverage across SQL dialects
DO2: Engine independence	Semantic representations remain invariant under engine substitution	Favorable assessment; experts confirmed representation stability across engine changes	Adapter maintenance burden increases with number of supported engines
DO3: Deterministic behavior	Framework specifies deterministic semantics for operations with divergent engine behavior	Positive validation: proposed policies address primary sources of cross-engine divergence	Policy completeness depends on comprehensive divergence cataloging
DO4: Validation capability	Framework enables semantic equivalence verification without output comparison	Acknowledged utility; representation-level validation reduces reliance on result reconciliation	Formal verification tooling required for production deployment
DO5: Practical applicability	Framework implementable within existing analytics infrastructure without engine modifications	Varied assessment; implementation complexity concerns noted by multiple experts	Adapter development effort and version tracking represent significant overhead

Table 3: Evaluation Summary by Design Objective [9]

7. Discussion

7.1 Implications for Practice

Such a framework influences the design of analytics platforms. For example, semantic consistency should be considered a first-class architecture-level concern rather than an implicit assumption that is not explicitly addressed. In most cases, it is reasonable for organizations that use multi-engine analytics to define equivalence criteria and tolerance requirements based on their specific use cases. This is because data validity problems may occur due to a code error in the code that collects or transforms the data, an error in communication between people involved in different parts of the data pipeline, or production issues [1].

The CSR approach builds architectures with an intermediate semantic layer, mediating between user-facing query interfaces and engine implementations [15]. This layer provides semantic governance, policies, and cross-engine validation without needing to modify the engine implementations. This would allow organizations to stepwise redress semantic divergence using the deterministic policy approach, tackling sources of divergence with the most impact on their analytics.

7.2 Implications for Research

This work lays the foundations of semantic interoperability in distributed environments: the definitions of analytical intent, execution semantics, and semantic consistency establish a lexicon for discussing cross-engine equivalence that up to now has never been described formally. From a design science perspective, the paper positions itself as the beginning of a method for how to design and evaluate architectural frameworks when no practical implementations are available. Expert evaluation is a more systematic approach to validating artifacts addressing emerging architectural patterns based on design science research frameworks [9].

7.3 Limitations

The study has several limitations, and it also states it recommends further work, mainly that the framework has not been deployed in production. The architecture is conceptually sound and has been shown to be practically useful in expert analysis, but there may be problems in practice not anticipated by the architecture specification itself. The scope is SQL-like query languages. Other extensions of procedural analytics, machine learning pipelines, and streaming computations are theoretically possible, but may require more fundamental extensions to the basic CSR model.

The expert sample was relatively small yet highly skilled and diverse and was taken from professional networks. Further validation using other practitioner samples is therefore warranted. The framework does not consider the performance overhead of implementing its semantic constraints, which can result in slower query execution and greater resource usage. As specialized stream processing engines achieve performance gains of more than 2 orders of magnitude compared to relational database engines for certain workloads, performance effects must be factored into the design of semantic enforcement mechanisms, which can only be evaluated experimentally at the implementation level [11].

Conclusion

Heterogeneity in execution engines is bound to be an architectural reality, given the prospects for distributed analytics ecosystems. Heterogeneity of semantics on the other hand is not an architectural given. In this work we have further formalized semantic consistency as an architectural property and represented it in the Canonical Semantic Representation for preservation of analytical intent in heterogeneous execution environments.

10.48047/jocaaa.2026.35.04.16

The three layers in the architecture, namely parsing, semantic validation and engine adaptation, separate between the meaning of the analysis and the mechanics of execution. User studies with experts on distributed system architectures confirmed that the framework met five design objectives and met the design science criterion of the utility of an artifact. It provides precise definitions of analytical intent, execution semantics and semantic consistency, which are prerequisites to a sound treatment of cross-engine equivalence, as well as an architecture that organizations can use to manage the semantic behavior of their multi-engine analytics deployments.

Future research should address experimental production implementation, extensions to other analytics models, and links to other data governance mechanisms. The increasing diversity of analytics architectures means that semantic conformance between different data sources and a system's ability to diagnose and recover from silent analytical divergence will differentiate future enterprise analytics platforms.

References

- [1] Neoklis Polyzotis et al., "Data Management Challenges in Production Machine Learning," SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data (May 2017) [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/3035918.3054782>
- [2] Michael Armbrust et al., "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," 11th Annual Conference on Innovative Data Systems Research (CIDR '21), January 11–15, 2021. [Online]. Available: <https://15721.courses.cs.cmu.edu/spring2023/papers/02-modern/armbrust-cidr21.pdf>
- [3] AMIT P. SHETH and JAMES A. LARSON, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," ACM Computing Surveys (CSUR), Volume 22, Issue 3 (September 1990). [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/96602.96604>
- [4] Zachary G. Ives et al., "An Adaptive Query Execution System for Data Integration," ACM, 1999. [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/304181.304209>
- [5] Fabian Neuhaus, "What is an ontology?" arXiv, 2018. [Online]. Available: <https://arxiv.org/pdf/1810.09171>
- [6] Mattia Tritto et al., "GradeSQL: Test-Time Inference with Outcome Reward Models for Text-to-SQL Generation from Large Language Models," arXiv, 2025. [Online]. Available: <https://arxiv.org/pdf/2509.01308>
- [7] Ken Peffers et al., "THE DESIGN SCIENCE RESEARCH PROCESS: MODEL FOR PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH," arXiv. [Online]. Available: <https://arxiv.org/pdf/2006.02763>
- [8] Guido L. Geerts, "A design science research methodology and its application to accounting information systems research," International Journal of Accounting Information Systems, 2011. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S1467089511000200>
- [9] John Venable et al., "FEDS: a Framework for Evaluation in Design Science Research," European Journal of Information Systems, [Online]. Available: <https://www.tandfonline.com/doi/pdf/10.1057/ejis.2014.36>
- [10] Jan Michels et al., "The New and Improved SQL:2016 Standard," SIGMOD Record, June 2018 (Vol. 47, No. 2) [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/3299887.3299897>

10.48047/jocaaa.2026.35.04.16

- [11] Michael Stonebraker and Ugur Çetintemel, "“One Size Fits All”: An Idea Whose Time Has Come and Gone," Proceedings of the 21st International Conference on Data Engineering (ICDE 2005) [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1410100>
- [12] Vincent Lafage, "Revisiting 'What Every Computer Scientist Should Know About Floating-point Arithmetic,'" arXiv, 2020. [Online]. Available: <https://arxiv.org/pdf/2012.02492>
- [13] Koustuv Sinha et al., “Compositional Language Understanding with Text-based Relational Reasoning,” arXiv, 2018. [Online]. Available: <https://arxiv.org/pdf/1811.02959>
- [14] PAT HELLAND, "Data on the outside versus data on the inside," COMMUNICATIONS OF THE ACM, 2020. [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/3410623>
- [15] Sayed Hoseini et al., "A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes," Journal of Web Semantics, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570826824000052>