Categorical Logic for Verifying Query Correctness and Data Integrity in Relational Databases*

Oleh Ilarionov[1,*,†], Hanna Krasovska[1,†], Hryhorii Hnatiienko[1,†] and George Gaina[1,†]

[1] *Taras Shevchenko National University of Kyiv, st.Volodymyrivska, 60, Kyiv, Ukraine*

Abstract

This paper develops a categorical framework for jointly verifying query soundness and referential integrity in relational databases. A database schema is represented as a small category whose objects correspond to tables and whose morphisms represent attribute projections and foreign keys. Database instances are interpreted in Par(Set), which allows NULL values and broken references to be modeled as partial functions rather than by introducing an additional logical value. For a selected fragment of conjunctive SQL queries, query semantics is represented by Set-valued profunctors. Query soundness is characterized by the existence of a natural transformation from the query profunctor to the corresponding attribute-projection profunctor, ensuring compatibility between projected attributes, join paths, and the schema structure. Referential integrity is formulated through totality, functionality, and membership of foreign-key images in the relevant primary-key sets. The approach is illustrated on an RBAC schema with one-to-many and many-to-many relationships and implemented as a prototype in Catlab.jl. The present scope is limited to conjunctive queries with equality predicates, while richer SQL constructs and large-scale empirical evaluation are left for future work.

Keywords

categorical logic, relational databases, query verification, data integrity, profunctors, commutative diagrams, formal methods

1. Introduction

Modern relational databases form the foundation of information systems in many critical sectors, including the financial sector, healthcare, e-commerce, and scientific research. The growth of data volumes, the increasing complexity of schemas, and the proliferation of distributed architectures raise the bar for ensuring the correct execution of queries and the preservation of data integrity. Errors in queries or violations of referential integrity can lead to logical anomalies, data inconsistencies, and significant risks to information security.

Traditional methods for ensuring query correctness and data integrity are based primarily on relational algebra, logical inference (in particular Datalog), constraint checking at the trigger and DBMS mechanism levels, or query containment analysis. Although these approaches are effective for certain classes of problems, they have certain conceptual limitations when dealing with complex schemas..

First, relational algebra and trigger mechanisms ensure the verification of syntactic correctness and compliance with local constraints, but do not always provide formal tools for proving the semantic correctness (soundness) of a query with respect to the entire schema structure.

Second, Datalog-based methods are effective for recursive queries and logical inference, but are not oriented toward structural modeling of the schema as a holistic object or compositional analysis of many-to-many relationships in terms of categorical constructions.

*ITS-2025: Information Technologies and Security, December 11, 2025, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

oleg.ilarionov@knu.ua (O. Ilarionov); hanna.krasovska@knu.ua (H. Krasovska); hnatiienko@knu.ua (H. Hnatiienko) and ggaina@knu.ua (G. Gaina)

0000-0002-7435-3533 (O. Ilarionov); 0000-0003-1986-6130 (H. Krasovska); 0000-0002-0465-5018 (H. Hnatiienko) and 0000-0003-0260-0950 (G. Gaina)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

CEUR-WS.org/Vol-4337/paper12.pdf

Third, query containment approaches allow for verifying the inclusion or equivalence of queries, but operate primarily at the level of formal expressions and do not integrate the semantic properties of the schema (in particular, complex dependencies and cyclic foreign keys) into a unified verification model. Furthermore, modern verification tools, such as Cosette (for query equivalence using SMT) or Q*Cert (certification in Coq), focus on equivalence but do not integrate referential integrity with soundness into a unified categorical framework. Similarly, HoTTSQL provides Coq proofs of SQL semantics with NULL, VeriEQL provides SMT for bounded equivalence with FK/NULL, and UML-B/Event-B provides integrity proofs via refinement.

Finally, functional database models (particularly within the Categorical Query Language) provide a rigorous categorical description of schemas and transformations between them; however, the question of combining structural integrity and the verification of specific queries within a single algorithmic mechanism remains a subject for further research.

These limitations become particularly relevant in complex schemas with a large number of tables and foreign keys, where partial testing or local checks do not guarantee global consistency between the schema structure and query results. The absence of an integrated formal framework that would simultaneously cover the schema structure, query semantics, and integrity assurance mechanisms justifies the need for further development of categorical verification methods.

The aim of this research is to construct a unified category-logical framework for the formal verification of query semantic correctness (soundness) and the preservation of referential integrity in relational databases based on profunctor semantics and bicategorical modeling, taking into account partial functions in Par(Set) for NULL values and potential violations. The proposed approach aims to integrate structural schema analysis, categorical query interpretation, and algorithmic verification of the commutativity of the corresponding diagrams, which creates the basis for constructing a mathematically rigorous and potentially automated toolkit for the static verification of relational systems.

2. Review of Previous Research

The application of categorical logic to relational databases began with the functorial semantics of schemas [1] and was further developed in Categorical Query Language (CQL) [2], algebraic models [3], sketch formalism [4], compositional structures [5], and topos semantics [6, 7]. These approaches allow for a rigorous description of schemas, constraints, and data transformations; however, they do not fully solve the problem of simultaneously verifying the semantic correctness (soundness) and referential integrity of queries for complex schemas with “many-to-many” relationships.

The functorial approach to databases was systematically developed in the work of D. I. Spivak [1], where a schema is viewed as a small category and an instance as a functor into the category of sets. Data migration operations are formalized via adjunctive constructions, which provides a rigorous semantic description of transformations between schemas and lays the foundation for a categorical analysis of queries.

Further development of this direction is presented in [2], where the Categorical Query Language (CQL) is proposed as a practical implementation of functorial data integration. CQL uses categorical constructions to align schemas and supports automated generation of SQL queries. At the same time, the focus is on data migration and integration, while the issue of formal verification of individual queries is considered indirectly.

An algebraic generalization of this approach is presented in [3], where schemas are interpreted as algebraic theories and instances as their models. Such an interpretation provides a rigorous description of dependencies and constraints in terms of universal properties; however, the issue of algorithmic verification of the semantic correctness of specific queries remains open.

The sketch approach presented in [4] formalizes ER models via finite-limit sketches. Integrity constraints are encoded as commutative diagrams, and updates are described via the bi-category of spans. This approach provides a rigorous structural description of the schema, but does not offer an integrated mechanism for verifying the correctness of the results of a specific query with respect to the current state of the database.

Another approach to formalizing queries was proposed by D. I. Spivak [8] in his work on lifting problems, where the execution of a query is interpreted as a lifting problem in the corresponding category. This approach reduces the correctness check to an analysis of the commutativity of diagrams and factorizations of morphisms . At the same time, this interpretation focuses primarily on the logical aspect of constraints and does not integrate compositional modeling of complex relations.

From the perspective of classical database theory, the results [9] on the equivalence and inclusion of conjunctive queries are fundamental. Their work laid the theoretical foundation for the query containment problem, which remains a central tool for the formal verification of query equivalence. However, this approach operates primarily at the level of syntactic representations of queries and does not model the schema structure as a categorical object, which limits the possibility of integrated analysis of structural and semantic properties. Research on the formal verification of query equivalence and inclusion has been further developed to account for integrity constraints. In particular, the analysis of query containment in the presence of tuple-generating dependencies (tgds) and equality-generating dependencies (egds) is performed using chase-and-backchase algorithms [10]. Based on these results, a number of tools for automatic query equivalence verification have been created: Cosette [11] uses SMT solvers to verify the equivalence of SQL queries, HoTTSQL [12] provides mechanized proofs in Coq of SQL semantics accounting for NULL values, VeriEQL [13] supports bounded equivalence checking in the presence of integrity constraints (FK, NULL), Mediator [14] focuses on verifying the equivalence of database-level applications, and UML-B/Event-B methods [15] allow proving integrity through incremental refinement of models followed by SQL code generation. At the same time, most of the approaches mentioned focus on equivalence checking or query optimization, whereas the integration of semantic soundness with referential integrity within a single formalism, especially for complex “many-to-many” relationships, remains under-researched.

The compositional dimension of categorical semantics is explored in [5], where monoidal categories and profunctors are used to describe compositional processes. Such constructions naturally generalize “many-to-many” relationships, similar to those arising in the RBAC model (see Figure 1).

The theoretical foundation for these constructions is the bicategorical formalism introduced in [16], which generalizes categories of relations and profunctors and creates an abstract framework for describing the composition of relations. Despite the fundamental nature of this approach, its application to the algorithmic verification of relational systems remains limited.

The logical foundations of the categorical formalism are systematically presented in [6] and [7], which consider sub-object classifiers, the internal logic of categories, and topological semantics. These results provide a theoretical foundation for the formalization of predicates and partial relations; however, their direct application to the verification of relational queries requires additional constructive interpretation.

To generalize the approaches discussed and systematize their characteristics in the context of modeling complex relationships, query verification, and automation capabilities, a comparative analysis is presented in Table 1.

The table demonstrates that different approaches focus on specific aspects - schema structural correctness, logical equivalence of queries, or data integration - but do not combine these properties into a single formal mechanism. The current literature provides a range of powerful tools for describing database schemas, integrity constraints, and data transformations: functorial

semantics of schemas and migrations [1–3], sketch formalism for structural constraints [4], compositional constructs based on profunctors and monoidal categories [5], as well as topological semantics for internal logic [6, 7].

At the same time, existing approaches are largely focused on individual aspects: structural schema modeling, logical equivalence of queries, or data integration. Integrating these aspects into a single formalism that would allow for the simultaneous verification of query semantic correctness (soundness) and referential integrity - especially for complex schemas with “many-to-many” relationships—remains an open problem.

Table 1:
Comparative analysis of approaches to modeling and verifying relational databases.

Approach	Schema Model	N:M model	Query verification	Automation
CQL	Schema as a category	Via an explicit intermediate table	Via functor equivalence	SQL generation
Datalog	Logical rules	Naturally (via rules)	Logical correctness of inference	Prolog, Soufflé
Query Containment	No schema model	Not considered	Inclusion/equivalence checking	Partially
Sketch-based	Finite-limit sketches	Via diagrams	Verification of circuit constraints	No tool
Cosette / HoTTSQL	No schema model	Not supported	SMT/HoTT for equivalence/	Coq/ Agda
VeriEQL	No schema model	Not considered	SMT for bounded equivalence with FK/NULL	SMT solvers
Mediator	No schema model	Not considered	SMT for application-level equivalence	SMT
UML-B/ Event-B	Sketches/ clarifications	Via diagrams	Proving integrity through refinements	Rodin/SQL generation

3. Categorical Model of a Relational Schema

To illustrate typical structural features of relational systems, we will use a comprehensive example: a Role-Based Access Control (RBAC) system schema. This example was chosen due to its industrial relevance, the presence of two independent “many-to-many” relationships, and the need for strict verification of access rights, which perfectly illustrates the limitations of the functor approach and the advantages of profunctor semantics. Figure 1 shows a classic ER diagram of the RBAC model.

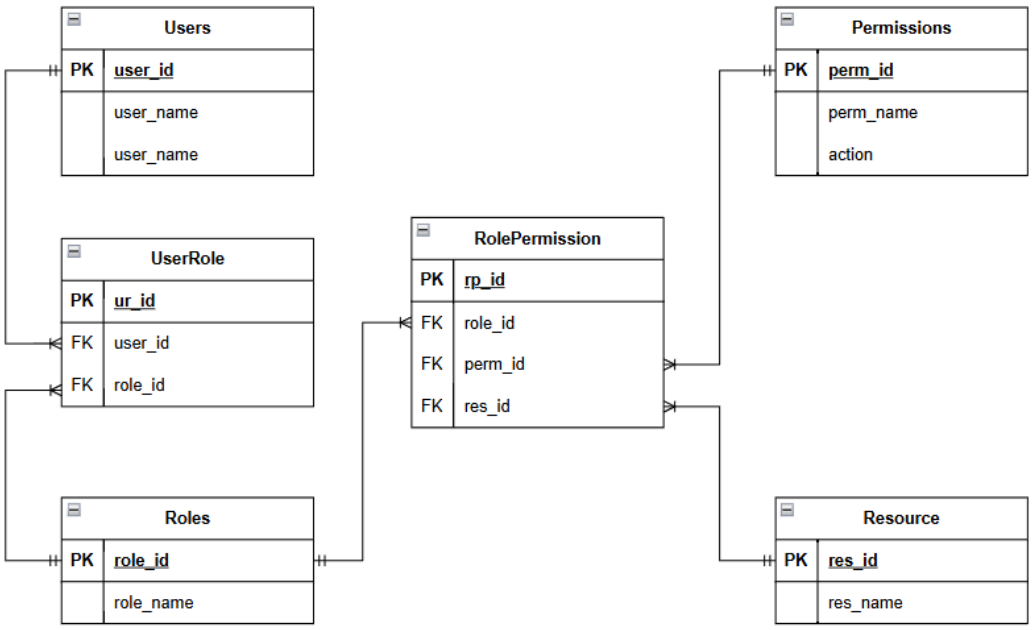


Figure 1: ER diagram of the RBAC model illustrating "many-to-many" relationships via intermediate tables

3.1. Category representation of the schema

We model the relational schema using the small category S. Objects of the category S correspond to database tables, and morphisms correspond to attribute projections and foreign keys.

Let $Par(Set)$ be a category of sets and partial functions. The functor of instances $I: S \to \{Par\}$ maps each object $A \in S$ to the set $I(A)$ of records in table A, and each morphism $f: A \to B$ to a partial function $I(f): I(A) \rightharpoonup I(B)$.

The value of $I(f)(a)$ is equal to the foreign key of the record $a \in I(A)$ or is undefined $(\perp)$ if the value is NULL.

The attribute functor is defined by two specialized variants:

1. $Attr_{input}: S^{op} \to Set$ - attributes of input tables (FROM, JOIN),
2. $Attr_{res}: S_{res} \to Set$ - attributes of the resulting projection (SELECT), where S_{res} is a subcategory of objects participating in the result set of the query.

Query Q is represented by a profunctor $Q: S^{op} \times S_{res} \to Set$ in the bicategory $Prof(Par(Set))$. We work within the standard bicategory $Prof$, whose objects are small categories, 1-cells are profunctors $P: C \nrightarrow D$ (that is, functors $P: D^{op} \times C \to Set$ with values in Set), and 2-cells are natural transformations between them. The instance functor $I: S \to Par(Set)$ is used exclusively for modeling database states and referential integrity; it does not alter the Set-valued structure of profunctors. Accordingly, the query profunctor $Q: S^{op} \times S_{res} \to Set$ and the attribute functors $Attr_{input}$, $Attr_{res}$ all take values in Set. The notation $Prof(Par(Set))$ used elsewhere in the text refers to this bicategory $Prof$ applied to schemas interpreted via $Par(Set)$, not to profunctors with values in $Par(Set)$.

The tensor product of profunctions is defined by the coend:

$$(Q \otimes R)(x, z) = \int^{y} Q(x, y) \times R(y, z)$$

The criterion of soundness of the query is the existence of a natural transformation

$$\eta : Q \Rightarrow Attr_{input}^{op} \otimes Attr_{res} \tag{1}$$

We formulate referential integrity for the morphism $f : A \to B$ as the totality and functionality of the partial function $I(f)$, together with the condition $\forall a \in I(A) : I(f)(a) \in PK(I(B))$, where $PK(I(B)) \subseteq I(B)$ is a subset of the primary keys of table B.

Example. In the RBAC schema, the objects are the tables *User, Role, Permission, UserRole, and RolePermission.* Morphisms such as *user_of_ur, role_of_ur, permission_of_rp*, etc., implement foreign keys between these tables (Figure 2).

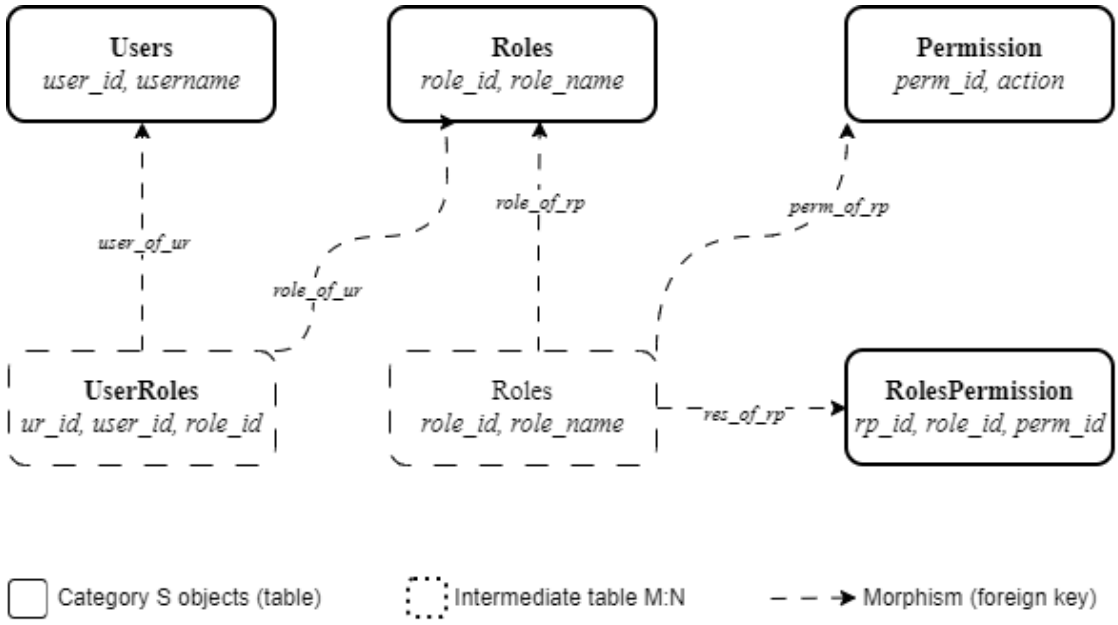


Figure 2: Categorical RBAC schema: objects S are tables, morphisms are foreign keys. The solid tables UserRole and RolePermission are denoted by a dashed border.

3.2. Referential integrity via pullback diagrams

Referential integrity requires that every foreign key in a database refer to an existing record in the target table. In the Set category, a foreign key is modeled as a regular function $f : A \to B$. However, for real-world databases where NULL values or integrity violations are allowed, the $Par(Set)$ category is more natural, in which f is a partial function: $I(f) : I(A) \rightharpoonup I(B)$, defined only on the subset of records $I(A)$. The indeterminacy $I(f)(a)$ corresponds to a NULL value of the foreign key for the record a .

Integrity for the morphism $f : A \to B$ into $Par(Set)$ is expressed by three conditions: totality $dom(I(f)) = I(A)$, functionality (each $a \in I(A)$ corresponds to exactly one $b \in I(B)$), and the image belonging to the primary keys $\mathrm{Im}(I(f)) \subseteq PK(I(B))$.

To formally verify these conditions (figure 3), a pullback diagram is constructed along $I(f)$ and a canonical inclusion diagram $inc : PK(I(B)) \to I(B)$:

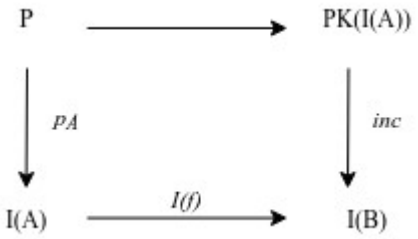


Figure 3: Pullback diagram along I(f) and inc

The object P consists of pairs (a,b) such that $I(f)(a)=b$ and $b \in PK(I(B))$. Integrity is preserved if and only if the projection $p_A: P \to I(A)$ is an isomorphism. If p_A is only a monomorphism, this indicates the presence of NULL values; if only an epimorphism, it indicates references outside $PK(I(B))$. In case of a violation, the set $I(A) \setminus \mathrm{Im}(p_A)$ explicitly indicates the violating records. The projection $p_B: P \to PK(I(B))$ is a monomorphism in all cases as a consequence of the universal pullback property along inc .

For the RBAC scheme, three foreign keys are checked: $r_{RP}: RolePermission \to Roles$, $s_{RP}: RolePermission \to Resources$, and $u_{UR}: UserRole \to Users$. For each of them, the algorithmic check consists of traversing the records $I(A)$, verifying the definiteness of $I(f)(a)$, the existence of $I(f)(a)$ in $PK(I(B))$, and the absence of duplicates. The complexity of the check is $O(/I(A)/)$ with indexing by primary keys.

Before formulating the main result, we establish the necessary technical construction that connects attribute functors with the bicategorical environment Prof(Par(Set)).

Lemma 3.1 (Induction of profunctors via companions).

Let $F_*: D^{op} \times C \to Set$ be the companion of the functor $F: C \to D$, defined as $F_*(y,x) := Hom_D(y, F(x))$.

The attributive functors $Attr_{input}: S^{op} \to Set$ and $Attr_{res}: S_{res} \to Set$ induce companions:

$$(Attr_{input})_*: Set^{op} \times S^{op} \to Set, (V,x) \mapsto Hom_{Set}(V, Attr_{input}(x))$$

$$(Attr_{res})_*: Set^{op} \times S_{res} \to Set, (V,z) \mapsto Hom_{Set}(V, Attr_{res}(z))$$

Both profunctors have a coherent common type with respect to the Set categories, which makes their tensor product $(Attr_{input})_* \otimes (Attr_{res})_*$ a type-correct profunctor of type $S^{op} \times S_{res} \to Set$ □

4. Topos Logic and Its Application to Query Verification and Data Integrity

Profunctor semantics enables the modeling of queries with M:N relationships [5]; however, to formally verify semantic soundness and referential integrity, a tool is needed to formulate and verify logical properties of the schema and data. This capability is provided by the internal logic of the topos.

Modeling is performed in the topos Set (with an extension to $Par(Set)$ for NULL values), where the instance functor $I: S \to Par(Set)$ maps the schema category S to the database state. The subobject classifier $\Omega = \{ \top, \bot \}$ is used to characterize subsets and predicates.

The natural transformation (1) verifies the correctness of the query: the commutativity of the diagram guarantees that the result of the attribute projection corresponds without loss or addition (Figure 3).

4.1. Subobject Classifier and Modeling of Referential Integrity

The category $Par(Set)$ is a regular category: it has finite limits, a core of equivalence pairs, and every morphism factors through a regular epi and mono. This provides a sufficient structure for formulating predicates and integrity constraints without resorting to the internal logic of the topos.

Predicates and subobjects in Par(Set). For an object $X \in Par(Set)$, a subobject is defined as the class of monomorphisms $m: S \to X$. The characteristic function of a predicate χ_P is defined not by a morphism $X \to \Omega$ (as in a topos), but directly by the subobject - the set of elements of X that satisfy P. Regular logic allows predicates to be formulated using $\exists$, $\wedge$, and equality, which is sufficient to describe the constraints of a relational schema.

Modeling NULL and partial functions. The partiality of the instance functor $I(f): I(A) \rightharpoonup I(B)$ naturally models NULL: the value of $I(f)(a)$ is undefined ($\perp$) if and only if $a \notin dom(I(f))$. This does not require a third truth value - a partial function is either defined or not.

SQL three-valued logic (3VL) with UNKNOWN arises only when comparing with NULL. Within the current model, all predicates φ are conjunctions of attribute equality predicates. For such queries, two cases suffice: either $I(f)(a)$ is defined and the equality holds, or $I(f)(a)$ is undefined and the tuple is excluded from the result. Thus, 3VL reduces to a partial function in $Par(Set)$ without introducing a separate UNKNOWN value.

Referential integrity. For a morphism $f: A \to B$ in a schema S, referential integrity is formulated as three conditions in $Par(Set)$ logic:

- Totality: $dom(I(f)) = I(A)$, i.e., $I(f)$ is a total function.
- Functionality: $I(f)$ is a function (not a multivalued mapping).
- Primary key membership: $\mathrm{Im}(I(f)) \subseteq PK(I(B))$, which is expressed as the subobject inclusion $\mathrm{Im}(I(f)) \to PK(I(B))$ in $Par(Set)$.

All three conditions are formulated using the tools of regular logic (via existence, equality, and subobject inclusions) without reference to the subobject classifier Ω.

4.2. Internal logic of the Set topos for formulating query properties

Using the internal logic of the topos Set, formulated in the previous subsection, as well as the bicategorical structure $Prof(Par(Set))$, we establish the main result of this work.

Definition 4.1 (Target SQL fragment). Consider the class *of conjunctive queries (CQ)* of the form SELECT $a_1, \ldots, a_k$ FROM $T_{1,\ldots}, T_n$ WHERE φ, where φ is a conjunction of attribute equalities (without NOT, OR, aggregates, or outer joins). Semantics — set semantics (without duplicates). NULL values are modeled via $Par(Set): I(f)(a) = \perp$ if the value is NULL.

Based on the fixed language fragment, we define the semantics it generates

Definition 4.2 (Denotational Semantics of CQ). For an instance I, the denotation of the query $Q(I)$ is defined as the set of tuples:

$$Q(I) = \{(a_1, \ldots, a_k) \mid \exists \text{ entries } t_1 \in I(T_1), \ldots, t_n \in I(T_n) : \varphi(t_1, \ldots t_n) \wedge a_i = \pi(t_j)\},$$

where $\pi_i(t_j)$ is the value of the i-th output attribute obtained by projection from the j-th record t_j of the corresponding table T_j. In particular, if a_i is taken from a column of table T_j, then $\pi_i(t) = t[col_i]$.

Given the denotation $Q(I)$, we can precisely formulate what it means for a query to be sound with respect to a schema.

Definition 4.3 (Soundness). A query Q is *sound* with respect to a schema S and an instance I if every tuple $r \in Q(I)$ obtained exclusively via schema morphisms: does not contain attributes outside of $Attr_{input}^{op} \cup Attr_{res}$ and does not violate the JOIN structure $\forall r \in Q(I)\ \exists \eta(x,z)(r) \in (Attr_{input}^{op} \otimes Attr_{res})(x,z)$.

These definitions allow us to formulate the soundness criterion as the existence of a natural transformation between profunctors, as stated in the following theorem.

Theorem 4.1 (Criterion of semantic correctness of a query). Let S be a small category of database schemas, $I: S \to Par(Set)$ be an instance functor, $Attr_{input}: S^{op} \to Set$ and $Attr_{res}: S_{res} \to Set$ be attribute functors, and let the query be represented by the profunctor $Q: S^{op} \times S_{res} \to Set$ in the bicategory $Prof(Par(Set))$.

A query Q is semantically sound with respect to attribute projection if and only if there exists a natural transformation where the tensor product $\otimes$ is defined by the coend:

$$(P \otimes R)(x,z) = \int^{y} P(x,y) \times R(y,z) \tag{2}$$

Proof.

$(\Rightarrow)$ Suppose that query Q is sound. Then, for every pair $(x \in S, z \in S_{res})$ and every $q \in Q(x,z)$, there is a unique way to obtain this string via the composition of the input attributes and the projection. This defines the component $\eta_{(x,z)}(q) \in (Attr_{input}^{op} \otimes Attr_{res})(x,z)$.

The naturalness of η is verified by an explicit naturalness square. For morphisms $f: x \to x'$ into S and $g: z \to z'$ into S_{res}, the commutativity of such a square is required (Figure 4):

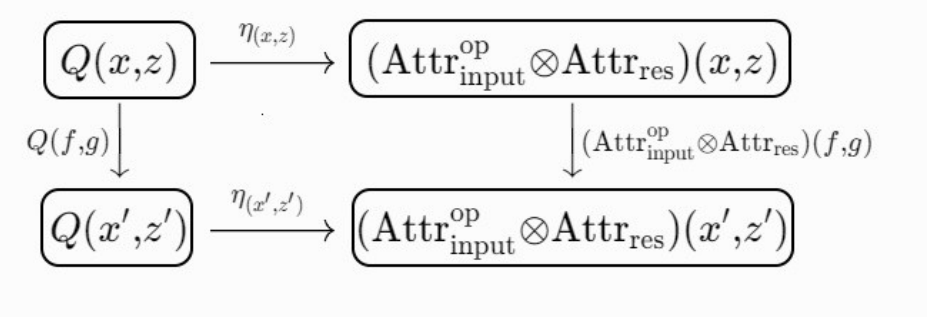


Figure 4: Commutative diagram of the natural transformation (1) .

Commutativity on the left: $\eta(x',z') \circ Q(f,g) = (Attr_{input}^{op} \otimes Attr_{res})(f,g) \circ \eta(x,z)$ follows from the fact that $Q(f,g)$ implements the substitution of attributes via the morphisms f (contravariantly in x) and g (covariantly in z), whereas the right side performs the same substitution via the coend. Both paths coincide by definition of the denotational semantics of CQ (Definition 4.2): JOIN corresponds to f, and the SELECT projection corresponds to g.

$(\Leftarrow)$Suppose a natural η exists. Let r be an arbitrary row resulting from the execution of Q on instance I. Then $r \in Q(x,z)$ for some x, z, and the component $\eta_{(x,z)}(r)$ uniquely maps r to an element of the tensor product. Since the tensor product via the coend precisely describes all correct projections of attributes, no row can appear outside the semantics of the schema. Thus, the query is sound.

Example 4.1 (Unsound query). Let $Q_{bad}: S^{op} \times S_{res} \to Set$ assign a singleton {*} to every pair $(u, p) \in I(Users) \times I(Permissions)$, ignoring the JOIN structure of the schema entirely. In the RBAC instance, the user $charlie$ holds only the $finance$ role, which carries no $full_{access}$. Nevertheless, $Q_{bad}(charlie, full_access) \neq 0$, while the attribute projection gives $A(charlie, full_access) = \varnothing$, because no chain of morphisms through $UserRole$ and $RolePermission$ witnesses this pair. The inclusion $Q_{bad} \subseteq A fails$, so Theorem 4.1 guarantees that no natural transformation $\eta: Q_{bad} \Rightarrow (Attr_{input})_* \otimes (Attr_{res})_*$ exists.

5. Algorithm for Formal Verification of Query Correctness and Data Integrity

The algorithm combines the verification of referential integrity and the semantic correctness of queries (soundness) into a single static verification procedure. It is suitable for implementation in Catlab.jl.

Formalization of the problem. Let:

- S is a small category of the RBAC database schema,
- $I: S \to Par(Set)$ - instance functor (a specific database state),
- $Attr: S \to Set$ - an attribute functor (the set of columns in each table),
- $Q: S^{op} \times S_{res} \to Set$ - a profunctor modeling a query. The verification task consists of checking two properties:

    1. Referential integrity: for every morphism $f: A \to B$, there exists a function $I(f)$ that is a total function and whose image belongs to $PK(I(B))$.
    2. Query soundness: there exists a natural transformation (1)

The formal verification algorithm consists of three main stages.

Step 1. *Checking referential integrity*

For each external key $f: A \to B$:

1. verify totality - $dom(I(f)) = I(A)$;
2. check functionality - for each $a \in I(A)$, there is exactly one corresponding b;
3. check membership $I(f)(a) \in PK(I(B))$.

Step 2. *Constructing the query profunctor*

Construct Q as a composition of profunctors via span structures and WHERE filters. The complexity of $O(|UserRole| \times d)$ with indexes, where d is the maximum degree of the role.

Step 3. *Soundness check*

Compute the natural transformation η and verify the commutativity of the diagram $Q \Rightarrow Attr^{op}_{input} \otimes Attr_{res}$. If the diagram is commutative, the query is sound. Complexity: $O(|U| \times |P|)$.

The complete implementation of the verification algorithm in Julia v1.12 using Catlab.jl v0.16 is available in an open-source repository. The code includes: defining the RBAC scheme using @acset_type, constructing the instance functor $I: S \to Par(Set)$, computing the profunctor Q via the coend (function build_query_profunctor), independent construction of the attribute projection $Attr^{op}_{input} \otimes Attr_{res}$ (the build_attr_projection function), verification of the condition (1) (the

check_soundness function), demonstration of an incorrect query counterexample (build_unsound_query), and generation of SVG diagrams of naturality squares. The repository is available at: https://github.com/ai0master/CatLogDB.

6. Discussion of Results and Limitations

The proposed framework integrates profunctor semantics with regular logic in $Par(Set)$ into a unified formalism for joint verification of semantic soundness and referential integrity. The soundness criterion reduces to the existence of a natural transformation (1), while referential integrity is characterized by the isomorphism condition on $p_A : P \to I(A)$. Both criteria are structural and operate without executing queries on real data.

The Catlab.jl implementation confirms algorithmic correctness for the RBAC instance. The algorithm correctly identifies the authorized user for the DELETE query on FINANCE_DATA, identifies violating records via $I(A) \setminus \mathrm{Im}(pA)$, and verifies naturality of η. Unlike SMT-based tools (Cosette, VeriEQL), the η-criterion serves as a structural pre-filter that rejects unsound queries before invoking a solver, potentially reducing the solver's search space.

The current model has two principal limitations. First, predicates are restricted to conjunctions of attribute equalities; NOT, OR, aggregates, and outer joins are outside scope. Full 3VL support requires an extended categorical setting that explicitly tracks undefined values as a distinguished element, which is a planned direction for future work. Second, empirical evaluation is limited to a single small RBAC instance and does not provide scalability evidence. Systematic benchmarking against SMT-based tools and measurement of pre-filter effectiveness on larger schemas constitute planned future work.

7. Conclusions

The paper introduced a category-based verification scheme for relational databases in which query soundness and referential integrity are expressed within a common formal setting. By representing a database schema as a small category and interpreting database instances in Par(Set), the framework treats foreign keys as partial functions and makes NULL values or broken references explicit at the level of morphisms. This provides a structural basis for reasoning about database correctness before query execution.

For the considered fragment of conjunctive SQL queries, query semantics is represented by a Set-valued profunctor. The proposed soundness criterion reduces the verification problem to the existence of a natural transformation from the query profunctor to the corresponding attribute-projection profunctor. Referential integrity is handled through totality, functionality, and membership of foreign-key images in the relevant primary-key sets. Taken together, these criteria connect query structure, attribute projection, and schema-level constraints in a single categorical model.

The feasibility of the approach was demonstrated on an RBAC schema containing independent many-to-many relationships mediated by associative tables. The Catlab.jl prototype constructs the relevant schema and query profunctors, checks referential-integrity conditions, detects violating records, and verifies the naturality condition on a test instance. Although the experimental setting is intentionally small, it shows that the formal criteria can be translated into an executable verification procedure.

The proposed method differs from SMT-based query-equivalence tools such as Cosette and VeriEQL by first analysing the categorical structure of the schema and query, rather than treating equivalence primarily as a solver-level problem. In this role, the framework can serve as a static schema-auditing method and as a structural pre-filter before SMT-based verification. The main

limitations of the present study are the restriction to conjunctive queries with equality predicates and the lack of large-scale empirical evaluation. Future work should extend the supported SQL fragment, incorporate a fuller treatment of SQL three-valued logic, and evaluate the approach on larger schemas and real database instances.

Declaration on Generative AI

During the preparation of this work, the authors used AI-assisted tools exclusively for grammar and spelling checking and for the generation of illustrative figures. All conceptual development, mathematical modeling, formal proofs, and scientific interpretations were performed by the authors independently. The authors reviewed and edited all AI-assisted content and take full responsibility for the publication

References

[1] D.I. Spivak. Functorial data migration. *Information and Computation* 217 (2012), 31–51. DOI: 10.1016/j.ic.2012.05.001. URL: https://www.sciencedirect.com/science/article/pii/S0890540112001010.

[2] P. Schultz, R. Wisnesky. Algebraic data integration. *Journal of Functional Programming* 27 (2017), e24. DOI: 10.1017/S0956796817000168. URL: https://www.cambridge.org/core/journals/journal-of-functional-programming/article/algebraic-data-integration/AC36B1C39D4C76A18CB54F75B0D26796.

[3] P. Schultz, D.I. Spivak, C. Vasilakopoulou, R. Wisnesky. Algebraic databases. *Theory and Applications of Categories* 32 (2017), 547–619. URL: https://www.tac.mta.ca/tac/volumes/32/16/32-16.pdf.

[4] M. Johnson, R. Rosebrugh, R.J. Wood. Entity-Relationship-Attribute designs and sketches. *Theory and Applications of Categories* 10 (2002), 94–112. URL: https://www.tac.mta.ca/tac/volumes/10/3/10-03.pdf.

[5] B. Fong, D.I. Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press, Cambridge, 2019. URL: https://arxiv.org/abs/1803.05316.

[6] S. Awodey. *Category Theory*. 2nd ed. Oxford University Press, Oxford, 2010. URL: https://global.oup.com/academic/product/category-theory-9780199237180.

[7] B. Jacobs. *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics 141, Elsevier, Amsterdam, 1999. ISBN: 9780444501707. URL: https://www.sciencedirect.com/bookseries/studies-in-logic-and-the-foundations-of-mathematics/vol/141.

[8] D.I. Spivak. Database queries and constraints via lifting problems. *Mathematical Structures in Computer Science* 24(6) (2014), e240602. DOI: 10.1017/S0960129513000479. URL: https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/database-queries-and-constraints-via-lifting-problems/9B3D9C74EB12F65BAD045D53C8D4549F.

[9] A.K. Chandra, P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. *Journal of the ACM* 24(1) (1977), 156–178. DOI: 10.1145/800105.803397. URL: https://dl.acm.org/doi/10.1145/800105.803397.

[10] S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, MA, 1995. URL: https://wiki.epfl.ch/provenance2011/documents/foundations+of+databases-abiteboul-1995.pdf.

[11] S. Chu, C. Wang, K. Weitz, A. Cheung. Cosette: An Automated Prover for SQL. In: *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR 2017)*, Chaminade,

California, USA, 2017, pp. 1–7. URL: https://www.cidrdb.org/cidr2017/papers/p51-chu-cidr17.pdf.

[12] S. Chu, K. Weitz, A. Cheung, D. Suciu. HoTTSQL: Proving Query Rewrites with Univalent SQL Semantics. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2017), Barcelona, Spain, 2017, pp. 510–524. https://doi.org/10.1145/3062341.3062348.

[13] Y. He, P. Zhao, X. Wang, Y. Wang. VeriEQL: Bounded Equivalence Verification for Complex SQL Queries with Integrity Constraints. *Proceedings of the ACM on Programming Languages* 8(OOPSLA1) (2024), Article 132, 132:1–132:29. DOI: 10.1145/3649849. URL: https://dl.acm.org/doi/10.1145/3649849.

[14] Y. Wang, I. Dillig, S.K. Lahiri, W.R. Cook. Verifying Equivalence of Database-Driven Applications. *Proceedings of the ACM on Programming Languages* 2(POPL) (2018), Article 56, 56:1–56:29. DOI: 10.1145/3158144. URL: https://www.cs.sfu.ca/~yuepeng/pubs/popl18.pdf.

[15] A. Al-Brashdi, M. Butler, A. Rezazadeh. Incremental Database Design using UML-B and Event-B. In: *Proceedings of the 2nd Workshop on Formal and Model-Driven Techniques for Developing Trustworthy Systems*, EPTCS 271, 2018, pp. 34–47. DOI: 10.4204/EPTCS.271.3. URL: https://eprints.soton.ac.uk/415795/.

[16] J. Bénabou. Introduction to bicategories. In: *Reports of the Midwest Category Seminar I*. Lecture Notes in Mathematics 47, Springer, Berlin, Heidelberg, 1967, pp. 1–77. DOI: 10.1007/BFb0074299. URL: https://link.springer.com/chapter/10.1007/BFb0074299.