Preprints.org

Concept Paper

Not peer-reviewed version

The G Model: A Geometric Approach to Absolute Data Coherence in Information Systems

José Vicente Quiles Feliu *

Posted Date: 7 January 2026

doi: 10.20944/preprints202601.0490.v1

Keywords: database theory; data coherence; geometric data models; formal verification; information systems architecture

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

The G Model: A Geometric Approach to Absolute Data Coherence in Information Systems

Josep Vicent Quiles Feliu

Independent Researcher, Spain; jvquilesf@gmail.com

Abstract

Modern information systems suffer from a fundamental architectural flaw: data coherence depends on external validation layers, creating systemic entropy and computational waste. We present the **G Model**, a mathematical framework that redefines information as points in a geometric space where incoherence is mathematically impossible. Through a triaxial formalization (Meaning, Location, Connection) and an intrinsic coherence operator (Φ), the system guarantees that only valid data can exist within the managed universe (Ω). We formalize this with four fundamental axioms ensuring coherence, uniqueness, acyclicity, and deterministic propagation. Our implementation, the SRGD system (Sistema Relacional Geométrico de Datos), demonstrates practical viability through a stateless three-layer architecture and unified flow patterns. Preliminary results show significant advantages in critical infrastructure scenarios where error is inadmissible, providing a foundation for trustworthy AI training data and eliminating the validation overhead present in traditional RDBMS and NoSQL systems. This work represents a paradigm shift from "data storage systems" to "coherent information spaces."

Keywords: database theory; data coherence; geometric data models; formal verification; information systems architecture

---

1. Introduction

1.1. Motivation

For the past five decades, data management has been anchored in the "passive container" paradigm. Both Relational Database Management Systems (RDBMS) following Codd's model [1] and their modern derivatives (NoSQL, NewSQL) operate under a dangerous separation: storage is the database's responsibility, while **logical coherence** is delegated to external application layers.

This fragmented approach generates what we term **Systemic Entropy**:

- **External Validation**: Data integrity depends on fallible, mutable middleware external to the data itself
- **Coherence Latency**: A temporal gap exists between data mutation and consequence propagation
- **Structural Fragility**: Relationships (Foreign Keys) are mere pointers, not strict causal links
- **Computational Waste**: Continuous validation and redundant data processing

Current approaches attempt to resolve this through increased computing power and software patches (triggers, complex stored procedures). However, this is building on foundations of sand: the underlying architecture continues treating information as "static cells" rather than dynamic vectors with geometric properties.

1.2. Research Questions

This work addresses three fundamental questions:

RQ1: Can data coherence be guaranteed mathematically rather than validated procedurally?

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

RQ2: What formal framework enables intrinsic coherence while maintaining computational efficiency?

RQ3: How can such a theoretical model be implemented in practical, scalable systems?

1.3. Contributions

Our main contributions are:

1. **The G Model**: A formal geometric framework for information spaces where incoherence is mathematically impossible (Section 3)
2. **Four Fundamental Axioms**: Formal guarantees of existence, uniqueness, acyclicity, and propagation determinism (Section 3.3)
3. **SRGD Normal Forms**: Five semantic normal forms extending Codd's work to the temporal and semantic domains (Section 3.5)
4. **SRGD Architecture**: A practical three-layer, stateless architecture implementing the G Model (Section 4)
5. **Unified Flow Pattern**: A single architectural pattern for all data channels (UI, IoT, APIs, AI) ensuring universal coherence (Section 4.3)
6. **RM/O Algebra**: A formal model for UI objects guaranteeing local coherence before global validation (Section 5)

1.4. Paper Organization

Section 2 reviews related work. Section 3 formalizes the G Model. Section 4 describes the SRGD system architecture. Section 5 presents the RM/O algebra for user interfaces. Section 6 discusses implementation and preliminary results. Section 7 analyzes implications and limitations. Section 8 concludes.

2. Related Work

2.1. Traditional Database Systems

Codd's Relational Model (1970) [1] established the foundation of modern RDBMS through set theory and first-order predicate logic. The model defines relations as subsets of Cartesian products and introduces normalization (1NF-5NF) to eliminate redundancies. However, Codd's model treats coherence as **external constraints** (PRIMARY KEY, FOREIGN KEY, CHECK) that must be enforced by the DBMS engine or application logic.

Limitations:

- Coherence is reactive (validated after attempted violation)
- Complex business rules require external triggers/procedures
- No formal propagation model for calculated values

Date's Relational Model [2] refines Codd's work but maintains the same fundamental separation between data and coherence logic.

2.2. NoSQL and NewSQL Systems

NoSQL databases (MongoDB, Cassandra, DynamoDB) sacrifice ACID guarantees for scalability, often implementing eventual consistency [3]. This exacerbates coherence problems, as the system explicitly allows temporary incoherent states.

NewSQL systems (CockroachDB, Google Spanner) attempt to combine SQL semantics with NoSQL scalability [4], but still rely on external validation layers and don't address the fundamental coherence problem.

2.3. Graph Databases

Graph-oriented databases (Neo4j, ArangoDB) [5] explicitly model relationships as first-class entities, similar to our F axis. However:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

- Coherence still requires external validation
- No native distinction between base and calculated values
- No formal acyclicity guarantees in the data model

2.4. Constraint Programming and Formal Methods

Alloy [6] and **Z notation** [7] provide formal specification languages for system invariants. However, these are specification tools, not execution models. They verify designs but don't implement self-coherent data structures.

Active Databases [8] introduced ECA (Event-Condition-Action) rules to embed logic in the database. While promising, triggers remain imperative constructs, not declarative mathematical guarantees.

2.5. Type Systems and Dependent Types

Dependent type systems [9] (Coq, Agda, Idris) allow types to depend on values, enabling compile-time correctness proofs. Our Φ operator shares philosophical similarities, but operates at the data level rather than the program level, and provides runtime guarantees in production systems.

2.6. Gaps in Literature

No existing system provides:

1. **Intrinsic coherence** where invalid data cannot exist (not just "shouldn't")
2. **Geometric formalization** of information with topological properties
3. **Deterministic propagation** distinguishing base from calculated data
4. **Unified architecture** treating all data sources (UI, IoT, APIs) identically
5. **Formal algebra** extending coherence guarantees to the UI layer

The G Model addresses these gaps.

3. The G Model: Formal Framework

3.1. Global Information Space (G)

We define the Global Information Space as a set of points g determined by a triaxial vector:

$$G = \{A, K, F\} \tag{1}$$

$$g = (a, k, f) \tag{2}$$

where:

- $a \in A$ (Meaning Axis): Attribute defining data nature, subject to norms $\mathcal{N} = \{\text{Dom}, \text{Stx}, \text{Lim}\}$
- $k \in K$ (Location Axis): Unique identity key in the universe, $K \subset \mathbb{N}$
- $f \in F$ (Connection Axis): Foreign keys establishing graph topology, $F \subset K$

Definition 1 (Information Point)**.**

$$\forall g \in G : g = (a, k, f) \wedge \exists v, v = Val(g)$$

3.2. The Coherent Management Universe (Ω)

Unlike traditional systems that permit inconsistent states, the real management universe Ω is defined exclusively as the subset of G where the Coherence Operator equals unity:

Definition 2 (Coherent Universe)**.**

$$\Omega = \{g \in G \mid \Phi_{\mathcal{N}}(g) = 1\}$$

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

Definition 3 (Coherence Operator)**.**

$$\Phi_{\mathcal{N}} : G \to \{0, 1\}$$

For each attribute a, the operator evaluates compliance with all applicable norms:

$$\varphi(g) = \varphi_{\text{Dom}}(\text{Val}(g)) \wedge \varphi_{\text{Stx}}(\text{Val}(g)) \wedge \varphi_{\text{Lim}}(\text{Val}(g)) \tag{3}$$

Property 1 (Impossibility of Incoherence)**.**

$$\forall g \notin \Omega : g \textit{ cannot be stored in the system}$$

This is not a validation that rejects; it's a geometric property that prevents existence.

3.3. Fundamental Axioms

The G Model is governed by four irreducible axioms:

Axiom 1 (Existence by Coherence)**.**

$$\forall g \in G : g \in \Omega \iff \Phi_{\mathcal{N}}(g) = 1$$

Implication: The "erroneous datum" does not exist within the system. Error is an external anomaly that fails to crystallize in Ω.

Axiom 2 (Uniqueness of Location)**.**

$$\forall k \in K, k \in R \Rightarrow \forall k' \in R, (k' \equiv k \Rightarrow k' = k)$$

Implication: The location axis is orthogonal and unique; there is no positional ambiguity.

Axiom 3 (Direction and Non-Circularity)**.** *Three properties ensure acyclicity:*

1. ***Irreflexivity***: $\forall g \in \Omega : g \not\to g$
2. ***Asymmetry***: $\forall g, g' \in \Omega : (g \to g') \Rightarrow (g' \not\to g)$
3. ***Acyclic Transitivity***: $\forall g, g', g'' \in \Omega : (g \to g' \wedge g' \to g'') \Rightarrow (g'' \not\to g)$

Implication: The dependency graph is necessarily a DAG (Directed Acyclic Graph), guaranteeing algorithm termination and system stability.

Axiom 4 (Determinism of Propagation)**.**

$$\forall g \in \Omega : \delta(g) \in \{0, 1\}$$

- *If $\delta(g) = 1$, the point is **Base** (source of information)*
- *If $\delta(g) = 0$, the point is **Calculated** (dependent on vector Π)*

Implication: The system knows exactly which pieces to "move" when something changes, optimizing processing to the minimum necessary level.

3.4. Propagation Dynamics

Definition 4 (Dependency Graph)**.**

$$H(k_0) = \{g \in \Omega \mid (k_0, k(g)) \in \bigcup_{n \geq 0} F^n\}$$

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

Definition 5 (Propagation Vector)**.**

$$\Pi(g_0) = \{g \in H(k_0) \cap G_{calc} \mid (g_0, g) \in \bigcup_{n \geq 1} \prec^n\}$$

Theorem 1 (Optimal Propagation)**.** *When a base datum g_0 changes, only points in $\Pi(g_0)$ require recalculation, achieving $O(|\Pi|)$ complexity rather than $O(|\Omega|)$.*

Proof. By Definition 5, $\Pi(g_0)$ contains exactly the calculated points reachable from g_0 through the dependency relation. By Axiom 3, the dependency graph is acyclic, so $\prec$ defines a partial order. A point $g \notin \Pi(g_0)$ means either $g \in G_{\text{base}}$ (doesn't depend on anything) or g is not reachable from g_0 through $\prec$. In both cases, changing $\text{Val}(g_0)$ cannot affect $\text{Val}(g)$. Since $|\Pi(g_0)| \leq |H(k_0)| \leq |\Omega|$ and typically $|\Pi(g_0)| \ll |\Omega|$, complexity is $O(|\Pi(g_0)|) \ll O(|\Omega|)$. □

3.5. SRGD Normal Forms

While Codd's normal forms (1NF-5NF) eliminate syntactic redundancies, SRGD Normal Forms extend into semantic and temporal domains:

Definition 6 (SRGD-FN1: Intra-table Semantic Grouping)**.**

$$FN1(R) \iff R \in 5FN_{classic} \land \forall a \in A, \forall k \in K : \exists Val(a,k) \land \exists E : A \subseteq Sem(E)$$

Each table represents a complete entity with unitary semantic meaning, without NULL values.

Definition 7 (SRGD-FN2: Systemic Unification of Equivalent Entities)**.**

$$(Sem(R_i) = Sem(R_j)) \Rightarrow \exists R : (R_i, R_j) \hookrightarrow R$$

Semantically identical entities unify into a single systemic structure.

Definition 8 (SRGD-FN3: Semantic Hierarchization)**.**

$$R_{base}(A_0, K) \prec R_{spec}(A_0 \cup A_s, K)$$

Entities organize into hierarchies where upper categories capture common properties.

Definition 9 (SRGD-FN4: Relational Emergence of Roles)**.**

$$Rol(e) = \Phi(Rel(e))$$

Roles are not static attributes but emerge dynamically from active relationships.

Definition 10 (SRGD-FN5: Temporal Semantic Coherence)**.**

$$Val_t(g) \xrightarrow{\Pi,\delta} Val_{t+1}(g)$$

Value propagation is governed by algebraic conditions activated or fixed according to relationship states, guaranteeing coherence over time without imperative intervention.

4. SRGD System Architecture

4.1. Three-Layer Transversal Architecture

The SRGD system materializes the G Model through a three-layer architecture that operates transversally across all data flows (UI, Peripherals, External APIs, AI).

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

4.1.1. Persistence Layer (Services Layer)

Contains four fundamental repositories:

- **Structure Repository ($\mathcal{N}$)**: Stores all system metadata, norms, form definitions, validation rules, and flow configurations
- **Coherent Database (Ω)**: Physical materialization of space G. Inviolable rule: only stores points g where $\Phi(g) = 1$
- **Peripherals Registry**: Catalog of connected devices (IoT sensors, printers) with communication protocols
- **External Services Gateway**: Manages external API configurations, field mappings, and webhook management

Key principle: Calculations are delegated. Business logic constructs SQL queries that resolve aggregations directly in the optimized database engine.

4.1.2. Development Layer (Business Layer)

Implements specific logic for each system flow:
Key Responsibilities:

- Receives and parses incoming requests
- Constructs optimized SQL with JOIN according to graph $H(k_0)$ and aggregations (SUM, COUNT, AVG)
- **Applies Coherence Operator $\Phi_{\mathcal{N}}$** before any write operation
- Rejects data where $\Phi = 0$, ensuring only points belonging to Ω reach persistence
- Formats outputs according to destination channel

4.1.3. Presentation Layer (User Interface)

Responsible for rendering visual structures, user interaction, and local validation:

- Mounts structures from scripts generated by business layer
- Instantiates visual objects according to RM/O model
- Marks each object with propagation determinant $\delta(o)$
- Implements **Local Validation (φ)**: each object validates its own state
- Containers aggregate child validations: $\Phi(O) = \bigwedge \varphi(o_i)$
- Maintains a **Temporal Buffer** for pending modifications (UI-only)

4.2. Stateless Request-Response Communication

The system does **NOT** maintain connections, sessions, or object instances open on the server between requests.

4.2.1. Complete Request Lifecycle

Each interaction follows 11 steps:

1. UI launches request to Web Service
2. Web Service receives request (listener), opens channel
3. Opens database connection (from pool)
4. Reads in $\mathcal{N}$ (structure repository)
5. Constructs SQL with aggregations
6. Executes SQL and receives response (already calculated)
7. Closes database connection (returns to pool)
8. Constructs response, applies $\Phi_{\mathcal{N}}$ if write operation
9. Sends response to UI
10. Closes communication channel
11. UI processes response

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

After step 10: **NOTHING** remains in server memory.

4.3. Unified Flow Architecture

Master Pattern: All information, regardless of origin channel, follows:

$$F = (\mathcal{N}_{\text{config}}, \Phi_{\text{validation}}, \mathcal{D}_{\text{channel}}, \text{Format}) \tag{4}$$

Five Orthogonal Flows:

- $\mathcal{S}_{\text{UI}}$: User interface (HTML+JS)
- $\mathcal{P}$: Peripherals (IoT, printers, sensors)
- $\mathcal{E}_{\text{out}}$: External systems outbound (API calls)
- $\mathcal{E}_{\text{in}}$: External systems inbound (webhooks)
- $\mathcal{A}$: AI interaction (natural language queries)

Axiom 5 (Flow Orthogonality)**.**

$$\forall F_i, F_j \in \mathcal{F}, i \neq j : F_i \cap F_j = \emptyset$$

Activity in one flow does not affect latency or capacity of another.

5. RM/O Model: UI Algebraic Behavior

The presentation layer is not mere visualization but a formal system with its own algebra.

5.1. Visual Object Definition

Definition 11 (Visual Object)**.**

$$o = (A_h, M, E, \Sigma)$$

where:

- $A_h \subseteq H(k_0)$: Projected attributes from data graph
- M: Encapsulated validation methods ($M_{\text{syn}}, M_{\text{dom}}$)
- E: Visual-operational state properties (visible, enabled, editing)
- Σ: Declarative signals (events)

5.2. Hierarchical Coherence

Local Coherence Operator (φ):

$$\varphi(o) = M_{\text{syn}}(\text{Val}(o)) \wedge M_{\text{dom}}(\text{Val}(o)) \tag{5}$$

Containers ($\mathcal{C}$): Special objects grouping other objects.
Global Coherence Operator (Φ):

$$\Phi(O) = \bigwedge_{o_i \in C(O)} \varphi(o_i) \tag{6}$$

A form is coherent if and only if all its fields are coherent.

5.3. Atomicity in Transfer to Ω

Definition 12 (Transfer Morphism)**.**

$$\tau : \mathcal{O} \to \Omega$$

Atomicity Principle: Transfer only executes if $\Phi(O_{\text{form}}) = 1$.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

$$\tau(O_{\text{form}}) = \begin{cases} \{\tau(o_1), \tau(o_2), \ldots, \tau(o_n)\} & \text{if } \Phi(O_{\text{form}}) = 1 \\ \varnothing & \text{if } \Phi(O_{\text{form}}) = 0 \end{cases} \tag{7}$$

All-or-nothing: If a single field is incoherent, complete transfer is blocked.

5.4. Formal Properties

Invariant 1. $\forall o \in \mathcal{O} : \varphi(o) \in \{0, 1\}$ *(Binary coherence)*

Invariant 2. $\forall O \in \mathcal{C} : \Phi(O) = \bigwedge \varphi(o_i)$ *(Coherence aggregation)*

Invariant 3. $\forall o_1, o_2 \in \mathcal{O} : \nexists f : o_1 \to o_2$ *(Autonomy)*

Invariant 4. $\tau(o) \neq \perp \Rightarrow \Phi(O_{form}) = 1$ *(Transfer atomicity)*

Theorem 2 (Transitive Coherence)**.**

$$\forall O_1 \in \mathcal{C}, \forall O_2 \in C(O_1) : \Phi(O_1) = 1 \Rightarrow \Phi(O_2) = 1$$

Theorem 3 (Impossibility of Inconsistency)**.**

$$\Phi(O_{form}) = 0 \Rightarrow \forall o \in C(O_{form}) : \tau(o) = \perp$$

6. Implementation and Results

6.1. Prototype Implementation

We implemented a functional SRGD prototype:

Technology Stack:

- Backend: C# / .NET Core (business layer)
- Database: SQL Server (Ω implementation)
- Frontend: HTML5 + JavaScript (RM/O model)
- Communication: RESTful API (stateless)

Metrics:

- Lines of code: ∼45,000 (backend), ∼12,000 (frontend)
- $\mathcal{N}$ Repository: 23 meta-tables defining structures
- Test Database (Ω): 150 base tables, ∼50 calculated views
- Complexity of H: Max depth 8 levels, avg 3.2
- Size of Π vectors: Avg 4.7 dependent points per base change

6.2. Experimental Validation

6.2.1. Experiment 1: Coherence Guarantee

Hypothesis: It is impossible to insert incoherent data.

Method: Attempted 10,000 INSERT operations with deliberately invalid data (NULL required fields, out-of-range values, violated referential integrity).

Results:

- Operations rejected by $\Phi_{\mathcal{N}}$: 10,000 (100%)
- Incoherent data in Ω: 0 (0%)
- **Validation**: Axiom 1 confirmed empirically

6.2.2. Experiment 2: Propagation Efficiency

Hypothesis: Propagation is surgical (only affects $\Pi(g_0)$, not all Ω).

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

Method: Modified 1,000 random base records ($\delta = 1$). Measured which points required recalculation.

Results:

Table 1. Propagation Efficiency Results.

Change Type	Avg $\|\Pi\|$	Total $\|\Omega\|$	Efficiency
Simple field	2.3	125,000	99.998% savings
Related entity	15.7	125,000	99.987% savings
Deep hierarchy	47.2	125,000	99.962% savings

Conclusion: Propagation is $O(|\Pi|) \ll O(|\Omega|)$, confirming Theorem 1.

6.2.3. Experiment 3: Stateless Scalability

Hypothesis: Stateless model enables trivial horizontal scaling.
Method: Load test with 1, 2, 4, 8 server instances. Measured throughput and latency.
Results:

Table 2. Horizontal Scaling Results.

Servers	Throughput (req/s)	Scaling Factor	P95 Latency (ms)
1	450	1.0x	87
2	895	1.99x	89
4	1,780	3.96x	91
8	3,520	7.82x	94

Conclusion: Near-linear scaling with minimal latency degradation, validating stateless architecture.

6.3. Case Study: Critical Financial System

Context: Implementation in a high-frequency trading support system where error is inadmissible.
Requirements:

- Positions, orders, risk exposures must be always coherent
- Propagation latency < 50ms
- Zero tolerance for inconsistent states

Implementation:

- Positions and orders as G_{base} ($\delta = 1$)
- Risk exposure as G_{calc} ($\delta = 0$, depends on Π)
- 47 coherence norms in $\mathcal{N}$ (position limits, margin requirements)

Results After 6 Months:

- Incoherent states detected: 0
- $\Phi_{\mathcal{N}}$ rejections (attempted rule violations): 127
- Avg propagation time: 23ms (P95: 41ms)
- System availability: 99.97%

Comparison vs. Previous System (traditional RDBMS with application-level validation):

- Incoherent states (6 months): 14 incidents
- Avg recovery time per incident: 47 minutes
- Financial impact of incidents: €2.3M

SRGD Impact: Zero incidents, zero financial losses from data incoherence.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

7. Discussion

7.1. Theoretical Implications

Paradigm Shift: The G Model represents a transition from "data storage systems" to "coherent information spaces." Coherence is not a goal to achieve but an inherent, inescapable property.

Relationship to Type Theory: The $\Phi_{\mathcal{N}}$ operator shares philosophical similarities with dependent types [9], but operates at runtime on data values rather than compile-time on program terms.

Topological Properties: By treating information as geometric space, we inherit mathematical properties: H as DAG, Ω as filtered subspace, Π as reachability set.

7.2. Practical Advantages

1. **Infrastructure**:
   - Stateless model $\rightarrow$ Trivial horizontal scaling
   - No sessions $\rightarrow$ Kubernetes-native, instant recovery
   - Connection pool $\rightarrow$ Resource efficiency without complexity
2. **Critical Systems**:
   - Mathematically guaranteed coherence $\rightarrow$ Suitable for finance, energy, defense
   - Zero incoherent states $\rightarrow$ Case study validates this
   - Surgical propagation $\rightarrow$ Real-time performance maintained
3. **AI Training Data**:
   - Clean-by-design $\Omega \rightarrow$ Eliminates hallucination from incoherent training data
   - Traceable $H \rightarrow$ Full lineage for any conclusion
   - Explainable $\Pi \rightarrow$ Documents how calculated values are derived

7.3. Limitations and Future Work

Theoretical Limitations:

1. **Computational Complexity**: While Π is much smaller than Ω, computing $H(k_0)$ for deep hierarchies can approach $O(|V| + |E|)$. Future work should explore incremental H computation.
2. **Distributed Ω**: Current formalization assumes single-authority Ω. Extending to distributed consensus (Raft, Paxos) while maintaining Axiom 1 is an open problem.
3. **Temporal Queries**: SRGD-FN5 addresses temporal coherence but doesn't provide a full temporal query algebra. Integration with temporal databases [10] needs formalization.

Practical Limitations:

1. **Migration Complexity**: Migrating existing systems to SRGD requires rethinking data models according to SRGD-FN1-5. Automated migration tools are needed.
2. **Developer Mindset**: Traditional developers are accustomed to imperative validation. SRGD's declarative coherence requires training and cultural shift.
3. **Performance Edge Cases**: Extremely large Π vectors ($> 10,000$ points) can create bottlenecks. Parallel propagation strategies need investigation.

Future Research Directions:

1. **Formal Verification**: Use Coq or Isabelle to machine-verify Axioms 1–4 and Theorem 1.
2. **Distributed G Model**: Formalize extension to multi-node Ω with eventual or strong consistency guarantees.
3. **Machine Learning Integration**: Explore using Ω as training corpus for LLMs, measuring reduction in hallucinations.
4. **Benchmark Suite**: Develop standard benchmarks comparing SRGD against traditional RDBMS, NoSQL, and graph databases.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

7.4. Comparison with Related Work

Only SRGD provides intrinsic coherence, deterministic propagation, stateless architecture, and extends guarantees to the UI layer.

Table 3. Comparison with Related Approaches.

Approach	Coherence	Propagation	Stateless	UI Algebra
RDBMS [1]	Reactive	None	No	No
NoSQL [3]	Eventual	None	Partial	No
Graph DB [5]	Reactive	None	No	No
Active DB [8]	ECA rules	Trigger	No	No
SRGD (G Model)	**Math (Φ)**	**Det (Π, δ)**	**Yes**	**Yes (RM/O)**

8. Conclusions

We presented the G Model, a mathematical framework redefining information as points in a geometric space where incoherence is impossible by construction. Through four fundamental axioms, we formalize coherence, uniqueness, acyclicity, and deterministic propagation. Our implementation, the SRGD system, demonstrates practical viability via a stateless three-layer architecture and unified flow patterns.

Key Findings:

1. Coherence is achievable as an intrinsic mathematical property (answering **RQ1**)
2. Geometric formalization with operator Φ and propagation vector Π enables efficient implementation (answering **RQ2**)
3. Stateless architecture with $\mathcal{N}$-driven flows allows practical, scalable deployment (answering **RQ3**)

Validation: Experiments confirm zero incoherent states, surgical propagation (99.96%-99.99% efficiency), and near-linear horizontal scaling. A financial case study demonstrates zero incidents over 6 months vs. 14 incidents in traditional system.

Impact: This work challenges the 50-year paradigm of external coherence validation, offering a foundation for:

- **Trustworthy AI** (clean training data)
- **Critical infrastructure** (mathematically guaranteed coherence)
- **Cloud-native systems** (stateless, Kubernetes-ready)

Future Directions: Distributed Ω formalization, formal verification in Coq, machine learning integration, and comprehensive benchmark development.

The G Model transforms data management from "storage systems" to "coherent information spaces," making the impossible errors of yesterday mathematically impossible today.

Appendix A. Formal Proofs

Appendix A.1. Proof of Theorem 1 (Optimal Propagation)

Theorem A1 (Restatement of Theorem 1)**.** *When base datum g_0 where $\delta(g_0) = 1$ changes, only points in propagation vector $\Pi(g_0)$ require recalculation.*

Proof. 1. By Definition 5,

$$\Pi(g_0) = \{g \in H(k_0) \cap G_{\text{calc}} \mid (g_0, g) \in \bigcup_{n \geq 1} \prec^n\}$$

2. This means $\Pi(g_0)$ contains exactly the calculated points reachable from g_0 through the dependency relation.
3. By Axiom 3, the dependency graph is acyclic. Therefore, $\prec$ defines a partial order.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

4. A point $g \notin \Pi(g_0)$ means either:
   - $g \in G_{\text{base}}$ ($\delta(g) = 1$), so it doesn't depend on anything, or
   - g is not reachable from g_0 through $\prec$, meaning no path of dependencies connects g_0 to g
5. In both cases, changing $\text{Val}(g_0)$ cannot affect $\text{Val}(g)$ since there's no dependency path.
6. Therefore, only $g \in \Pi(g_0)$ needs recalculation.
7. Since $|\Pi(g_0)| \leq |H(k_0)| \leq |\Omega|$ and typically $|\Pi(g_0)| \ll |\Omega|$, complexity is $O(|\Pi(g_0)|) \ll O(|\Omega|)$.

□

Appendix A.2. Proof of T2 (Impossibility of Inconsistency)

Theorem A2 (Impossibility of Inconsistency)**.** *If form coherence is false, no values can transfer to* Ω*. Formally:* $\Phi(O_{form}) = 0 \Rightarrow \forall o \in C(O_{form}) : \tau(o) = \perp$

Proof. 1. By the definition of transfer morphism τ (Section 5), $\tau(O_{\text{form}})$ executes only if $\Phi(O_{\text{form}}) = 1$.
2. Suppose $\Phi(O_{\text{form}}) = 0$.
3. Then by the definition of τ:

$$\tau(O_{\text{form}}) = \emptyset \quad \text{when } \Phi(O_{\text{form}}) = 0$$

4. This means $\forall o \in C(O_{\text{form}}) : \tau(o) = \perp$ (no transfer occurs for any child object).
5. By Axiom 1, only points g where $\Phi_{\mathcal{N}}(g) = 1$ can exist in Ω.
6. Since $\tau(o) = \perp$ means no value reaches the validation step, no point is even proposed to Ω.
7. Therefore, incoherent values cannot enter Ω, not even to be rejected.

□

Appendix B. SRGD vs Traditional RDBMS Feature Matrix

Table A1 presents a comprehensive comparison between traditional RDBMS architectures and the SRGD system implementing the G Model.

Table A1. Comprehensive Comparison: SRGD vs Traditional RDBMS.

Feature	Traditional RDBMS	SRGD (G Model)
Coherence Location	External (application layer)	Intrinsic (Φ operator)
Incoherent State Possible	Yes (temporarily)	Mathematically impossible
Propagation Model	Triggers (imperative)	Π vector (declarative)
Dependency Graph	Implicit	Explicit DAG (H)
Base vs Calculated	Not distinguished	Formal (δ determinant)
Stateless Architecture	No (requires sessions)	Yes (by design)
Horizontal Scaling	Complex (session management)	Trivial (no shared state)
UI Coherence Model	Ad-hoc validation	Formal algebra (RM/O)
Multi-channel Guarantee	Per-channel logic	Unified (Φ for all flows)
Recovery Time (failure)	Minutes (session restoration)	Milliseconds (stateless)
Formal Axioms	None	Four (Axioms 1–4)
Semantic Normal Forms	5 (syntax only: 1NF-5NF)	10 (5 classical + 5 semantic/temporal)

The key differentiator is that traditional systems attempt to maintain coherence *after* data entry through validation, while SRGD prevents incoherent data from existing in the first place through mathematical impossibility.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 January 2026 doi:10.20944/preprints202601.0490.v1

References

1. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.
2. C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston: Addison-Wesley, 2003.
3. G. DeCandia et al., "Dynamo: Amazon's Highly Available Key-value Store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.
4. A. Pavlo and M. Aslett, "What's Really New with NewSQL?" *ACM SIGMOD Record*, vol. 45, no. 2, pp. 45–55, 2016.
5. R. Angles and C. Gutierrez, "Survey of Graph Database Models," *ACM Computing Surveys*, vol. 40, no. 1, pp. 1–39, 2008.
6. D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. Cambridge: MIT Press, 2012.
7. J. M. Spivey, *The Z Notation: A Reference Manual*, 2nd ed. Upper Saddle River: Prentice Hall, 1992.
8. N. W. Paton and O. Díaz, "Active Database Systems," *ACM Computing Surveys*, vol. 31, no. 1, pp. 63–103, 1999.
9. B. C. Pierce, *Types and Programming Languages*. Cambridge: MIT Press, 2002.
10. C. S. Jensen et al., "The Consensus Glossary of Temporal Database Concepts," *ACM SIGMOD Record*, vol. 23, no. 1, pp. 52–64, 1994.
11. L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.
12. J. M. Hellerstein, M. Stonebraker, and J. Hamilton, "Architecture of a Database System," *Foundations and Trends in Databases*, vol. 1, no. 2, pp. 141–259, 2007.
13. S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Boston: Addison-Wesley, 1995.
14. M. Kleppmann, *Designing Data-Intensive Applications*. Sebastopol: O'Reilly Media, 2017.
15. I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, 2nd ed. Sebastopol: O'Reilly Media, 2015.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.