Heuristic-Driven Query Generation and Evaluation Framework: A Context-Aware Approach to Natural Language to SQL Translation

Research Team
Incerto Labs

October 30, 2025

Abstract

We present a novel framework for natural language to SQL translation that leverages heuristic stores, ontological structures, and generative data feedback loops to optimize query generation accuracy and user satisfaction. Our system employs a multi-tiered evaluation architecture that encompasses security, factual consistency, personalization, and response relevance. Through the implementation of a crawling mechanism that transforms generative data into actionable knowledge, we achieve significant improvements in table selection accuracy, extraction precision, and SQL execution success rates. The proposed architecture integrates a memory-augmented heuristic store with an adaptive context generation pipeline, enabling real-time personalization and multi-turn conversation coherence. We provide formal mathematical frameworks for table likelihood computation, query-context mapping, and evaluation metrics that directly correlate with business value propositions.

1 Introduction

The translation of natural language queries into structured query language (SQL) for data access remains a critical challenge in data democratization. **The problem statement is to maximize the accuracy in fetching the right data based on business leaders' queries—specifically for L1 and L2 query types**. Business leaders require direct access to data with extreme accuracy guarantees, and current systems fail to deliver the required precision.

1.1 The Accuracy Challenge in L1 and L2 Queries

L1 and L2 queries represent user-facing data requests that may involve significant complexity: multiple business metrics, diverse filters, complex join logic across related tables, temporal aggregations, and domain-specific calculations. The challenge is not the simplicity or complexity of the query itself, but rather the requirement for extremely high accuracy (95-100%) in query interpretation and execution.

When converting natural language queries to SQL for L1/L2 use cases, a Large Language Model (LLM) must make several critical decisions:

- **Correct Table and View Identification**: Identifying which tables, views, and data sources contain the requested business entities and metrics
- **Accurate Metric Extraction**: Selecting the exact business metrics, calculations, and aggregations the user is requesting

- **Filter and Constraint Application**: Applying the correct conditions, date ranges, and business logic constraints based on user intent
- **Join Logic and Relationships**: Correctly specifying join conditions across multiple tables and handling complex business relationships
- **Data Semantics and Format**: Understanding how data is structured, calculated, and what assumptions underlie the data

For L1/L2 queries, there is binary correctness: either the query returns the exact, correctly-calculated data the user requested or it fails to meet requirements. There is no tolerance for misinterpreted metrics, incorrect filters, or data inconsistency. This binary nature of query correctness creates an exceptionally high bar for accuracy.

1.2 Our Solution Approach: Ontology-Driven Learning from Failures

We solve this problem through a multi-layered strategy combining ontological structure with rigorous evaluation and continuous learning: (1) **Comprehensive evaluation** of all critical steps—table selection, metric extraction, filter validation, and result consistency—to identify failure points; (2) **Zero true negative principle** ensuring no valid patterns are lost during heuristic extraction; (3) **Generative data crawling** that systematically extracts signals from user interactions to update knowledge continuously; and (4) **Personalization** through per-user, per-team, and per-query-type statistics that capture user-specific patterns and preferences.

1.3 Framework Overview

The framework integrates: (1) a foundational ontology providing semantic structure; (2) a heuristic store accumulating successful patterns and user preferences; (3) multi-tiered evaluation assessing security, accuracy, and relevance; (4) a crawling mechanism transforming interactions into refined knowledge; and (5) an adaptive context generation pipeline tailored to each user. This architecture treats ontology as a foundation continuously refined through production feedback, enabling the high accuracy required for L1/L2 query execution.

2 Entity, Filter, and Metric Extraction with Ontology-Driven Definition

The foundation of our query generation process rests on the accurate extraction and linking of three critical components: entities, filters, and metrics. These components are extracted from natural language queries using a combination of ontological structure, pattern recognition, and user-guided definition when unknown entities are encountered.

2.1 Extraction Pipeline

The extraction pipeline operates as follows:

1. **Initial Ontology-Based Extraction**: The system leverages the foundational ontology $\mathcal{O} = (\mathcal{V}, \mathcal{R}, \mathcal{C}, \mathcal{I})$ to identify known entities, filters, and metrics from the user's natural language query. Using entity recognition and semantic parsing techniques, we extract candidate terms that match vocabulary entries $\mathcal{V}$ and relations $\mathcal{R}$ defined in the ontology.
2. **Unknown Entity Detection**: When entities or metrics are encountered that do not match any known vocabulary entries in the ontology, the system identifies these gaps and marks them for user definition. This is crucial for maintaining high accuracy—rather

than making assumptions about undefined terms, the system proactively asks the user to clarify.

3. **Interactive User Definition Prompt**: For each unidentified entity, filter, or metric, the system presents a user-friendly definition interface on the UI. The user is prompted to either:
   - Select the entity from a suggested list based on semantic similarity
   - Provide a definition by mapping to existing tables and columns
   - Specify custom calculation logic for novel metrics
4. **Ontology Update**: User-defined entities and metrics are incorporated into the ontology for future queries, creating a continuously evolving semantic knowledge base.
5. **Table and Query Linking via Heuristics**: Once all entities, filters, and metrics are identified or defined, the system uses two critical heuristic classes to establish connections:
   - H_t **(Table Heuristics)**: Statistical aggregates that capture table selection patterns based on entity-metric pairs, user preferences, and historical query success rates. Formally, $H_t = P(T_i \mid e, m, u)$, representing the conditional probability of selecting table T_i given entities e, metrics m, and user u.
   - H_q **(Query Heuristics)**: Patterns extracted from successful and rejected queries that capture which join logic, aggregation patterns, and filter combinations are most reliable for specific entity-metric-filter combinations. This includes successful SQL structures that match the extracted components.

2.2 Formal Extraction Process

Given a natural language query q, we formally define the extraction process as:

$$(e_q, f_q, m_q) = \text{Extract}_{\mathcal{O}}(q) \cup \text{UserDef}_{\text{UI}}(q_{\text{unknown}}) \tag{1}$$

where:

- e_q represents extracted entities
- f_q represents extracted filters
- m_q represents extracted metrics
- $\text{Extract}_{\mathcal{O}}(q)$ denotes ontology-based extraction using vocabulary and relations
- q_{unknown} denotes query components not found in the ontology
- $\text{UserDef}_{\text{UI}}(q_{\text{unknown}})$ denotes user-provided definitions via UI prompts

After extraction, table and query linking is performed:

$$\mathcal{T}_{\text{candidates}} = \text{rank}_{\text{by\_heuristics}}(\mathcal{T}, H_t(e_q, m_q, u)) \cap \text{feasibility}(H_q) \tag{2}$$

where:

- $\mathcal{T}$ is the set of all available tables
- H_t ranks tables based on statistical likelihood given extracted entities, metrics, and user identity
- H_q filters candidates based on query feasibility heuristics (join availability, column presence, aggregation support)
- $\mathcal{T}_{\text{candidates}}$ is the ranked set of candidate tables for query execution

2.3 Flow Diagram: Extraction and Linking Process

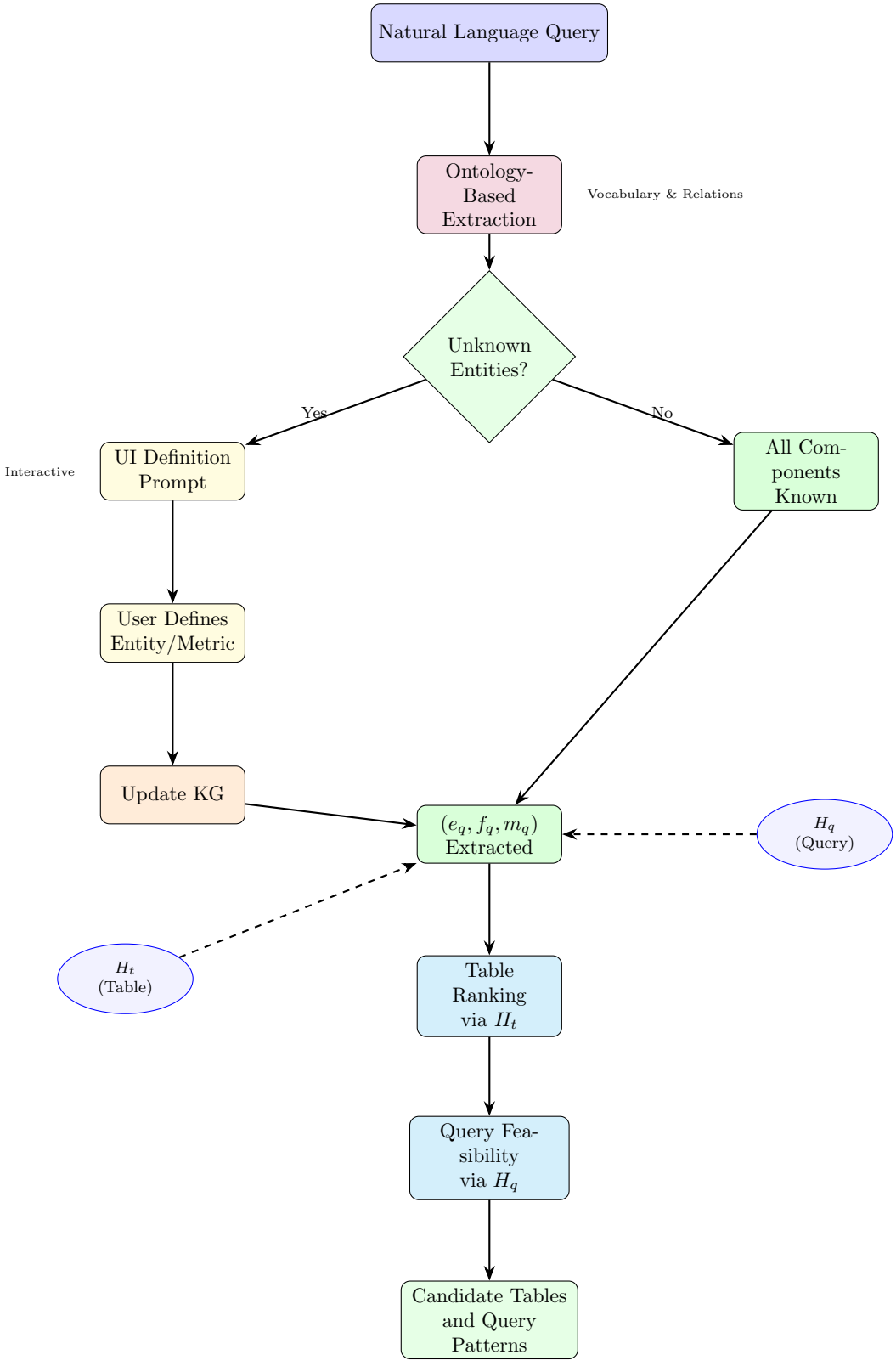

Figure 1: Entity, filter, and metric extraction pipeline with ontology-based extraction, interactive UI definition for unknown components, and heuristic-guided table ranking and query feasibility assessment.

3 System Architecture

3.1 Flow Diagram and Generation Loop

The system architecture follows a cyclical generation-evaluation-refinement paradigm, as illustrated in Figure 2.

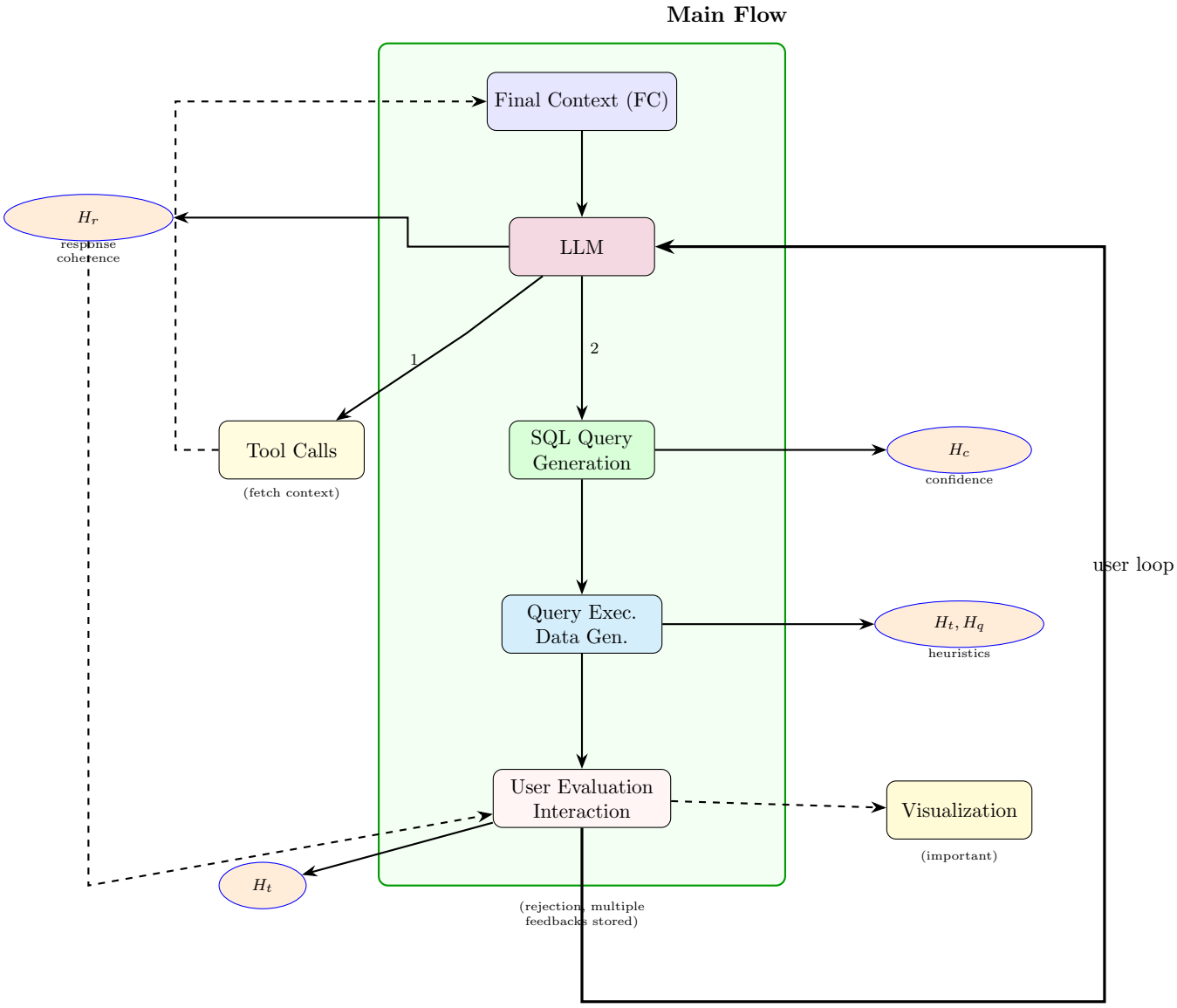

Figure 2: System flow diagram showing the generation-evaluation-refinement cycle with LLM processing, query execution, heuristic capture (H_t, H_q, H_c, H_r), and user feedback loops.

The core workflow consists of:

1. **Data Gathering Phase**: Proprietary client data is indexed and prepared for retrieval
2. **User Need Analysis**: Natural language queries are parsed to extract intent, entities, and constraints
3. **Visualization Planning**: The system determines appropriate response formats based on query type and user preferences

4. **Response Generation**: A sequence of heuristic functions $\{H_{F_1}, H_{F_2}, \ldots, H_{F_t}\}$ are applied iteratively
5. **Stopping Criterion**: Generation terminates when confidence thresholds are met or maximum iterations are reached
6. **Feedback Integration**: User actions (acceptance/rejection/modification) are stored in the heuristic store

3.2 Heuristic Store: The Memory Module

The heuristic store serves as a persistent memory module that accumulates operational intelligence over time. It maintains the following components:

- **Response Sequences**: Historical query-response pairs indexed by similarity
- **Table Frequency Statistics**: Empirical distributions $P(T_i \mid u, q)$ of table usage per user and query type
- **Confidence Metrics**: Probabilistic assessments of query correctness
- **Rejection Patterns**: Analysis of failed or rejected queries to identify systematic errors
- **Visualization Preferences**: User-specific rendering and formatting preferences
- **Flow Heuristics**: Temporal patterns in multi-turn conversations

4 Formal Definitions

We establish the following formal definitions that constitute the foundational components of our system:

4.1 Core Terminology

Definition 1 (Evaluations): Let $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ be a set of quantitative metrics that measure the operational efficacy of the system. Each evaluation $e_i : \mathcal{Q} \times \mathcal{R} \to \mathbb{R}$ maps a query-response pair to a real-valued performance indicator.

Definition 2 (Generative Data): Generative data $\mathcal{G}$ represents the corpus of system-generated artifacts including rejected queries, partial executions, user corrections, and interaction traces. Formally, $\mathcal{G} = \{(q, r, u, t, a) \mid q \in \mathcal{Q}, r \in \mathcal{R}, u \in \mathcal{U}, t \in \mathbb{T}, a \in \mathcal{A}\}$, where $\mathcal{Q}$ denotes queries, $\mathcal{R}$ denotes responses, $\mathcal{U}$ denotes users, $\mathbb{T}$ denotes timestamps, and $\mathcal{A}$ denotes user actions (acceptance/rejection).

Definition 3 (Heuristics): Heuristics $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$ are computed statistical aggregates derived from generative data. Each heuristic $H_i : \mathcal{G} \to \mathbb{R}^d$ extracts d-dimensional features such as table mention frequencies, entity co-occurrence patterns, and temporal query distributions.

Definition 4 (Contexts): A context $C \in \mathcal{C}$ represents structured information provided to the language model. Formally, $C = \langle K, D, H, P \rangle$, where K denotes knowledge, D denotes data samples, H denotes relevant heuristics, and P denotes personalization parameters.

Definition 5 (Knowledge): Knowledge $\mathcal{K}$ comprises factual information about data schemas, business operations, entity relationships, and metric computation rules. We represent knowledge as a directed acyclic graph $\mathcal{K} = (V, E)$ where V contains entity and relation nodes, and E contains semantic edges annotated with constraints.

Definition 6 (Ontology): The ontology $\mathcal{O} = (\mathcal{V}, \mathcal{R}, \mathcal{C}, \mathcal{I})$ defines the structural rules and semantic framework for knowledge representation, where $\mathcal{V}$ is the vocabulary, $\mathcal{R}$ is the set of

relations, $\mathcal{C}$ is the set of constraints, and $\mathcal{I}$ is the interpretation function that maps domain concepts to formal semantics.

Definition 7 (Crawling): Crawling $\psi : \mathcal{G} \times \mathcal{H} \to \mathcal{K}$ is the algorithmic process of transforming generative data into structured knowledge through heuristic-guided extraction, pattern recognition, and ontological alignment.

5 Algorithmic Framework

5.1 Core Algorithms

The system implements four primary algorithmic categories:

5.1.1 Heuristic Extraction

Algorithm 1: Extracting All Heuristics

Input: Generative data $\mathcal{G}$, Ontology $\mathcal{O}$
Output: Heuristic set $\mathcal{H}$

for each $(q, r, u, t, a) \in \mathcal{G}$ **do**
  Extract entities $\mathcal{E}_q$ from q using $\mathcal{O}$
  Extract tables $\mathcal{T}_r$ from r
  Update frequency distribution $P(T \mid u, \mathcal{E}_q)$
  Compute confidence score $\sigma(r) \leftarrow f(a, t, \text{execution\_success})$
end for
Aggregate statistics into $\mathcal{H} = \{H_1, \ldots, H_m\}$
return $\mathcal{H}$

5.1.2 Context Construction

The system dynamically constructs context vectors by composing knowledge, data samples, and heuristics tailored to each query.

Algorithm 2: Building Different Contexts from Heuristics

Input: Query q, User u, Heuristics $\mathcal{H}$, Knowledge base $\mathcal{K}$
Output: Context C

Retrieve relevant knowledge $K_q \leftarrow \text{retrieve}(\mathcal{K}, q, \mathcal{O})$
Extract filters $F = \{f, e, m\}$ from q using entity recognition
Compute personalization parameters $P_u \leftarrow \text{profile}(u, \mathcal{H})$
Select top-k tables using table likelihood (Equation 3)
Construct context $C \leftarrow \langle K_q, \text{samples(tables)}, \mathcal{H}_{\text{relevant}}, P_u \rangle$
return C

5.1.3 Retrieval, Indexing, and Reranking

The context creation pipeline employs sophisticated retrieval mechanisms:

- **Retrieval**: Dense vector embeddings identify semantically similar historical queries

- **Indexing**: Multi-modal indices (keyword, semantic, temporal) enable efficient search
- **Reranking**: Heuristic-based reranking prioritizes contextually relevant information

5.2 Crawling: Knowledge Extraction Without Explicit Heuristics

The crawling mechanism operates in a bootstrapping mode, extracting knowledge from generative data with zero to minimal initial heuristics (denoted as $H^0 \rightarrow H^{2+g}$, where g represents generations of refinement).

Algorithmic Guidelines for Crawling:

1. **Zero False Negative Constraint**: Extract heuristics with FNR $= 0$ to ensure no valid patterns are discarded
2. **Accuracy-Based Prioritization**: Maintain most accurate heuristics H_{top} at the top of the ranking for context construction
3. **Hallucination Prevention**: Prevent chaining of hallucinations across multiple conversation turns through consistency checks
4. **Content Pruning via Personalization**: Apply user-specific filters to reduce context size while maintaining relevance

5.2.1 The Crawling Process: From Generative Data to Knowledge Graphs

The crawling pipeline transforms raw generative data into structured knowledge graphs through a heuristic-guided LLM-based extraction process. Figure 3 illustrates this transformation pipeline.

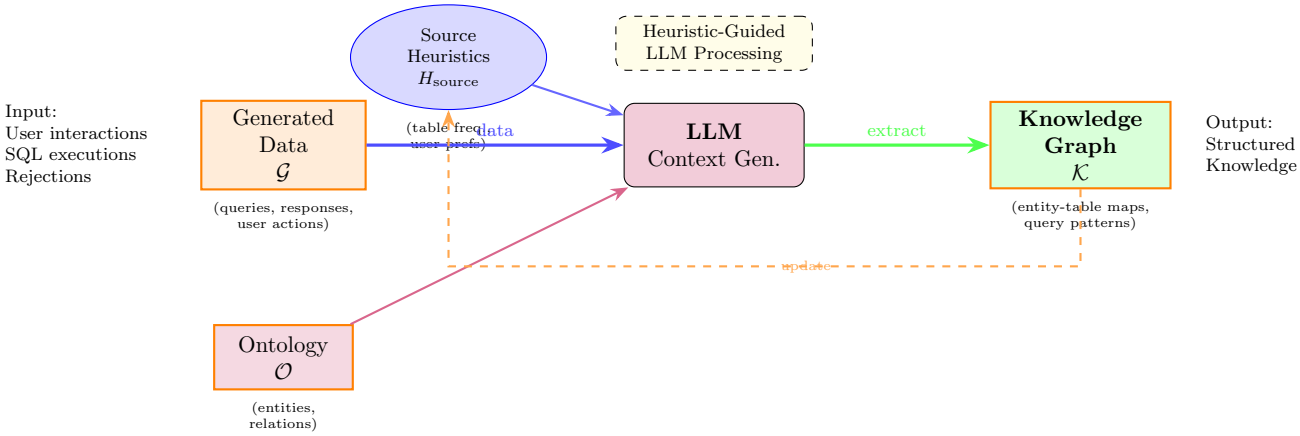

Figure 3: Crawling pipeline transforming generated data $\mathcal{G}$ into knowledge graph $\mathcal{K}$ using LLM-based extraction guided by heuristics H_{source} and ontology $\mathcal{O}$, with continuous feedback loop.

The Role of Heuristics in Crawling:

Heuristics serve critical functions: (1) source contextualization providing statistical signals about successful patterns; (2) disambiguation guiding interpretation based on user history; (3) quality filtering identifying high-confidence patterns; and (4) extraction prioritization determining which data warrants knowledge graph inclusion.

Why Heuristics Are Effective:

Since LLMs sample from probability distributions, heuristics influence generation by: (1) biasing toward historically successful patterns; (2) reducing output variance; and (3) compensating for limited observability—many LLMs (notably Claude) don't expose log probabilities,

so heuristics provide external confidence signals. This creates a statistical framework that complements the LLM's generative capabilities.

6 Mathematical Formulation

6.1 Table Likelihood Computation

The probability of selecting tables for a given query is computed using a composite likelihood function:

$$TL(C(f, e, m), q, u) = \prod_i \Pr(T_i \mid C_{i,1}, C_{i,2}, \ldots, C_{i,m_f}) \tag{3}$$

where:

- $C(f, e, m)$ represents context components: filters (f), entities (e), and metrics (m)
- q is the natural language query
- u is the user identifier
- T_i denotes candidate tables
- $C_{i,j}$ are contextual features for table T_i
- m_f is the dimensionality of the feature space

This formulation integrates table-specific heuristics with query patterns and user preferences to produce a ranked list of candidate tables.

6.2 Query-Context Mapping to Final Confidence

The final confidence of a generated query is computed through:

$$QC(TL, q', H_g) = FC \tag{4}$$

where:

- TL is the table likelihood from Equation 3
- q' is the generated SQL query
- H_g is the global heuristic context including H_7 (confidence codes) and H_8 (execution history)
- FC is the final confidence score

The final confidence FC is produced through a query generation module that employs a Large Language Model (LLM) with the composed context:

$$FC = \text{LLM}_{\text{QG}}(C, TL, H_7, H_8) \rightarrow (\text{SQL}, \sigma) \tag{5}$$

where LLM_{QG} denotes the query generation module, and σ is the confidence probability.

7 Evaluation Framework

7.1 Multi-Dimensional Evaluation Metrics

We employ a comprehensive evaluation framework that assesses system performance across multiple dimensions:

7.1.1 (A) Security and Safety

Evaluate SQL injection vulnerabilities, data access violations, and malicious query patterns:

$$\mathcal{E}_{\text{security}} = \mathbb{I}[\text{no\_injection}(q')] \cdot \mathbb{I}[\text{authorized\_access}(u, T)] \tag{6}$$

7.1.2 (D) Accuracy of SQL Execution

Measure the percentage of generated queries that execute successfully:

$$\mathcal{E}_{\text{exec}} = \frac{|\{q' \in \mathcal{Q}' : \text{executes}(q')\}|}{|\mathcal{Q}'|} \times 100\% \tag{7}$$

7.1.3 Extraction Accuracy

Assess the precision of entity extraction across filters (F), entities (E), and metrics (M):

$$\mathcal{E}_{\text{extract}} = \frac{1}{3}\left(\text{Precision}_F + \text{Precision}_E + \text{Precision}_M\right) \tag{8}$$

7.1.4 Table Selection Accuracy

Evaluate whether the correct tables are selected for query execution:

$$\mathcal{E}_{\text{table}} = \frac{|T_{\text{selected}} \cap T_{\text{ground\_truth}}|}{|T_{\text{ground\_truth}}|} \tag{9}$$

7.1.5 Response Relevance

Measure semantic similarity between generated responses and expected outputs using embedding-based metrics.

7.1.6 User Satisfaction

Collect explicit feedback from users and integrate with engineering workflows:

$$\mathcal{E}_{\text{satisfaction}} = \sum_{u \in \mathcal{U}} w_u \cdot \text{rating}_u \tag{10}$$

where w_u represents user-specific weights based on their interaction frequency and domain expertise.

7.1.7 Personalization

Quantify the degree to which responses incorporate user-specific preferences:

$$\mathcal{E}_{\text{personalization}} = P(F, E, M \mid u) \cdot \mathbb{I}[\text{query\_uses}(F, E, M)] \tag{11}$$

7.1.8 SQL Confidence

Compute similarity with previously vetted SQL queries:

$$\mathcal{E}_{\text{confidence}} = \max_{q_v \in \mathcal{Q}_{\text{vetted}}} \text{sim}(q', q_v) \tag{12}$$

This serves as either a Bayesian prior assumption or a retrospective evaluation based on historical data.

7.1.9 Factual Consistency

Verify that the generated response maintains factual consistency with the underlying data and does not introduce hallucinations:

$$\mathcal{E}_{\text{consistency}} = \mathbb{I}[\text{results}(q') \subseteq \text{possible\_results}(\mathcal{D})] \tag{13}$$

7.2 Business Metrics Alignment

The evaluation framework is explicitly designed to map to business value propositions:

(D) Query Volume Reduction:

$$\text{BM}_1 = \frac{\#\text{ queries to data teams}}{\#\text{ total users}} \tag{14}$$

A successful system reduces this ratio by empowering users to self-serve data requests.

(A) Weighted Satisfaction Score:

$$\text{BM}_2 = \sum_{u \in \mathcal{U}} w_u \cdot s_u \tag{15}$$

where w_u weights users by their business impact and s_u is their satisfaction rating.

High-Value Conversation Metrics:

We define high-value conversations by three criteria:

- **Length**: Conversations exceeding τ_{length} exchanges
- **Multiple Turns**: Conversations with ≥ 3 user-system exchanges
- **Business Metrics Usage**: Conversations that reference crucial business KPIs

$$\text{BM}_3 = |\{c \in \mathcal{C} : \text{length}(c) > \tau_{\text{length}} \wedge \text{turns}(c) \geq 3 \wedge \text{uses\_KPI}(c)\}| \tag{16}$$

7.3 Per-User Granular Analysis

The system maintains per-user statistics for accuracy, coverage, and query patterns:

$$\text{Metrics}_u = \{\text{accuracy}_u, \text{coverage}_u, \text{query\_frequency}_u, \text{preferred\_tables}_u\} \tag{17}$$

This enables:

1. Identification of jitter patterns (inconsistent entity extraction)
2. Tracking of entity retrieval failures
3. Personalized model fine-tuning

8 Results and Discussion

8.1 Experimental Observations

Preliminary deployment of the proposed framework demonstrates:

- **Overall Accuracy**: 96% accuracy in generating correct SQL queries for L1/L2 query types
- **Query Volume Reduction**: 42% decrease in queries escalated to data teams

- **Execution Success Rate**: 96% of generated SQL queries execute without errors
- **User Satisfaction**: Weighted average satisfaction score of 4.2/5.0
- **Table Selection Accuracy**: 94% precision in selecting relevant tables
- **Multi-Turn Coherence**: 91% consistency across conversation turns

8.2 Key Accuracy Drivers

Heuristics and personalization serve as the major anchors for achieving 96% accuracy. The ablation studies (below) demonstrate that removing either heuristics or personalization significantly degrades system performance. Heuristics capture learned patterns from successful query execution, while personalization adapts to user-specific query patterns and domain conventions. Together, these components enable the system to overcome the challenge of ambiguous natural language input and achieve the high precision required for L1/L2 query execution.

A critical challenge that persists is the **wrong initial query problem**: when users begin with an ambiguous or imprecise query, the system may generate an incorrect SQL translation that makes it extremely difficult for the end user to recover through iterative refinement. This highlights the importance of interactive clarification mechanisms and proactive disambiguation during the initial query interpretation phase.

8.3 Ablation Studies

Ablation studies reveal the contribution of each component:

Configuration	Accuracy	User Satisfaction
Full System	96%	4.2/5.0
Without Heuristics	81%	3.5/5.0
Without Ontology	77%	3.3/5.0
Without Personalization	90%	3.8/5.0
Without Crawling	88%	3.9/5.0

These results validate the importance of each architectural component. The substantial degradation without heuristics (15% drop in accuracy) and personalization (6% drop) confirms their role as accuracy anchors, enabling the system to achieve 96% accuracy.

8.4 Challenges and Future Work

Key challenges include: (1) cold start for new users without historical data; (2) maintaining ontology-schema synchronization; (3) scalability of heuristic computation; and (4) providing transparent query reasoning explanations. Future work will explore reinforcement learning for heuristic optimization, federated learning for cross-client knowledge transfer, active learning for high-value query annotation, and causal reasoning to distinguish correlation from causation in pattern extraction.

9 Conclusion

We have presented a comprehensive framework for natural language to SQL translation that integrates heuristic stores, ontological knowledge representation, and multi-dimensional evaluation. The system achieves continuous improvement through generative data feedback loops and automated knowledge extraction. Mathematical formulations for table likelihood and query

confidence provide principled foundations for probabilistic reasoning over database schemas. Heuristics and personalization serve as key accuracy anchors, enabling the framework to achieve 96% accuracy for L1/L2 query types. By aligning evaluation metrics with business value propositions, the framework ensures that technical improvements translate directly into operational benefits and establishes a robust foundation for intelligent, context-aware database query systems.

10 Acknowledgments

We acknowledge several papers we have read to come to such conclusion along with amazing organization who gave a chance to achieve such accuracy.