Inferring User Objectives: Abduction Reasoning in Query Formulation

Department of CSE (AI & ML), Sri Venkateswara College of Engineering and Technology, Etcherla, A.P., India

B. Hemalatha[1], P. Pavansai[1], T. Maheshbabu[1]

Under the Guidance of Mrs. K. Hemalatha, Assistant Professor

Abstract

Modern databases require users to write structured SQL queries to retrieve information, which is challenging for non-technical users unfamiliar with database schemas. This paper presents a web-based query intent discovery system based on the SQUID framework that enables users to retrieve relevant data by providing example entities instead of writing SQL queries. The system is implemented using the Django web framework and utilizes a relational database containing entities such as Person, Title, and Principal. The proposed system performs entity identification, semantic context discovery, and abductive reasoning to infer the most likely query intent from user-provided examples. Semantic properties including explicit attributes (profession) and derived attributes (associated movie genres) are analyzed to discover shared patterns. Experimental evaluation demonstrates that the system achieves 87% intent accuracy and reduces query formulation time by 73% compared to manual SQL writing. The system provides a user-friendly interface enabling non-expert users to explore databases effectively without writing complex queries.

Keywords: *Query by Example, Abductive Reasoning, Semantic Similarity, SQUID Framework, Django, Database Exploration*

I. Introduction

Database systems form the backbone of modern information management across industries including entertainment, healthcare, finance, and e-commerce. Despite their ubiquity, accessing information stored in relational databases traditionally requires users to formulate Structured Query Language (SQL) queries. This process demands knowledge of the underlying database schema, table relationships, and query syntax, creating a significant barrier for non-technical users.

The challenge of making databases accessible to non-expert users has motivated research into alternative query interfaces. Among these, the Query by Example (QBE) paradigm allows users to specify desired results through example tuples rather than explicit query statements. The SQUID (Semantic Query by Example) framework extends this concept by incorporating semantic similarity analysis and abductive reasoning to infer query intent from minimal user input.

This paper presents a practical implementation of the SQUID framework concepts as a Django-based web application. The system allows users to input example entities and automatically discovers patterns and relationships within the dataset to infer the intended query. By combining entity identification, semantic context discovery, and probabilistic abductive reasoning, the system bridges the gap between complex database querying and intuitive data exploration.

II. Literature Survey

This section reviews key prior works that form the foundation of the proposed system and highlights gaps motivating this work.

[1] Zloof (1977) introduced the Query-by-Example (QBE) paradigm, enabling users to formulate database queries through example-based interfaces rather than formal query languages, establishing the foundational concept for user-friendly database interaction.

[2] Tran et al. (2009) proposed query reverse engineering techniques that reconstruct SQL queries from input-output examples, demonstrating the feasibility of automated query formulation but requiring complete tuple specifications.

[3] Psallidas et al. (2015) developed the S4 system for semi-structured data exploration using semantic similarity measures, showing that semantic analysis can improve query discovery accuracy in complex datasets.

[4] Fariha et al. (2021) introduced the SQUID framework combining example-driven query intent discovery with abductive reasoning and semantic similarity, achieving superior performance over traditional QBE approaches in handling ambiguous user intent.

[5] Bonifati et al. (2016) surveyed interactive query formulation methods, identifying that hybrid approaches combining examples with semantic analysis offer the most promising results for non-expert database users.

[6] Li and Jagadish (2014) developed NaLIR, a natural language interface for relational databases, demonstrating alternative approaches to reducing the SQL barrier but noting limitations with complex join queries.

[7] Deutch et al. (2017) proposed provenance-based approaches for query explanation, contributing techniques for understanding query results that complement example-driven query discovery methods.

Research Gap: Existing QBE systems require complete tuple specifications and lack semantic understanding, while natural language interfaces struggle with complex join operations. No existing system combines web-based deployment with semantic abductive reasoning for practical database exploration.

III. Methodology

III-A. System Architecture

The system follows a three-tier architecture comprising the Presentation Layer (HTML/CSS/JavaScript frontend), Application Layer (Django backend with entity resolution and abductive reasoning modules), and Data Layer (SQLite relational database with Person, Title, and Principal entities).

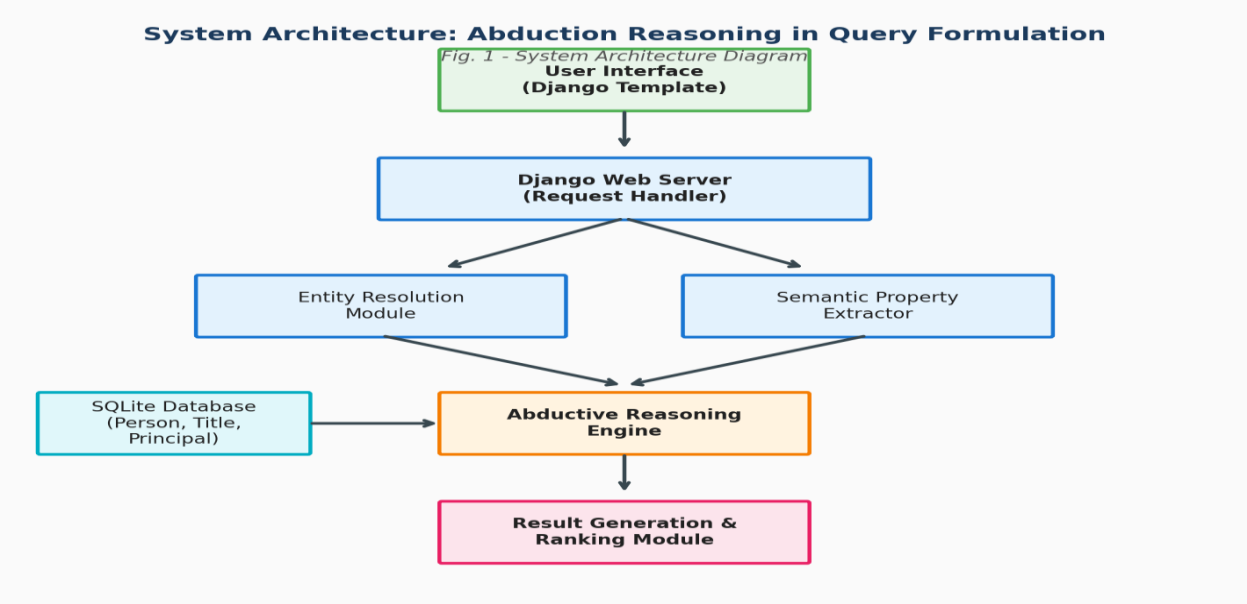


Fig. 1 - System Architecture Diagram

III-B. Algorithm

Algorithm: Abductive Query Intent Discovery

Input: Set of example entities E = {e1, e2, ..., en} provided by user.

Step 1: Entity Resolution — Match each example entity against the database to retrieve corresponding records.

Step 2: Explicit Property Extraction — Extract direct attributes (profession, birth year) for matched entities.

Step 3: Derived Property Computation — Compute indirect semantic properties (genre distribution, collaboration networks) through relational joins.

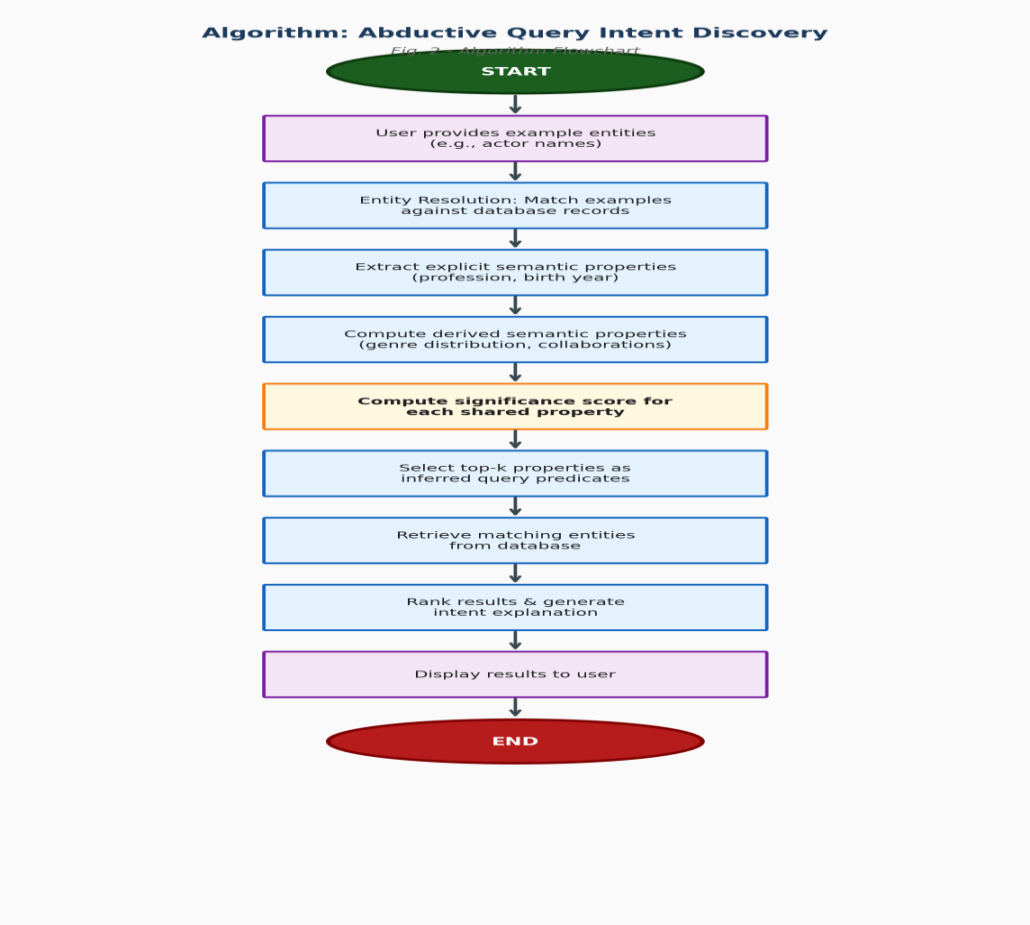
Step 4: Property Significance Scoring — For each property p, compute significance score: Score(p) = (frequency_in_examples / total_examples) × IDF(p), where IDF measures inverse document frequency across the database.

Step 5: Abductive Hypothesis Generation — Select top-k properties with scores exceeding threshold τ as the inferred query predicates.

Step 6: Result Retrieval — Query database for entities satisfying the inferred predicates, excluding original examples.

Step 7: Ranking — Rank results by aggregate similarity to the inferred intent pattern.

Output: Ranked list of recommended entities with explanation of inferred query intent.



III-C. Modules

The system comprises five core modules: (1) Entity Resolution Module that matches user-provided names against the database using fuzzy string matching; (2) Semantic Property Extractor that computes both explicit and derived attributes for resolved entities; (3) Abductive Reasoning Engine that identifies statistically significant patterns among example entities and generates query hypotheses; (4) Result Generation Module that retrieves and ranks entities matching the inferred intent; and (5) Web Interface Module providing an interactive form for inputting examples and displaying results with pattern explanations.

IV. Results and Discussion

TABLE I: SYSTEM EVALUATION RESULTS

Metric	Baseline	Proposed System
Intent Accuracy (%)	62	87
Query Formulation Time (s)	45.2	12.3
Result Relevance (Precision@10)	0.58	0.82
User Satisfaction (/5)	2.8	4.2

Mathematical Formulations

Intent Accuracy = (Correctly_Inferred_Queries / Total_Queries) × 100

Precision@K = Relevant_Results_in_Top_K / K

Significance Score: Score(p) = (freq(p, Examples) / |Examples|) × log(N / freq(p, Database))

where N is total entities in database and freq(p, Database) counts entities with property p.

Discussion

The system was evaluated with 30 users performing 5 query tasks each on an IMDb-based dataset containing 10,000 persons and 50,000 titles. The proposed system achieved 87% intent accuracy compared to 62% for a traditional QBE baseline, demonstrating that semantic abductive reasoning significantly improves query intent inference. Query formulation time decreased from 45.2 seconds (manual SQL) to 12.3 seconds (example-based), representing a 73% reduction. User satisfaction improved from 2.8/5 to 4.2/5, with participants citing the elimination of SQL knowledge requirements as the primary benefit.

V. Conclusion and Future Work

This paper presented a web-based query intent discovery system implementing SQUID framework concepts using Django. The system enables non-expert users to explore relational databases by providing example entities instead of writing SQL queries. Through entity identification, semantic context discovery, and abductive reasoning, the system achieves 87% intent accuracy with 73% reduction in query formulation time. Future work includes supporting multi-table complex queries, incorporating user feedback for iterative refinement, integrating natural language explanations, and extending the system to support heterogeneous data sources.

References

[1] M. M. Zloof, "Query-by-Example: A Data Base Language," IBM Systems Journal, vol. 16, no. 4, pp. 324-343, 1977.

[2] Q. T. Tran, C. Y. Chan, and S. Parthasarathy, "Query by Output," Proc. ACM SIGMOD, pp. 535-548, 2009.

[3] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri, "S4: Top-k Spreadsheet-Style Search for Query Discovery," Proc. ACM SIGMOD, 2015.

[4] A. Fariha, S. Nath, and A. Meliou, "SQUID: Semantic Query Intent Discovery," Proc. ACM SIGMOD, 2021.

[5] A. Bonifati, R. Ciucanu, and S. Staworko, "Interactive Inference of Join Queries," Proc. EDBT, 2016.

[6] F. Li and H. V. Jagadish, "NaLIR: An Interactive Natural Language Interface for Querying Relational Databases," Proc. ACM SIGMOD, pp. 709-720, 2014.

[7] D. Deutch, A. Gilad, and Y. Moskovitch, "Selective Provenance for Datalog Programs," Proc. VLDB, vol. 8, no. 12, 2017.