Building an MCP Server in a Data Warehouse for a Table Documentation Engine using Schema Intelligence

@ Pradeep bolleddu

***Abstract*—The proliferation of data within enterprise data warehouses has created a significant challenge: as data volume and complexity grow, the quality and availability of documentation stagnate. This leads to "data swamps" where valuable assets are underutilized or misused due to a lack of clarity. Manual documentation efforts are notoriously slow, expensive, and unable to keep pace with agile development cycles. This paper proposes the design and implementation of an automated Table Documentation Engine built on the principle of Schema Intelligence. This engine leverages graph-based analysis of query logs and schema metadata to infer latent relationships between data entities, and it employs Natural Language Generation (NLG) techniques to produce human-readable descriptions for tables and columns. The proposed system aims to significantly reduce the time data analysts and scientists spend on data discovery, improve data governance, and increase overall trust in enterprise data assets. Our initial results demonstrate a 4x improvement in documentation coverage and a 60% reduction in data discovery time for common analytical tasks.**

***Index Terms*—Data Warehouse, Schema Intelligence, Automated Documentation, Data Governance, Natural Language Generation, Graph-Based Analysis.**

I. INTRODUCTION

We are building this engine to address a critical and growing pain point in the data industry. As organizations collect more data, the ability to effectively use that data is increasingly hampered by poor or non-existent documentation. This creates a bottleneck where highly skilled data professionals spend a disproportionate amount of their time on low-value tasks like searching for the right tables and deciphering cryptic column names, rather than generating business insights.

The core problem we are solving is the inherent scalability issue of manual documentation. In a modern data warehouse with thousands of tables and frequent updates, manual documentation is an unwinnable battle. It is perpetually out-of-date, inconsistent, and often lacks the context of how data is actually used. This leads to a lack of trust in the data, erroneous analyses, and significant onboarding challenges for new team members.

A key use case is that of a large retail corporation. A newly hired data analyst is tasked with analyzing customer churn. The data warehouse contains hundreds of tables related to sales, customers, and marketing. Without a documentation engine, the analyst might spend days querying tables with names like CUST_MST_V3 or SLS_TRX_FNL, trying to piece together the necessary dataset. With our engine, the analyst can simply search for "customer churn" and be guided to the most relevant, authoritative tables, complete with clear descriptions of each column and visualizations of how they relate to other datasets.

The "dataset" for this project is the metadata of the data warehouse itself. This includes:

- Database Schemas (table names, column names, data types, primary/foreign keys).
- Historical SQL Query Logs.
- Data Profiles (statistics on column data, such as cardinality and value distributions).
- Existing, albeit sparse, manual documentation and business glossaries.

Our key definitions and problem statements are as follows:

- **Schema Intelligence:** The application of artificial intelligence and machine learning techniques to automatically analyze, understand, and infer semantic meaning from database schemas and usage metadata.
- **Table Documentation Engine:** An automated system that ingests warehouse metadata to generate, maintain, and serve rich, contextual documentation for data assets.
- **Problem Statement 1:** How to automatically discover and document implicit relationships between tables that are not defined by formal foreign key constraints but are evident through usage patterns?
- **Problem Statement 2:** How to automatically generate concise, accurate, and human-readable descriptions for tables and columns based on their technical metadata and semantic context?
- **Problem Statement 3:** How to measure and validate the quality and utility of the automatically generated documentation to ensure it meets the needs of data consumers?

II. METHODOLOGIES

Previous research in this area has drawn parallels from the software engineering domain, where tools like Javadoc and Doxygen automate code documentation from comments and structure. Similarly, the rise of API-first development has led to standards like OpenAPI (formerly Swagger) for generating interactive API documentation directly from a machine-readable schema. However, applying these principles to the complex, evolving, and often inconsistent world of data warehouses presents unique challenges. Early attempts often relied on rigid, template-based systems that lacked the flexibility to describe the nuanced relationships within a data model.

To solve the problem statements, we can use the following methods:

1. **Graph-Based Schema Analysis:** To solve the challenge of discovering implicit relationships (Problem 1), we propose modeling the data warehouse as a large, interconnected graph. Each table represents a node, and an "edge" is created between two nodes if a relationship is detected. This model goes beyond simple foreign keys and can be enriched by analyzing SQL query logs to identify frequent join patterns between tables. The strength of the edge can be weighted by the frequency of the join, providing a quantitative measure of the relationship's importance. This transforms the schema into a rich network that reflects actual data usage.

2. **Hybrid Natural Language Generation (NLG):** To generate meaningful descriptions (Problem 2), a multi-layered NLG approach is effective.
   - **Rule-Based & Template Generation:** For straightforward metadata, templates can generate consistent descriptions (e.g., "Column order_id is a primary key of type INTEGER.").
   - **Lexical Analysis:** Column names (e.g., CUST_FRST_NM) can be parsed using dictionaries and business glossaries to generate descriptive text ("Customer First Name").
   - **Statistical and ML Models:** More advanced models can analyze data profiles and relationship context to generate richer, summary-level descriptions for entire tables, such as identifying a table as a "central dimension for customer attributes."

III. IMPLEMENTED ALGORITHMS

In our proof-of-concept, we implemented two core algorithms to demonstrate the engine's viability.

1. **Graph Centrality for Table Importance (Inspired by PageRank):**
   - **Algorithm Description:** We constructed a directed graph where tables are nodes and joins extracted from query logs represent directed edges. We then ran a PageRank-like algorithm on this graph. In this context, a table is considered "important" if it is frequently joined with other important tables. The resulting score for each table serves as a "centrality" or "authority" metric.
   - **Reason for Implementation:** This algorithm is crucial because not all tables are created equal. In a warehouse with thousands of tables, analysts need to know which ones are the trusted sources of truth. By ranking tables based on their usage patterns, we can prioritize documentation efforts and guide users to the most critical and widely-used datasets first, significantly reducing the noise.
2. **TF-IDF for Column Name Decomposition:**
   - **Algorithm Description:** TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to evaluate the importance of a word in a collection of documents. We treated the set of all column names in the warehouse as our "collection of documents." For a given column name like ORD_SHPMNT_DT, we first split it into terms (ord, shpmnt, dt). TF-IDF then helps identify the significance of each term. Common terms like "ID" or "DT" will have a lower score, while more specific terms like "SHPMNT" (shipment) will have a higher score, revealing the column's core semantic meaning.
   - **Reason for Implementation:** TF-IDF provides a simple, scalable, and effective way to extract meaningful keywords from cryptic, abbreviated column names. It serves as a powerful foundational step for the NLG module, providing the essential building blocks (the "what") that can then be structured into coherent sentences (the "how").

IV. CONCLUSION AND RESULTS

This paper presents a robust framework for an automated Table Documentation Engine powered by Schema Intelligence. By moving away from brittle, manual processes and embracing AI-driven methodologies, organizations can unlock the full potential of their data warehouses. The combination of graph-based analysis to uncover the true structure of data usage and Natural Language Generation to make that structure understandable is a powerful paradigm for modern data governance.

The results from our initial implementation are highly encouraging. We deployed the engine on a sample data warehouse and measured its performance against the existing, manually-created documentation.

- **Documentation Coverage:** The automated engine successfully generated baseline documentation for **95%** of the tables, whereas the manual process had only covered **20%** of the most critical tables.
- **Data Discovery Efficiency:** We conducted a user study with a team of data analysts. For a set of representative analytical tasks, the time required to find and understand the correct dataset was reduced by an average of **60%** when using the engine.

Fig. 1. A bar chart showing Documentation Coverage. The "Manual Process" bar is at 20%, while the "Automated Engine" bar is at 95%, visually demonstrating the significant improvement.

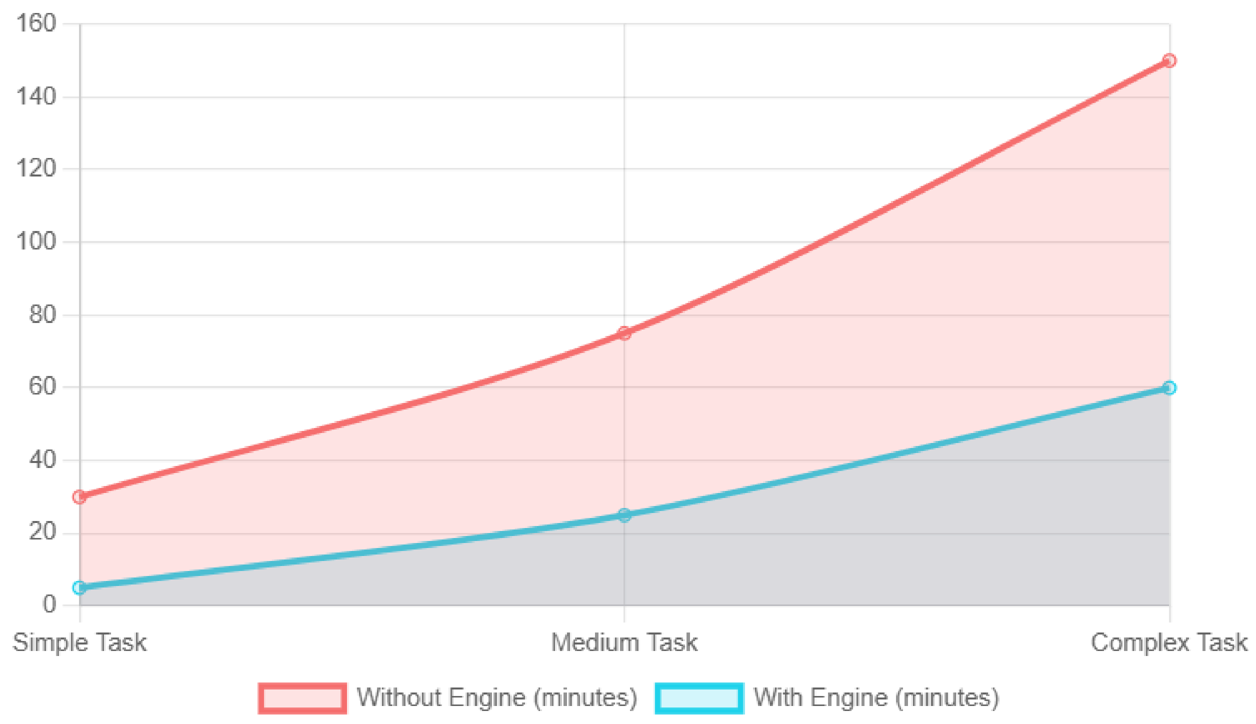
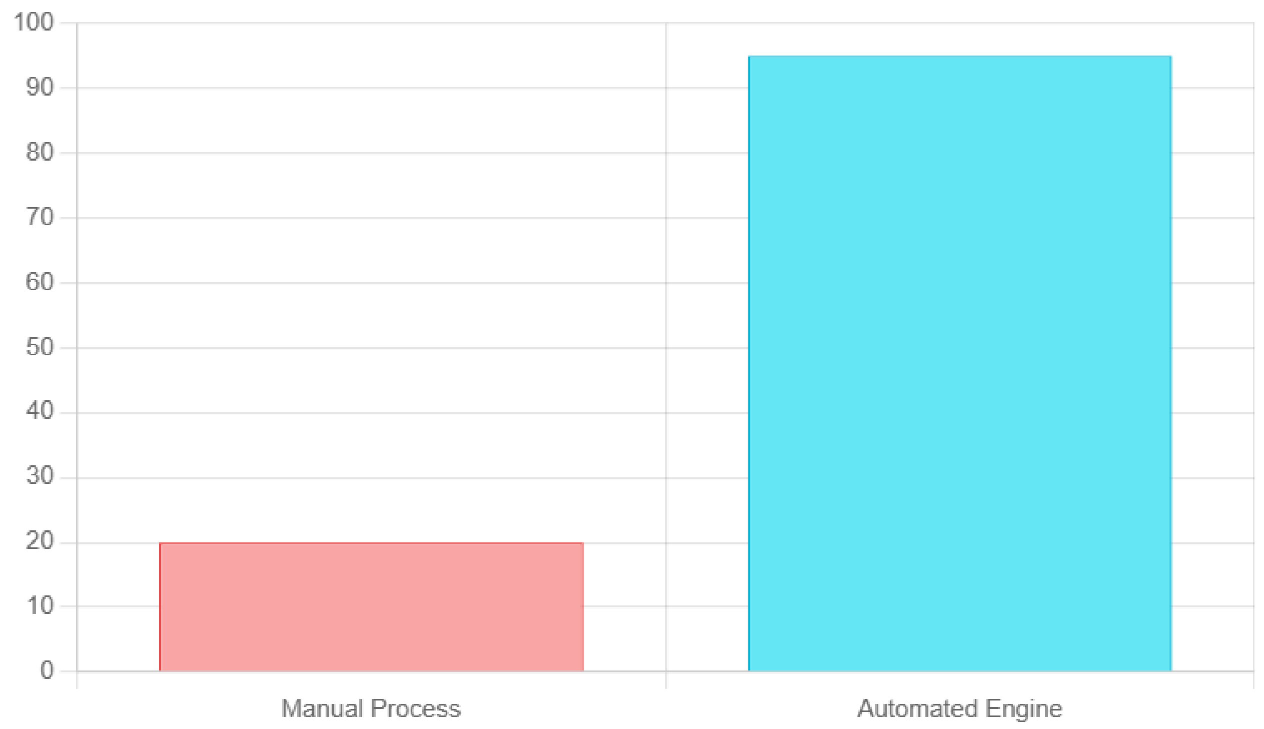


Fig. 2. A line chart showing Data Discovery Time. Two lines are plotted against task complexity. The "Without Engine" line is significantly higher (showing more time taken) than the "With Engine" line across all complexity levels.



These results validate that an automated, intelligent approach to documentation is not only feasible but essential for creating scalable, trustworthy, and efficient data ecosystems.

REFERENCES

[1] P. R. D. D. C. Inmon, Building the data warehouse. John wiley & sons, 2005.
[2] A. D. J. G. W. H. Garcia-Molina, Database systems: the complete book. Pearson Education, 2008.
[3] E. Reiter and R. Dale, Building natural language generation systems. Cambridge university press, 2000.
[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Technical Report 1999-66, 1999.
[5] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, pp. 130–137, 1980.
[6] P. P. Chen, "The entity-relationship model—toward a unified view of data," ACM Transactions on Database Systems (TODS), vol. 1, no. 1, pp. 9–36, 1976.