Towards an SLM-based Auditing of Relational Schemas and Data Quality for Practical Data Governance

Antony Seabra
BNDES - Social and Economic Development Bank of Brazil
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
amedeiros@inf.puc-rio.br

Claudio Cavalcante
BNDES - Social and Economic Development Bank of Brazil
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
cfrag@bndes.gov.br

Nicolaas Ruberg
BNDES - Social and Economic Development Bank of Brazil
Rio de Janeiro, Brazil
nic@bndes.gov.br

Sergio Lifschitz
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
sergio@inf.puc-rio.br

ABSTRACT

Ensuring the integrity and quality of relational schemas and their underlying data is essential for effective data governance in large-scale applications. This paper introduces an automated approach for schema and data quality auditing in relational databases, leveraging the capabilities of Small Language Models (SLMs), running in an on-premises infrastructure for compliance reasons. Our method automates the generation of analysis scripts to detect and report quality issues, producing both a composite quality score and actionable recommendations for improvement. The proposed solution addresses real-world governance challenges by analyzing 400 diverse production-grade schemas. The evaluation reveals common issues such as missing key constraints, normalization violations, data type inconsistencies, and integrity breaches. Notably, our experiments show that SLM-generated analysis scripts effectively identified issues, contributing to the resolution of 58.7% of the reported issues, enhancing overall data reliability and supporting more efficient data engineering workflows.

VLDB Workshop Reference Format:
Antony Seabra, Claudio Cavalcante, Nicolaas Ruberg, and Sergio Lifschitz. Towards an SLM-based Auditing of Relational Schemas and Data Quality for Practical Data Governance. VLDB 2025 Workshop: 14th International Workshop on Quality in Databases (QDB'25).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/antonyseabramedeiros.

1 INTRODUCTION

The escalating volume and complexity of data in modern enterprises necessitate robust and scalable data governance practices. Central to this is the assurance of both relational schema integrity and high data quality. Traditional analysis methods, reliant on manual inspection or rigid rule-based systems, struggle to keep pace with the dynamic nature and scale of real-world databases, often leading to data inconsistencies, inaccuracies, and ultimately, compromised decision-making. This paper addresses these challenges by presenting an evaluated solution for automated analysis of relational schemas and data quality, leveraging the semantic understanding capabilities of Small Language Models (SLMs), running on-premises for compliance reasons, for practical data governance. Our approach introduces an SLM-based framework designed to automatically assess relational schema integrity, normalization, and data quality. The system extends beyond assessment, providing actionable improvement suggestions for schema refinement and data cleaning. This enables data engineers to efficiently address identified issues, enhancing data reliability and streamlining data governance workflows.

The novelty of this work lies in the practical application of SLMs to automate schema and data quality analysis in real-world databases. By harnessing the contextual understanding of SLMs, improved with few-shot examples, we reduce the immediate need for extensive fine-tuning for every new issue type, enabling the identification of subtle anomalies and inconsistencies that are critical for effective data governance. We demonstrate the practical impact of our solution through extensive experiments on diverse, real-world databases, showcasing its ability to accurately identify data quality issues, generate meaningful quality scores, and provide actionable improvement suggestions. Our key contributions are the following.

C1. An evaluated SLM-based solution for automated analysis of relational schemas and data quality, designed for practical data governance.

C2. A comprehensive scoring mechanism that quantifies the overall quality of a database, reflecting schema integrity and data quality, for practical application.

C3. Actionable improvement suggestions generated by the SLM, enabling efficient data engineering workflows.

C4. An extensive experimental evaluation on 400 real-world schemas, representing 10 terabytes of data, demonstrating the efficacy and scalability of our approach in practical settings.

Our approach leverages SLMs to automate database analysis and reduce costs by minimizing reliance on proprietary tools and manual expert reviews. This enables faster feedback, improves data

Proceedings of the VLDB Endowment. ISSN 2150-8097.

governance, and makes high-quality schema validation accessible to organizations with limited resources. The remainder of this paper is organized as follows: Section 2 presents the concepts and techniques used in this work. Section 3 discusses related work. Section 4 presents our methodology, and Section 5 details our experiments and results. Finally, Section 6 concludes the paper and suggests directions for future work.

2 BACKGROUND

While Large Language Models (LLMs) have undeniably revolutionized natural language processing and code generation, their immense computational requirements, often proprietary nature, and inherent challenges with data residency frequently limit their direct applicability in environments with strict compliance regulations and on-premises infrastructure mandates. This has led to the emergence of Small Language Models (SLMs), which are pre-trained models with significantly fewer parameters (typically ranging from a few hundred million to under 10 billion) compared to their colossal counterparts. Despite their smaller scale, SLMs leverage similar transformer architectures and attention mechanisms [21] to learn complex linguistic patterns and semantic relationships, often demonstrating surprisingly competitive performance on a wide array of tasks when properly fine-tuned or given sufficient in-context examples.

The primary advantage of SLMs lies in their computational efficiency and reduced resource footprint. This allows them to be deployed and run effectively on standard server hardware, often without the need for specialized, expensive Graphics Processing Units (GPUs) or extensive cloud infrastructure. This capability is paramount for on-premises deployment, a critical requirement for organizations dealing with sensitive or proprietary data where compliance regulations (e.g., GDPR, HIPAA, internal data governance policies) dictate that data must remain within the organization's controlled environment. Running SLMs locally ensures maximum data privacy and security, as sensitive relational schemas and data quality insights never traverse external networks, providing full control over the model, its data interactions, and its lifecycle.

However, using SLMs also comes with inherent trade-offs when compared to larger LLMs. While more resource-efficient, SLMs generally possess less broad world knowledge and exhibit reduced zero-shot generalization capabilities. Their performance on highly complex, nuanced, or abstract tasks may lag behind models like GPT-4 or Gemini. This necessitates a more focused approach: SLMs often require more precise prompt engineering, higher-quality few-shot examples, or domain-specific fine-tuning to achieve optimal performance on specialized tasks such as relational schema analysis and data quality auditing. For our methodology, the on-premises deployability of SLMs is a cornerstone. Their ability to analyze relational schemas by interpreting SQL Data Definition Language (DDL) and inferring semantic relationships, coupled with their capacity to generate executable code (such as SQL scripts for data quality checks), makes them ideal for automating auditing processes within secure, internal infrastructures. While larger models might require less explicit guidance, the cost, security, and compliance benefits of SLMs, combined with targeted strategic integration, make them the superior choice for practical data governance applications.

Advances in Text-to-SQL have demonstrated that language models, including Small Language Models (SLMs), are capable of accurately extracting and interpreting schema structures such as table and column names, data types, and primary-foreign key relationships directly from SQL definitions. While these capabilities were initially applied to support natural language query generation, they also lay the foundation for more advanced tasks, such as relational schema analysis. By understanding how entities are organized and interrelated within a database, SLMs can be prompted not only to generate executable queries, but also to detect design flaws, such as missing constraints, normalization violations, and schema inconsistencies. Relational schema analysis is essential for ensuring the integrity and consistency of database designs. This process involves analyzing schemas to identify potential issues, such as missing primary or foreign keys, violations of integrity constraints, and deviations from normalization principles. Proper schema design is critical for data consistency, performance, and maintainability. In data warehouses, where data is often aggregated and transformed from multiple sources, schema analysis becomes even more critical. Different modeling approaches, such as star and snowflake schemas, introduce unique analysis considerations. For instance, ensuring dimensional consistency and fact table integrity is vital in data warehouse environments.

Data quality and governance are critical for reliable, compliant data use. Quality includes completeness, accuracy, consistency, and validity, while governance defines processes to manage and protect data. Traditional tools struggle with modern scale and complexity, prompting the use of AI and language models to detect flaws and suggest improvements. In this context, data stewards play a key role by defining domain-specific reference rules (e.g., valid ranges), which guide automated checks and enhance trust in data-driven processes.

3 RELATED WORK

Our work builds upon a rich body of research in relational schema analysis and data quality assessment. Traditional approaches to relational schema analysis and constraint discovery have relied on rule-based techniques and manual inspections. Notable works include [2, 13], which focus on schema matching and integration using data dependencies and logical mappings. AUDITOR [10] and LedgerDB [23] further extend these ideas through automatic integrity constraint discovery and provenance-based verification. The assessment of data quality has been widely explored, with research addressing anomaly detection [7, 8], consistent query answering [12], and large-scale verification frameworks [17]. Holoclean [7] and similar systems aim to automate data cleaning, but often require significant domain configuration. Broader frameworks on data governance have focused on policy-driven automation and lineage tracking [11, 19]. These efforts support trust in data pipelines, but still depend heavily on fixed rules and human oversight. Recent studies have explored the potential of Large Language Models (LLMs) for database-related tasks, such as schema summarization [14] and Text-to-SQL generation [24]. While promising, these applications mostly address comprehension and querying, not direct schema or data quality assessment. Our work complements this literature by applying SLMs to directly audit schemas and data,

extending the role of language models from interface assistants to internal quality assessors.

Beyond these foundational techniques, recent advancements in Large Language Models (LLMs) have opened new avenues for automating database tasks. Studies like [5, 14] have explored the use of LLMs for schema summarization, demonstrating their potential to generate human-readable descriptions of database schemas. Similarly, Text-to-SQL systems leverage LLMs to interpret natural language questions and generate corresponding SQL queries, requiring a thorough understanding of the underlying schema [15, 16, 24]. While these approaches contribute significantly to schema comprehension and query generation, they generally do not encompass the comprehensive schema analysis (e.g., constraint validation, normalization assessment, redundancy detection) or direct data quality assessment that our SLM-based methodology tackles, representing a notable gap in the existing literature. The application of LLMs, and particularly SLMs, to direct relational schema and data quality analysis is an emerging area of research that specifically targets this gap. While SLMs demonstrate promising capabilities in semantic understanding and code generation, the existing literature reveals a significant lack of research in their direct and comprehensive application to automated data quality assessment and improvement in relational databases. Traditional data quality analysis often relies on time-consuming rule-based systems or manual inspections [7, 8]. Recent works, such as those by [17] on large-scale data quality verification, [22] on data quality challenges in deep learning, and [3] on ethical dimensions of data quality, highlight the complexities and necessities of robust data quality practices, emphasizing the need for scalable and automated solutions beyond traditional methods.

Related to improving the accuracy and robustness of Text-to-SQL systems, particularly in handling complex queries and diverse database schemas. For example, studies have explored the use of schema linking and graph-based representations to enhance schema understanding [24]. Additionally, research has focused on addressing challenges related to handling ambiguous queries and generating efficient SQL statements [6, 18]. However, current LLM solutions for Text-to-SQL still face limitations in generating highly complex queries that involve multiple joins, nested subqueries, and intricate aggregation functions, which are often necessary for comprehensive database analysis. Recent research has focused on developing automated schema analysis tools that can identify anomalies and suggest improvements [1, 20]. Techniques like constraint discovery and schema matching have been explored to automate the analysis process [2, 13]. Furthermore, research has addressed the challenges of analysis evolving schemas in dynamic environments [4, 9]. Automated schema analysis tools are crucial for ensuring that schemas adhere to best practices and meet the evolving needs of data-driven applications. However, traditional schema analysis tools often rely on rigid rule-based systems that struggle to adapt to the dynamic nature and complexity of modern databases, necessitating more flexible and intelligent approaches.

Our approach directly addresses this unfulfilled need by harnessing the semantic understanding and code generation capabilities of SLMs. We focus on automating complex data validation, constraint checking, and overall quality assessment within relational databases, which often involve domain-specific knowledge and intricate data relationships.

4 METHODOLOGY

A key point of our work is the choice for Small Language Models. For organizations prioritizing data governance and stringent compliance, deploying SLMs on-premises offers a compelling solution for analyzing relational schemas and data quality. This strategy ensures that sensitive database metadata and proprietary business rules never leave the controlled internal network, directly addressing concerns around data residency, privacy regulations like GDPR, and internal security policies. A distinct approach to leverage these on-premises SLMs for schema and data quality analysis involves using dynamic prompts infused with few-shot example. In our methodology, for each specific relational schema or data quality issue to be addressed, the prompt is crafted to include a small, representative set of input-output examples. Importantly, when assessing a database, we do not know the issues in advance; once the model outputs its results, we manually inspect and validate whether the findings are correct or not.

4.1 Relational Schema Analysis

The system architecture comprises four main components: a Metadata Extractor, a Dynamic Prompt Generator with a Prompt Library, an SLM Analyzer and a Report Generator. It all begins with an Audit Request. The system then establishes a database connection using appropriate connectors and drivers. The Metadata Extractor retrieves schema metadata by querying the database for information on tables, columns, keys, and constraints. In parallel, the Dynamic Prompt Generator creates tailored prompts based on the characteristics of the database. The schema metadata, along with the generated prompts, is fed to the SLM Analyzer, which use a model to analyze the schema metadata guided by the instructions in the prompt, to identify potential issues such as missing primary keys, lack of foreign key constraints, or suboptimal normalization. The SLM's understanding of database design principles enables it to recognize and report on these issues. The Dynamic Prompt Generator is employed to create specific prompts for the SLM, tailoring the analysis to the particular schema and analysis request, and the available prompt library. This dynamic approach allows for a more focused and effective analysis. The findings of the SLM analysis are then compiled into an Audit Report by the Report Generator. This report provides a comprehensive overview of the identified schema issues, along with recommendations for improvement. We illustrate the application of our methodology to resolve common database scenarios encountered by organizations.

4.1.1 Missing Foreign Key. In this scenario, we have two tables: Customers and Orders. The Orders.CustomerID column lacks a foreign key referencing Customers.CustomerID. The Metadata Extractor retrieves schema details, and a Dynamic Prompt Generator creates a prompt asking the system to identify potential missing foreign keys. The SLM Analyzer uses this prompt to analyze the schema, detects the missing foreign key relationship, and suggests a SQL statement to fix it by adding the appropriate constraint:

```
ALTER TABLE Orders
ADD CONSTRAINT FK_Orders_Customers
FOREIGN KEY (CustomerID) \\
REFERENCES Customers(CustomerID);
```

The Report Generator incorporates the SLM's findings into a final Audit Report, highlighting the missing foreign key on the Orders table, explains potential risks like data inconsistencies or orphaned records, and presents a recommended SQL statement to fix the issue.

4.1.2 Normalization Issues. In a more complex customer-order schema involving Customers, Orders, and OrderDetails tables, normalization issues are identified. The OrderDetails table contains redundant product information (ProductName, Price), which should be stored in a separate Products table. This redundancy violates Second Normal Form, leading to potential data anomalies and inefficiencies. A Metadata Extractor gathers schema details, and a Dynamic Prompt Generator crafts a prompt aimed at detecting normalization issues. The SLM Analyzer, using this prompt, detects that ProductName and Price are functionally dependent on ProductID and should be separated. It classifies this as a normalization anomaly and recommends restructuring the schema by creating a Products table and modifying OrderDetails to reference it via ProductID. The SLM also generates SQL statements to implement the fix.

```
CREATE TABLE Products (ProductID INT PRIMARY KEY,
ProductName VARCHAR(255), Price DECIMAL(10, 2));

INSERT INTO Products (ProductID, %ProductName, Price)
SELECT DISTINCT ProductID, ProductName, %Price
ROM OrderDetails;

ALTER TABLE OrderDetails %DROP COLUMN ProductName,
DROP COLUMN Price;

ALTER TABLE OrderDetails ADD CONSTRAINT
FK_OrderDetails_Products FOREIGN KEY (ProductID)
REFERENCES Products(ProductID);
```

4.1.3 Entity Duplication Across Schemas. In a scenario where a company has two separate databases with similar customer data, one using a Customer table (CustomerID, CustomerName, CustomerAddress), and the other a Clients table (ClientID, ClientName, ClientLocation), schema duplication issues arise. These redundant entity definitions lead to data inconsistencies, reporting challenges, and integration difficulties. A Metadata Extractor gathers schema details from both databases. A Dynamic Prompt Generator then creates a prompt aimed at identifying semantically similar entities across schemas. The SLM Analyzer, guided by this prompt, compares table and column names, data types, and descriptions. Based on this analysis, it infers that the Customers and Clients tables likely represent the same business entity— customers— but are defined differently.

- A column mapping between the two tables (e.g., `CustomerID` ↔ `ClientID`, `CustomerName` ↔ `ClientName`).
- A strategy for consolidation, recommending unification into a single source of truth, such as a centralized table or data warehouse.
- SQL statements and data transformation suggestions to support data migration and harmonization.

The model expresses this classification as a textual explanation, which is then interpreted and categorized as a schema duplication anomaly. While the model does not return a numeric confidence score, we manually reviewed such outputs to validate their consistency and semantic plausibility. The SLM provides:

4.2 Data Quality Assessment

Extending the schema analysis framework, our methodology incorporates data quality assessment, generating Python or SQL code that, when executed, identifies specific data quality issues. Prompts guide the SLM to generate code for data quality verification and include few-shot examples of both flawed and clean data patterns, guiding the model toward generating appropriate validation logic. For instance, if the analysis request specifies a requirement for data completeness, the SLM might generate SQL code using conditions like `IS NULL` or aggregate counts to identify records with missing values in critical columns. To handle more complex issues, such as consistency checks or rule violations, the prompts include illustrative integrity constraints or example business rules. The generated code is reviewed by humans for syntax and logic correctness.

4.2.1 Date Range Validation. To ensure the consistency and logical validity of date ranges, the process begins with an Audit Request specifying the requirement that the final date must be equal to or greater than the initial date. The Metadata Extractor retrieves the relevant schema information, including the data types and constraints of the initial and final date columns. The Dynamic Prompt Generator then creates a targeted prompt for the SLM, instructing it to generate SQL code for date range validation.

The SLM Analyzer, equipped with this prompt and its knowledge of SQL date comparison operations, generates SQL code that defines a query to identify records where the final date is earlier than the initial date. This query typically uses date comparison operators (e.g., '>=', '<=') to check the logical relationship between the two date columns. The generated SQL code is then incorporated into the Audit Report, along with explanations of the validation criteria and potential implications of invalid date ranges. This report allows database administrators to review the code, execute it within their environment, and identify records that require correction.

4.2.2 Phone Number Standardization. Similarly, for phone number standardization, the Audit Request specifies the desired format for phone numbers. The Metadata Extractor retrieves the schema information for the phone number column. The Dynamic Prompt Generator creates a prompt instructing the SLM to generate Python code for phone number standardization.

```
def standardize_phone_number(phone_number):
  pattern = r"^(\d{3})\D*(\d{3})\D*(\d{4})$"
  match = re.search(pattern, phone_number)
  if match:
    return f"({match.group(1)}) {match.group(2)}-
    {match.group(3)}"
  return None
```

The SLM Analyzer, leveraging its knowledge of string manipulation techniques, generates Python code that defines a function to standardize phone numbers. This function typically uses string functions to extract and reformat the components of a phone number, such as the area code, prefix, and line number. The generated Python code is included in the Audit Report, along with explanations of the standardization process and potential benefits of

consistent phone number formats. Data stewards can then review and execute this code to standardize phone numbers within their environment.

4.2.3 Detecting Outliers. For outlier detection, the request specifies the criteria for identifying outliers, such as thresholds based on standard deviation. The Metadata Extractor retrieves the schema information for the order value column. The Dynamic Prompt Generator creates a prompt instructing the SLM to generate Python code for outlier detection. The SLM Analyzer, using its knowledge of statistical functions, generates Python code that defines a function to detect outliers in order values. This function typically calculates statistical measures like mean and standard deviation and identifies values that fall outside a defined range. The generated Python code is included in the report, along with explanations of the outlier detection method and potential implications of outliers. Data stewards can then review and execute this code to identify potential outliers in their order data.

4.3 Scoring Methodology

We employed a structured and customizable scoring framework to evaluate databases across ten categories—five related to schema analysis and five to data quality—each contributing 10% to a total score. This balanced 50/50 split reflects the dual importance of structural design and data content in overall reliability. For each category, the percentage of affected elements (e.g., tables with normalization issues) determines the score. While the default equal weighting offers a neutral and interpretable baseline, the framework supports reweighting based on database type, criticality, or organizational needs.

Within the relational schema domain, the evaluation encompassed five distinct categories. Firstly, *Naming Conventions* were assessed by evaluating the consistency and clarity of table and column names, with inconsistencies or ambiguities penalized for hindering maintainability. The assessment follows a predefined convention system specifying expected patterns, such as using `id_` for identifier columns. These rules are included in the few-shot examples used to prompt the SLM. Secondly, the identification of *Missing Primary or Foreign Keys* was conducted, pinpointing tables lacking essential constraints that could lead to data integrity violations. Thirdly, *Normalization Issues* were assessed based on adherence to 1NF, 2NF, and 3NF, detecting redundant structures (under-normalization) and excessive fragmentation (over-normalization). The evaluation considers whether the normalization level suits the schema's purpose, flagging both extremes as potential issues. Fourthly, *Schema Design Flaws* were identified, detecting structural inefficiencies or incorrect relationships, such as missing junction tables or improper entity relationships. Lastly, *Entity Duplication Across Schemas* was examined, identifying redundant entities spread across multiple schemas, which could result in inconsistencies and increased storage costs.

In the data quality domain, the evaluation also encompassed five distinct categories. Firstly, *Data Type Issues* were scrutinized, verifying the appropriateness of data types for stored values to ensure storage efficiency and prevent data corruption. Secondly, *Data Accuracy* was assessed using reference rules defined by data stewards, such as valid date or numeric ranges, to detect probable inaccuracies. As no gold standard is fully available, the SLM uses these references and heuristics to flag potential errors, which are then manually reviewed. Thirdly, *Data Integrity Issues* were examined, analyzing the enforcement of data constraints (e.g., unique, check, not null) to ensure data consistency and validity. Fourthly, *Data Standardization* was evaluated, assessing the uniformity of data representation across tables and columns to ensure consistency and facilitate data integration. Lastly, *Outliers Detection* was conducted, checking if data values fell within acceptable ranges to prevent outliers and invalid entries.

To illustrate the scoring methodology, we evaluated two versions of the open-source WebKeePass database: the original (WKP) and a revised version (WKP_v2), improved using suggestions from our system. WKP showed widespread issues across most categories, resulting in a total score of 55.4. In contrast, WKP_v2 achieved a lower score of 36.6, indicating better adherence to relational modeling and data quality best practices. Scores were based on normalized proportions of affected elements, and figure 2 presents category and total scores.

5 EVALUATION

Our experiments were carried out on a premises machine equipped with an Intel Xeon Gold 6248R CPU and 64GB of RAM, notably without a GPU. The software environment included Python 3.9 with relevant libraries and Ollama for accessing Llama 3 8B. We analyzed 400 diverse production-grade schemas from Microsoft SQL Server, each varying in table count. No existing benchmark was used as ground truth for this evaluation; instead, our approach relies on analyzing the issues detected in the databases, and manually reviewing each identified issue to assess whether the model's output is correct or not.

5.1 Relational Schema Analysis

Normalization issues were particularly prevalent, as shown in table 1, highlighting the challenges in achieving a balance between efficient data organization and query performance. Over-normalization can lead to excessive joins and slow down queries, while under-normalization can cause data redundancy and inconsistency. In a banking context, for instance, an Orders table might be split into excessively many tables (e.g., separate tables for order details, customer details, product details, shipping details), requiring complex joins for simple order retrieval queries. Conversely, under-normalization might result in a single table storing both customer details and order information, leading to duplication of customer information for each order.

Table 1: Distribution of Model Analysis Issues in Analysised Databases

c

Category	% issues found
Normalization Issues	34%
Naming Conventions	21%
Schema Design Flaws	16%
Missing Primary or Foreign Keys	6%
Entity Duplication	4%

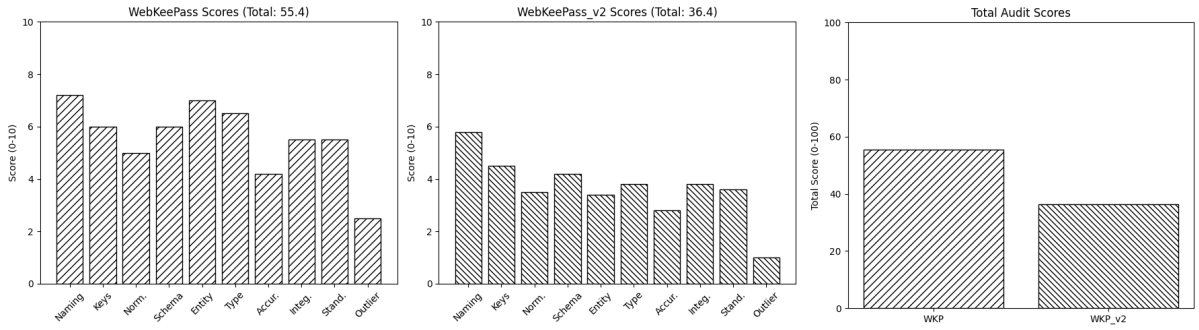

Figure 1: Scoring Methodology. Source: Authors.

Poor naming conventions were consistently observed across several analyzed schemas. Beyond naming, fundamental schema design flaws were frequently present, directly indicating potential inefficiencies, incorrect data representation, or future data integrity risks. These flaws often manifest as poorly designed relationships or the absence of necessary junction tables, leading to direct data redundancy or incorrect associations. For example, in a banking model, a many-to-many relationship between Customers and Accounts might be improperly implemented by duplicating customer information in each account record, rather than using a dedicated junction table (e.g., CustomerAccounts). This design not only wastes storage but also creates high potential for update anomalies and data inconsistencies. Similarly, in an orders model, the critical absence of a foreign key relationship between the Orders table and the Products table means the database cannot enforce that orders are only placed for valid, existing products, potentially leading to orders with references to non-existent items and thus, data integrity breaches at the application level.

5.2 Data Quality Assessment

The data quality analysis component of the evaluation focused on assessing the accuracy, consistency, and reliability of the data stored within the databases. The analysis revealed a prevalence of specific critical issues that can significantly impact data-driven decision-making and operational efficiency. The most common problems identified included data type issues, data integrity violations, and data standardization inconsistencies.

Table 2: Distribution of Data Quality Analysis Issues in Analysised Databases

c

Category	% issues found
Data Type Issues	28%
Data Integrity Issues	18%
Data Standardization	15%
Data Accuracy	8%
Outliers Detection	6%

Data type issues were highly prevalent, indicating that many databases were not utilizing the most appropriate data types for their stored values. In a banking model, this could manifest as storing monetary values as ‘VARCHAR‘ instead of ‘DECIMAL‘, leading to calculation errors and potential sorting issues. In an orders model, storing order dates as ‘VARCHAR‘ instead of ‘DATE‘ or ‘DATETIME‘ could hinder efficient date-based queries and reporting. Data integrity issues were also common, underscoring the importance of enforcing constraints to ensure data accuracy and consistency. Missing constraints, such as primary keys, foreign keys, or unique constraints, can lead to invalid or inconsistent data. In a banking model, a ‘Transactions‘ table might lack a foreign key constraint linking it to the ‘Accounts‘ table, allowing transactions to be recorded for non-existent accounts. In an orders model, a missing unique constraint on order IDs could result in duplicate entries for the same order.

5.3 Effectiveness of Proposed Actions

Following the identification and categorization of relational schema and data quality issues, the effectiveness of the remediation actions proposed by our SLM-based analysis solution was evaluated. The methodology applied involved executing the suggested SQL queries and Python scripts to address the identified problems, followed by a re-run of the analysis solution to quantify the percentage of resolved issues. As depicted in Table 3, the proposed actions demonstrated varying degrees of effectiveness across different categories. The SLM exhibited a relatively high success rate in resolving data type issues and data integrity violations, indicating its proficiency in generating code to enforce constraints and correct data type mismatches. Similarly, it successfully added missing foreign key constraints, ensuring referential integrity.

For addressing missing primary or foreign key constraints, which showed a 78% resolution rate, our solution primarily focused on generating SQL ALTER TABLE scripts. However, challenges arose when dealing with composite keys, where the SLM had to correctly identify and combine multiple columns into a single key. Additionally, complex scenarios involving cascading updates or deletes required the SLM to generate more intricate SQL, which sometimes exceeded its capabilities. In cases where existing data violated the new constraints, the SLM attempted to generate scripts to clean or transform the data, but these were often flagged for manual review due to potential data loss. For naming conventions, our solution

achieved a 85% success rate by generating SQL ALTER TABLE ... RENAME COLUMN scripts.

Table 3: Effectiveness of Proposed Actions: Percentage of Resolved Issues

Category	% Issues Resolved
Naming Conventions	85%
Missing Primary or Foreign Keys	78%
Data Type Issues	75%
Data Integrity Issues	58%
Data Standardization	52%
Outliers Detection	52%
Normalization Issues	45%
Data Accuracy	42%
Schema Design Flaws	38%
Entity Duplication	32%

Normalization issues and schema design flaws, with resolution rates of 45% and 38% respectively, proved to be more complex challenges. The SLM attempted to address these by generating CREATE TABLE, ALTER TABLE, and INSERT INTO scripts, for instance, by proposing new tables and foreign keys to correct denormalization. However, the intricate data dependencies and the need for precise data migration often required manual oversight to prevent inconsistencies or data loss. Similarly, entity duplication (32% resolution) involved the SLM generating queries to identify similar entities based on naming and data types, then suggesting consolidation scripts.

In data quality assessment, our solution achieved a 60% success rate in correcting data type mismatches. Some complex type conversions, such as converting VARCHAR columns containing dates or numerical values, required additional data validation and transformation logic, which the SLM could not always generate reliably. Data integrity violations, such as missing constraints, were addressed with a 58% success rate. The SLM analyzed the schema metadata to identify missing UNIQUE, NOT NULL, or CHECK constraints and generated the corresponding SQL. However, the automated generation of complex CHECK constraints or the handling of existing data that violated the new constraints often required manual intervention. Data standardization, with a 52% success rate, was approached by generating UPDATE scripts to transform data into consistent formats. For example, the SLM attempted to standardize date formats, address formats, or phone number formats using string manipulation techniques. However, semantic inconsistencies, such as variations in abbreviations or domain-specific formatting rules, often required manual review and correction.

Outliers detection and data accuracy improvements, with 52% and 42% resolution rates respectively, involved the use of statistical methods and data imputation techniques. For outliers, the SLM generated SQL queries to identify extreme values based on statistical measures like standard deviation or interquartile range. For data accuracy, the SLM attempted to generate UPDATE scripts to correct errors based on predefined rules or external data sources. The observed variances in remediation effectiveness highlight the strengths and limitations of the SLM-based analysis solution. While the SLM demonstrated significant potential in automating routine maintenance tasks and addressing straightforward data quality issues, complex schema restructuring and subjective tasks still require human oversight or further refinement of the model's capabilities.

5.4 Scalability of the Proposed Solution

The reliance on dynamic prompts with few-shot examples for on-premises SLMs fundamentally limits its scalability for comprehensive relational schema and data quality analysis. This approach quickly hits a critical bottleneck: the model's fixed context window (token limit). As database schemas grow in complexity with numerous tables, intricate relationships, and diverse data quality rules, providing a sufficient number of in-context examples and relevant schema definitions within a single prompt becomes unfeasible. This constraint forces the decomposition of analysis into fragmented, isolated queries, undermining the ability to perform holistic assessments or maintain a consistent understanding across an entire data ecosystem. Furthermore, managing, updating, and dynamically generating thousands of tailored few-shot prompts for every conceivable data quality check across a large enterprise becomes an unsustainable and error-prone engineering challenge, preventing any true scalability or robust performance for production-grade environments.

6 CONCLUSIONS AND FUTURE WORK

This paper explored the significant potential of leveraging Small Language Models (SLMs) for automating the auditing of relational database schemas and data quality within on-premises infrastructure, crucial for compliance-driven environments. Our non-invasive approach generates verification code rather than directly altering data, ensuring a secure analysis process. We've demonstrated SLMs' capabilities in identifying and suggesting solutions for critical issues such as missing foreign keys, normalization anomalies, data integrity violations, and data type inconsistencies. Our evaluation on 400 production schemas from a real-world banking organization revealed a pervasive presence of these issues and underscored how our SLM-based solution can substantially enhance schema quality and address data quality concerns. By empowering data stewards with generated SQL queries and Python scripts, our methodology provides actionable insights, fostering a more robust and reliable data environment essential for effective data governance and the development of trustworthy data-driven and AI solutions.

While our study showcases the promising potential of SLMs in database auditing, several avenues for future work remain vital for broader adoption. A primary direction involves exploring Retrieval-Augmented Generation (RAG) and fine-tuning techniques, and rigorously comparing their efficacy and scalability against our current dynamic prompt-based solution. Another important direction is to assess how the SLM's performance compares with that of human experts and conventional tools for schema and data quality assessment, both in terms of accuracy and cost-effectiveness. Beyond this, we aim to investigate more advanced code generation methods, such as parameterized queries and adaptive scripts, to seamlessly integrate the solution into diverse database environments. Enhancing the explainability of the SLM's suggestions and

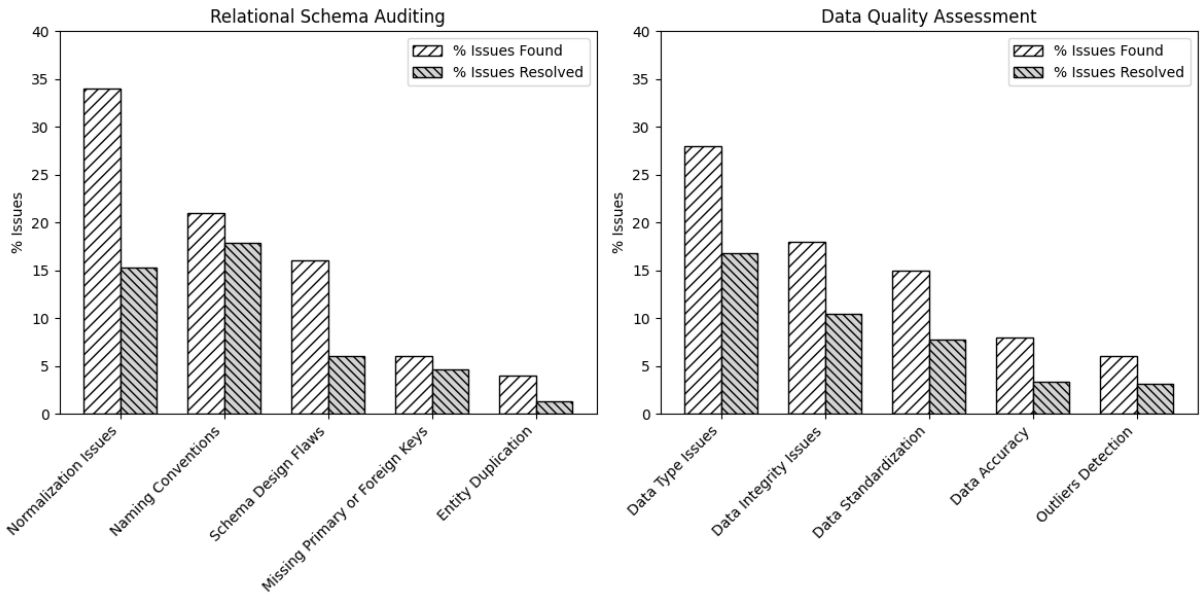

Figure 2: Analysis Results and Issues Resolved. Source: Authors.

building trust in the generated code, through detailed justifications or automated test case generation, is another key objective. Furthermore, incorporating domain-specific knowledge will refine the SLM's understanding of industry-specific data quality requirements. Lastly, optimizing the performance and scalability of our SLM-based solution for analyzing very large and complex databases will ensure its practical applicability in enterprise-level contexts. Addressing these directions will further solidify SLMs' role in advancing robust and reliable data management practices.

REFERENCES

[1] Martin Abadi. 2016. Data management in the age of large-scale machine learning. *Commun. ACM* 59, 1 (2016), 59–68.

[2] Marcelo Arenas, Leopoldo E Bertossi, and Jan Chomicki. 2018. Data exchange: Why and how. *Synthesis Lectures on Data Management* 10, 1 (2018), 1–170.

[3] Fabio Azzalini, Cinzia Cappiello, Chiara Criscuolo, Camilla Sancricca, Letizia Tanca, et al. 2023. Data Quality and Data Ethics: Towards a Trade-off Evaluation.. In *VLDB Workshops*.

[4] Philip A Bernstein, Robbert Van Renesse, et al. 2009. Data management for e-science. *Commun. ACM* 52, 1 (2009), 70–77.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Ben Cao, Tao Yu, Xi Victoria Li, Baishakhi Ray Chang, Wen-tau Yih, and Shiyu Chang. 2021. Dynamo: Dynamic query decomposition for complex text-to-sql. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3250–3263.

[7] Xin Chu, Ihab F Ilyas, and Paolo Papotti. 2016. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment* 9, 12 (2016), 936–947.

[8] Wenfei Fan. 2012. Data quality: From data cleaning to data integration. *Proceedings of the VLDB Endowment* 5, 13 (2012), 1974–1977.

[9] Alon Y Halevy, Zachary G Ives, Igor Tatarinov, and Peter Mork. 2006. Data exchange: Challenges and directions. *Proceedings of the VLDB Endowment* 1, 1 (2006), 668–675.

[10] Wentao Hu, Dongxiang Zhang, Dawei Jiang, Sai Wu, Ke Chen, Kian-Lee Tan, and Gang Chen. 2020. AUDITOR: a system designed for automatic discovery of complex integrity constraints in relational databases. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2697–2700.

[11] Ahmed Karkouch, Mohammed Alenezi, Rami Jaradat, et al. 2018. Data governance: A survey. *Information Systems* 79 (2018), 69–85.

[12] Tim Kraska, Stratos Idreos, Gustavo Alonso, Alon Y Halevy, and Surajit Chaudhuri. 2009. Consistent selection for data cleaning. *Proceedings of the VLDB Endowment* 2, 1 (2009), 1118–1129.

[13] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2002), 233–246.

[14] Raymond Li et al. 2023. Starcoder: code generation with a billion parameters. *arXiv preprint arXiv:2305.06133* (2023).

[15] Eduardo Nascimento, Gustavo Coelho, Lucas Feijó, Yenier Izquierdo, Grettel García, Aiko Oliveira, João Pinheiro, Antony Seabra, Antonio Furtado, Luiz Paes Leme, et al. 2025. On the Construction of Text-to-SQL Tools Based on Large Language Models for Real-World Relational Databases. In *International Conference on Web Information Systems and Technologies*. Springer, 151–167.

[16] João Pinheiro, Wendy Victorio, Eduardo Nascimento, Antony Seabra, Yenier Izquierdo, Grettel García, Gustavo Coelho, Melissa Lemos, Luiz André P Paes Leme, António Furtado, et al. 2023. On the Construction of Database Interfaces Based on Large Language Models.. In *WEBIST*. 373–380.

[17] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. 2018. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment* 11, 12 (2018), 1781–1794.

[18] Rui Shi, Tian Lin, Xiaojun Lin, Binyuan Wang, and William Yang Zhang. 2020. Denotation representation for text-to-sql. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4776–4788.

[19] Yogesh Simmhan, Beth Plale, and Dennis Gannon. 2005. Survey of data provenance in e-science. In *ACM Sigmod Record*, Vol. 34. ACM, 31–36.

[20] Michael Stonebraker, Daniel J Abadi, David J DeWitt, Samuel Madden, Eugene Shah, et al. 2007. C-store: A column-oriented dbms. *Proceedings of the VLDB Endowment* 1, 2 (2007), 963–976.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[22] Steven Euijong Whang and Jae-Gil Lee. 2020. Data collection and quality challenges for deep learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3429–3432.

[23] Xinying Yang, Yuan Zhang, Sheng Wang, Benquan Yu, Feifei Li, Yize Li, and Wenyuan Yan. 2020. LedgerDB: A centralized ledger database for universal audit and verification. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3138–3151.

[24] Tao Yu, Zhang Ruiqiang, Gaurav Yashesh, Xiao Cheng, Ziyu Chang, Shreya Xue, Raghav Aggarwal, Tim Chen, Bhavana Sodhani, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 3211–3221.