Database Modeling Automation from Natural Language Requirements

Júlia O. K. Menezes[1], **Claudio E. C. Campelo**[1]

[1]Systems and Computing Department
Federal University of Campina Grande (UFCG)

`julia.menezes@ccc.ufcg.edu.br, campelo@dsc.ufcg.edu.br`

Abstract. *This paper proposes an approach to support relational database modeling through the automatic generation of Entity-Relationship (ER) diagrams from natural language requirements, leveraging Large Language Models (LLMs) combined with prompt engineering. The method extracts entities, relationships, and attributes from textual descriptions to produce visual ER diagrams. The evaluation was conducted in two phases: first, by testing different LLMs based on structured criteria; and second, by validating the results with students familiar with database modeling. The results indicate that, while challenges remain in handling cardinalities and nullable constraints, the generated diagrams generally align well with the original requirements. These findings reinforce the potential of LLMs to enhance conceptual database modeling.*

1. Introduction

Relational databases are founded on the relational model, in which data is organized into relations—mathematical constructs that are commonly represented as tables [Elmasri and Navathe 2011]. Historically, it has been the most widely used data organization model, due to its ease of maintenance and easy access through query languages such as SQL (Structured Query Language) [Robinson et al. 2024].To design it, one must derive from the previously defined requirements a schema that describes how the data is organized within the system. This process must be carried out carefully, as the schema defines the entire logical structure of information storage.

In this context, diagrams that represent how entities relate to each other are essential tools for building a good schema. [Chen 1976] proposed the Entity-Relationship model as an innovative approach to database design. In it, entities represent objects or concepts in reality, while relationships show the links between these entities, using cardinalates to define their interactions [Magalhães and Heuser 2010].

Despite their visual simplicity and legibility, Entity-Relationship diagrams are only a representation of conceptual modeling, which is, in fact, the most challenging stage of the process. Logical modeling requires the responsible professional to accurately interpret often ambiguous requirements, identifying implicit and explicit details that directly impact the structure of the. Thus, even though the generation of the diagram can be automated, the quality of the model depends fundamentally on the correct abstraction and organization of the data. A well-designed diagram acts as a blueprint for the system, guiding its implementation and supporting the analysis of the information structure [Bagui and Earp 2003].

In this sense, although there have been tools for building ER diagrams for decades, most of them are limited to the graphical representation of the model, requiring the analyst to make all the effort of conceptual and logical design of the model. What still an open challenge is the development of solutions capable of more effectively supporting the stage prior to diagramming: the definition of entities, relationships, and attributes based on a precise interpretation of the system's requirements.

This paper proposes an automated approach to support the conceptual modeling of databases based on natural language textual requirements. To achieve this, we use large-scale language models combined with specific prompt engineering techniques [Sahoo et al. 2024] and visual diagram generation tools.

By integrating these technologies, it becomes possible to automate part of the reasoning involved in transforming textual descriptions into coherent conceptual structures, thereby reducing the analyst's cognitive load, minimizing errors, and accelerating the modeling phase. To validate the approach, we adopted a method that involved evaluating the generated outputs with volunteer students who had previously been familiarized with modeling concepts.

The contributions of this work are:

- a literature review that brings together advances in the field in a summarized way;
- design of a new approach to the automatic generation of ER diagrams.
- a comparative analysis of different LLMs in the database modeling task based on described requirements in natural language
- production of an unprecedented dataset with entity-relationship diagrams and their respective requirements.

2. Related Work

This section presents the main concepts and approaches related to the automatic generation of diagrams, with an emphasis on both classical Natural Language Processing (NLP) methods and emerging approaches based on Large Language Models (LLMs).

2.1. Automatic Generation of ER Diagrams from Natural Language Using Classical NLP Methods

The automatic generation of Entity-Relationship diagrams using classical NLP methods is a topic already explored in the literature. [Btoush and Hammad 2015] propose an approach to the English language that employs preprocessing techniques such as tokenization, *part-of-speech* (POS) tagging, *chunking*, and parsing, combined with heuristics to identify entities, relationships, and attributes. Similarly, [Togatorop et al. 2021] use classical NLP techniques along with a rule-based methodology to identify the elements present in ER diagrams from specifications in the Indonesian language.

In the context of the Vietnamese language, [Thuan et al. 2024] propose a multi-phase methodology that initially uses paraphrasing techniques and heuristic rules to identify diagram elements, and then applies additional heuristics to refine the labeling of these components.

Although these approaches have made significant contributions to the field, they rely on rigid techniques dependent on specific linguistic rules, which limits their scalability and adaptability to other languages or domains. In contrast, the present work proposes a new methodology based on LLMs for the automatic generation of ER diagram components. This approach stands out not only for employing modern natural language processing techniques but also for its flexibility: it is not restricted to a single language and can be applied to any language with operational LLMs. This characteristic significantly expands the reach and applicability of the proposed solution.

2.2. Use of LLMs for Diagram Generation

The use of LLMs for diagram generation constitutes an emerging area of study. Several works have initially demonstrated the generation of general-purpose diagrams from natural language texts. [Zala et al. 2023] propose a *framework* that enables the construction of a new type of diagram, an diagram plan, that can encompass across various domains — both within computing and in external fields — from textual descriptions. Furthermore, [Houndji and Akotenou 2023] suggest an approach for creating UML diagrams using *prompt engineering* techniques. These studies provide a relevant methodological foundation for the present work, which specifically focuses on the application of LLMs in the automatic generation of Entity-Relationship (ER) diagrams. However, although these approaches have shown promising results in various contexts, they have not been directly evaluated for ER diagram generation. This gap indicates that the potential of these methodologies for this specific type of diagram remains unknown, with potential limitations or challenges that have yet to be explored.

Regarding the generation of ER diagrams, the application of LLMs is still in its early stages. Although [Mishra et al. 2024] present significant advances through the use of fine-tuned models, such as Gemma-7B, to generate diagrams in Mermaid [1], their proposal is broad and generic, aimed at creating various types of diagrams. While it includes ER diagram cases, the study does not delve into the specific analysis or evaluation of this type of diagram, nor does it explore the unique challenges of conceptual database modeling based on textual requirements. In contrast, the present work focuses exclusively on ER diagram generation, adopting a carefully developed approach for this purpose — from the creation of a specific dataset to the definition of structured evaluation criteria. Moreover, it proposes a flexible solution, not limited to a single trained model, allowing the use of different LLMs and *prompt engineering* strategies, thus enhancing its adaptability to different contexts and languages.

Finally, [Salem et al. 2024] present a tool that combines LLMs with classic NLP techniques to extract information from textual requirements — or even from images of ER diagrams — and directly convert them into SQL commands. While the proposal is promising by focusing on automating the logical modeling stage, the work lacks clarity in fundamental methodological aspects. There are no precise descriptions of the internal workings of the tool, the criteria used for semantic extraction, or the specific role of LLMs in the process. This lack of information compromises reproducibility and hinders a critical evaluation of the approach's effectiveness. Furthermore, the tool does not include the visual generation of the ER diagram — an essential step for conceptual validation and effective communication between analysts and stakeholders. In this regard, the present work differentiates itself by offering a complete solution for conceptual database modeling, with a clear focus on the automatic and visual generation of ER diagrams from

[1]Mermaid — `https://mermaid.js.org/`

natural language, supported by a robust evaluation methodology, which includes model comparison and empirical validation with users.

3. Methodology

This section describes the dataset created to experimentally evaluate the developed approach. We manually created twenty documents, consisting of pairs of detailed database requirements and their respective diagrams. These documents present interrelated requirements and revolve around a common issue: a real estate agency, varying by the addition or removal of requirements.

The requirements were manually specified to cover a variety of scenarios relevant to ER diagrams, including situations not directly handled by the database and relationships not explicitly mentioned in the texts. One example is the following requirement, which addresses date verification — a responsibility of the application logic, not the database:

> If the interval between the last maintenance and the start of the new contract exceeds nine months, this information must be recorded.

To ensure the quality of the requirements, the GPT-4 language model was used as a revision assistant, with the following *prompt*, focused on identifying ambiguous descriptions and inconsistencies between requirements:

> *Analyze the following requirements and confirm that there are no ambiguities or inconsistencies. If any exist, indicate them and propose possible corrections.*

Additionally, ER diagrams were developed, used as templates based on the presented requirements and following classical modeling standards. These diagrams served as a reference for comparison with the outputs generated by the evaluated tool. This process of data creation and validation was crucial to ensure that the subsequent analysis was based on consistent and relevant information.

3.1. ER Diagram Generator

The proposed approach was implemented as a command-line-based (CLI) and transforms textual requirements contained in TXT-format documents into ER diagrams. The process consists of the following steps (also illustrated in Figure 1): First, a directory containing TXT files with database requirements is provided as input. Each document is then evaluated by an LLM, invoked via an API, which extracts and generates entities, relationships, and attributes into a JSON compatible with ERdot library. These are organized and formatted into a JSON file. Based on this JSON file, the ERdot library[2] converts the content into a .dot file, which is then rendered by the Graphviz tool[3], resulting in the final Entity-Relationship diagram image. The source code, the inputs, the output JSON, the final ER diagrams and the prompts used are publicly available[4].

[2]ERdot — `https://github.com/ehne/ERDot`
[3]Graphviz —`https://graphviz.org/`
[4]Repository — `https://github.com/juliaokmenezes/TXT-to-ERD`

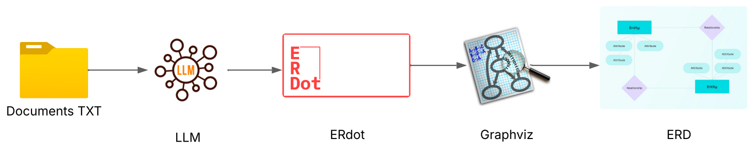

Figure 1. Flow diagram showing the main processes of the approach, from input to output, and their interactions.

3.2. Prompt Engineering

For the proper generation of diagrams, a specific prompt was developed, used to guide the model during the analysis of textual system descriptions and the subsequent creation of formal representations in the form of Entity-Relationship diagrams.

The developed prompt serves multiple purposes: it presents the problem domain to the model, establishes clear guidelines for output formatting, and uses the Few-Shot Prompting technique, providing examples of expected inputs and outputs to reinforce the desired pattern. The examples were organized between the system prompt and the human prompt to reinforce the expected behavior of the model.

Furthermore, the approach incorporates mechanisms for consistency verification and control, such as language standardization and data type selection, aiming to minimize ambiguities during the interpretation of requirements. Additionally, the Chain-of-Thought prompting technique was employed to guide the language model through a structured reasoning process, analogous to conventional practices in database modeling. This strategy contributes to a more accurate identification and classification of entities, relationships, and attributes.

3.3. Evaluation with Different LLMs

To assess the performance of different LLMs in the proposed approach, an experiment was conducted using three models: ChatGPT-4o[5], LLaMA 3-70B-8192[6], and Qwen-2.5-32B[7]. The adopted methodology followed these steps:

- **Oracle Definition**: Initially, reference diagrams were manually created based on previously defined requirements. These diagrams served as oracles for the comparison and evaluation of the diagrams generated by the models.
- **Diagram Generation by the Models**: Each of the three models was used with the diagram generation tool, using the same requirements employed in building the oracle, thereby generating their respective diagrams.

[5]ChatGPT-4o model by OpenAI, released in May 2024. More information available at: `https://openai.com/index/hello-gpt-4o/`

[6]LLaMA 3-70B-8192 model by Meta, accessed via Groq. Details at: `https://groq.com/` and technical information at: `https://ai.meta.com/llama/`

[7]Qwen-2.5-32B model by Alibaba, available via Groq. Details at: `https://groq.com/`. More information at: `https://github.com/QwenLM/Qwen2`

- **Sample Selection and Diagram Generation**: To ensure a controlled and representative evaluation, five requirement documents were randomly selected. Each document was submitted to the aforementioned LLMs to generate Entity-Relationship diagrams. The interaction with the LLMs was conducted via APIs: LLaMA 3 and Qwen were accessed through the Groq platform API, while ChatGPT-4o was accessed via its official API. The use of APIs was necessary due to the high computational cost associated with running large-scale LLMs locally. Each model generated one diagram per requirement document, resulting in a total of 15 diagrams for further analysis.
- **Evaluation Criteria**: The generated diagrams were compared to the oracle diagrams using a scoring system based on the accuracy of the following elements: entities, relationships, and cardinalities. For each of these three aspects, the scoring was as follows: 2 points when all elements were correct; 1 point when there were one or two errors; and 0 points when there were three or more errors. Additionally, a penalty of 1 point was applied if the model generated unnecessary or irrelevant elements.
- **Comparison and Analysis Procedure**: The evaluation was conducted manually, comparing each generated diagram with its corresponding oracle. The analysis aimed to identify recurring error patterns for each model, as well as their ability to accurately represent the initial requirements.

The results were consolidated into a score table per model, allowing a comparative analysis of the performance of the tested LLMs. [8]. From the evaluation, it was observed that the performances of the language models were similar, with the ChatGPT-4o and Qwen-2.5-32B models with 100% accuracy, and the LLaMA 3-70B-8192 with 97% accuracy. Therefore, to continue the methodology, we chose the diagrams generated by ChatGPT-4o for convenience.

3.4. Evaluation by Volunteers

In order to complement the technical analysis of the generated diagrams, an evaluation was conducted with the participation of fifty volunteer students from the Database course at the Federal University of Campina Grande (UFCG), during the last month of the 2024.2 academic term. The participants had already been previously exposed to data modeling concepts in class and had completed both theoretical and practical activities on the subject. During the manual evaluation by students, terms semantically similar (e.g., vehicle, car) were considered correct as long as the meaning remained aligned with the original requirement. This criterion was discussed in class during the feedback session to ensure consistency between evaluations, meaning that all groups applied the same interpretation standards when assessing the diagrams. For example, semantically equivalent terms (e.g., vehicle and car) were accepted as correct if their meaning aligned with the original requirement.

To ensure diversity of perspectives and greater robustness in the evaluation, the diagrams were distributed among groups of three to five students, with up to three groups redundantly evaluating the same diagram. Each group performed the evaluation on the basis of a structured questionnaire composed of the following specific analysis questions.

[8]Quantitative analysis results evaluation table — `https://acesse.one/dOkED`

- **Expected entities: does the diagram include all entities described in the requirements?**
  - Contains all entities.
  - Missing 1 or 2 entities.
  - Missing more than 3 entities.
- **Unexpected entities: does the diagram include unexpected, inadequate, or unnecessary entities?**
  - No unexpected entities.
  - Includes 1 or 2 unexpected entities.
  - Includes more than 3 unexpected entities.
- **Expected relationships: are all relationships specified in the requirements present?**
  - All expected relationships are present.
  - Missing 1 or 2 relationships.
  - Missing 3 or more relationships.
- **Unexpected relationships: are there any relationships in the diagram that are unexpected, inadequate, or unnecessary?**
  - No unexpected relationships.
  - Includes 1 or 2 unexpected relationships.
  - Includes more than 3 unexpected relationships.
- **Cardinality: are there any errors in the cardinalities of the relationships?**
  - No relationships with incorrect cardinalities.
  - 1 or 2 relationships with incorrect cardinalities.
  - 3 or more relationships with incorrect cardinalities.
- **Entity attributes: are the entity attributes consistent with the requirements?**
  - All entities contain the expected attributes.
  - 1 or 2 entities missing attributes.
  - 3 or more entities missing attributes.
- **Relationship attributes: are the relationship attributes consistent with the requirements?**
  - All relationships contain the expected attributes.
  - 1 or 2 relationships missing attributes.
  - 3 or more relationships missing attributes.
- **Describe the most critical issue found in the diagram, if any.**
- **Describe the second most critical issue found in the diagram, if any.**
- **If you did not find issues in the diagram but have suggestions for improvement, please list them.**
- **Answer this question only if you found no issues in all previous responses:** In the requirements descriptions, if you noticed any inconsistencies, ambiguities, or unclear descriptions, indicate which requirement(s) and the respective issue(s).

After collecting the forms, the points of greatest divergence among the groups were highlighted and discussed in class under the guidance of the instructor. This step aimed to understand whether the identified issues resulted from flaws in the automatic diagram generation or from students' difficulties in interpreting the presented concepts. At the end of the discussion, students were invited to review and, if necessary, revise their previous evaluations.

All code, prompts, and data (requirements specifications) were originally produced in English, although the proposed solution is potentially generalizable to other languages known by the LLM. Furthermore, since the participants in the experiments were native Portuguese speakers and their English proficiency could not be guaranteed, we translated the requirements provided to all participants to ensure that the results were not affected by misunderstandings.

4. Results

Based on the students' evaluations, it was possible to carry out an analysis using Google Colaboratory[9] with Python[10] to process and visualize the data.

4.1. Quantitative Evaluation of the Diagrams

Initially, the goal was to observe students' assessments regarding the correctness of the diagrams. In Figures 2, 3, 4, 5, 6, it is possible to observe the distribution of the student's responses to the diagrams.

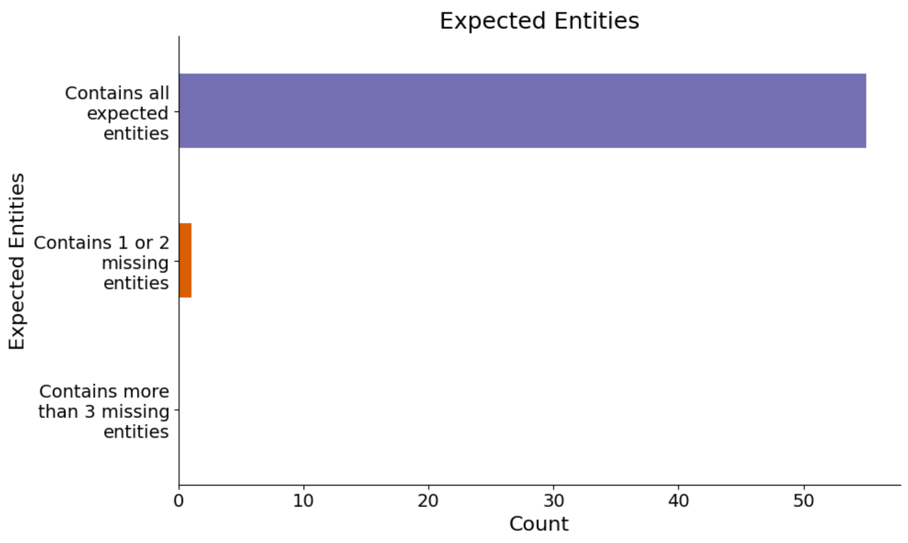

Figure 2. Frequency of Responses in the Evaluation of Entities

In general, the diagrams were well evaluated in terms of the representation of entities, relationships, and cardinalities. However, a recurring failure was identified in the generation of entity attributes. The qualitative analysis revealed that this failure is associated with a specific requirement, present in part of the documents, indicating a punctual issue rather than a structural one.

4.2. Qualitative Analysis of the Evaluations

When examining the qualitative responses, the following critical points stood out:

- **Difficulties in defining cardinalities:** The LLM showed difficulties in correctly determining cardinalities when they were not explicitly stated in the requirements.

[9]`https://colab.research.google.com/`
[10]`https://www.python.org/`

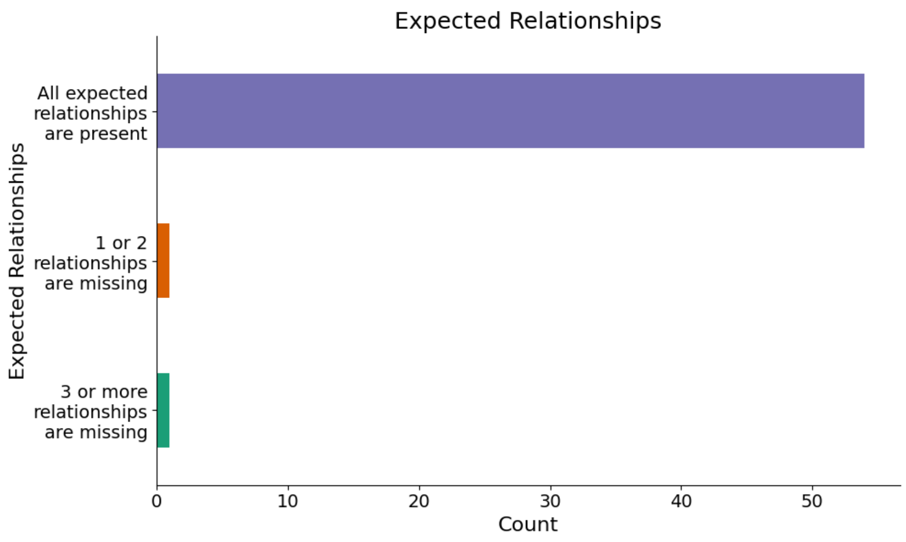

Figure 3. Frequency of Responses in the Evaluation of Relationships

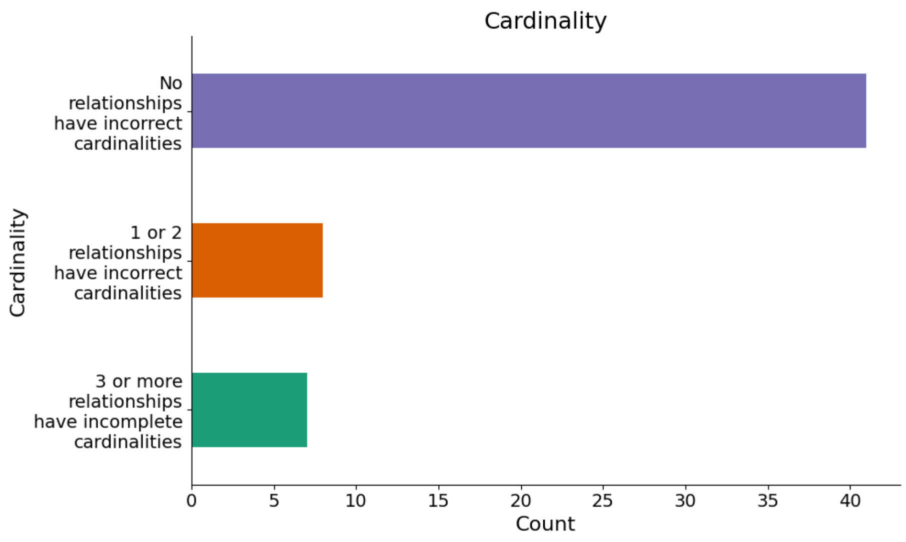

Figure 4. Frequency of Responses in the Evaluation of Cardinalities

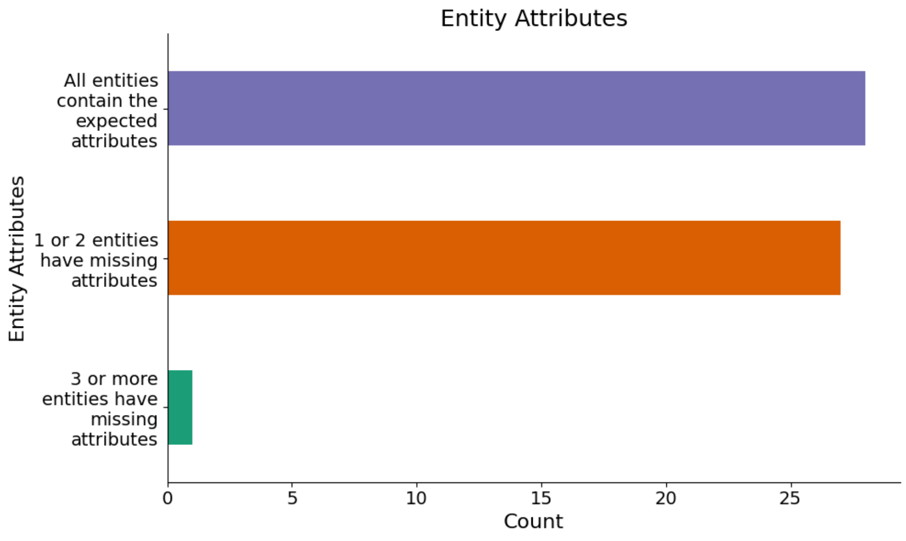

Figure 5. Frequency of Responses in the Evaluation of Entity Attributes

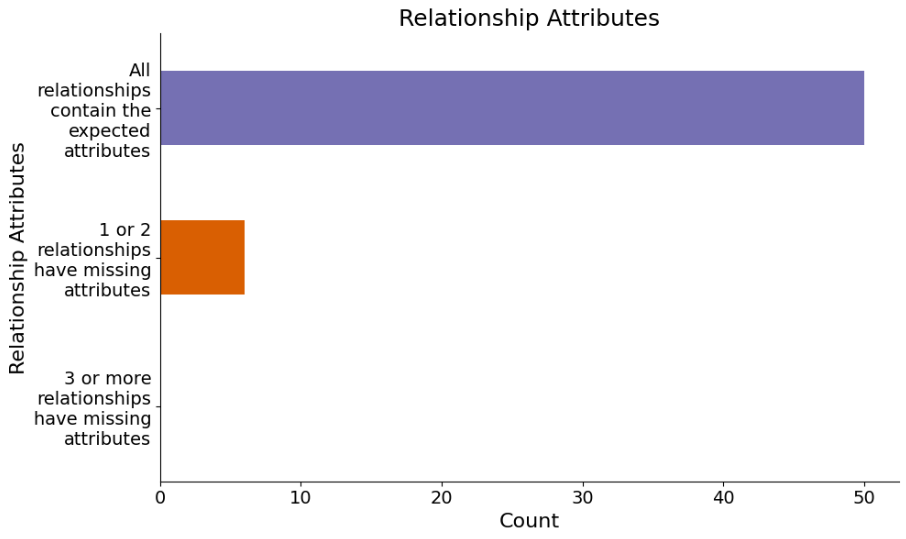

Figure 6. Frequency of Responses in the Evaluation of Relationship Attributes

The responses that received lower scores in this category often pointed out errors such as inversion of values or omission of cardinalities. This pattern reinforces the need for extra attention when interpreting implicit relationships.

- **Absence of the NOT NULL constraint:** Although some requirements did not directly address minimum cardinalities, the implicit presence of mandatory fields (for example, through the requirement of a NOT NULL flag) was overlooked in several situations. Criticisms related to the absence of this constraint correlate with lower quantitative evaluations, indicating that these failures negatively impact the overall perception of diagram quality. It is notable that Figure 5 highlights the evaluators' dissatisfaction with this point.

These inconsistencies highlight the limitations of the LLM in interpreting semantic nuances of the requirements, especially when they involve implicit inferences — such as the determination of unspecified cardinalities or the application of mandatory constraints. Although the model demonstrates competence in generating the general structure of diagrams, it still faces challenges when dealing with more complex contextual interpretations that require greater abstraction and domain understanding.

4.3. Incidence of Hallucinations

Regarding hallucinations — that is, the generation of unnecessary entities and relationships — it was observed that only 7.14% of the responses contained irrelevant relationships and 1.70% indicated improper attributes. This low percentage suggests that the task of identifying entities and relationships is more guided by explicit structural patterns in the requirements, which reduces the chance of the model extrapolating beyond the scope. Therefore, when the requirements are well-defined, the LLM tends to remain adherent to the problem domain, minimizing the generation of irrelevant elements.

4.4. Cases with Higher Error Incidence

When evaluating the diagrams with the highest incidence of errors, it is noticeable that the longer requirements and those composed of multiple steps present additional challenges for language models. For example:

> *"If the contract value exceeds 50% of the tenant's declared salary, a guarantor must sign the contract jointly. The guarantor can sign more than one contract."*
>
> *"Each payment must record the accounts involved (payer and receiver), in addition to the payment amount and date."*

In the former example, the model often generates only the existence of a guarantor, omitting the possibility of a single guarantor signing multiple contracts. This leads to the incorrect creation of a 1:1 relationship, when the correct relationship should be 1:N. In the latter example, a common failure was the omission of specific fields to represent the accounts involved.

When working with ER modeling, some situations can be represented in structurally different but semantically equivalent ways, for example, by transforming an M:N relationship into two 1:N relationships using an associative entity. During manual evaluation, such structurally analogous representations were accepted as long as they preserved the intended semantics of the original requirement.

These types of textual requirements demand that the language model go beyond simple entity and relationship extraction. They require a degree of abstraction and reasoning that surpasses a purely literal interpretation of the input. This underscores the importance of atomicity in the formulation of modeling statements, as a critical factor for enabling more accurate and semantically coherent ER diagram generation.

5. Conclusion and Future Work

The approach presented in this work proved to be promising by automating part of the conceptual modeling process of relational databases through the generation of ER diagrams using LLMs. The method demonstrates a good ability to correctly identify entities, relationships, and cardinalities, especially when the requirements were well defined.

The quantitative and qualitative evaluations, carried out with the participation of students, reinforced the effectiveness of the approach. A low incidence of hallucinations was observed, but also relevant limitations in the interpretation of implicit constraints, such as unspecified cardinalities and the lack of mandatory attributes. These challenges highlight difficulties that LLMs still face in dealing with semantic nuances and more complex contextual inferences. We plan to explore the potential of alternative prompting techniques to address these issues.

Among the contributions of this work, we highlight the creation of a novel dataset with textual requirements and their respective diagrams, in addition to the comparative analysis between different LLM models for this task. As future work, we propose improving the generation of attributes and cardinalities, integrating with more accessible interfaces, and adopting hybrid approaches that combine LLMs with heuristics to overcome the observed limitations.

References

Bagui, S. S. and Earp, R. W. (2003). *Database Design Using Entity-Relationship Diagrams*. Auerbach Publications.

Btoush, E. S. and Hammad, M. M. (2015). Generating er diagrams from requirement specifications based on natural language processing. *International Journal of Database Theory and Application*, 8(2):61–70.

Chen, P. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

Elmasri, R. and Navathe, S. B. (2011). *Fundamentals of Database Systems*. Addison-Wesley, 6th edition.

Houndji, V. R. and Akotenou, G. (2023). Umldesigner: An automatic uml diagram design tool. In *Proceedings of the International Conference on Deep Learning Theory and Applications (DeLTA 2023)*, volume 1875 of *Communications in Computer and Information Science*, pages 309–321. Springer, Cham.

Magalhães, M. and Heuser, C. A. (2010). *Projeto de Banco de Dados*. Editora Érica.

Mishra, M., Sheikh, S., and Tonpe, T. (2024). Fine-tuning language models for enhanced diagram generation: A deep learning approach. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, 12(IV):2819–2822.

Robinson, J., Ranjan, R., Hu, W., Huang, K., Han, J., Dobles, A., Fey, M., Lenssen, J. E., Yuan, Y., Zhang, Z., He, X., and Leskovec, J. (2024). Relbench: A benchmark for deep learning on relational databases. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, pages 21330–21341.

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *ArXiv*, abs/2401.10238. Department of Computer Science and Engineering, IIT Patna; Stanford University; Amazon AI.

Salem, N., Al-Tarawneh, K., Hudaib, A., Salem, H., Tareef, A., Salloum, H., and Mazzara, M. (2024). Generating database schema from requirement specification based on natural language processing and large language model. *Computer Research and Modeling*, 16(7):1703–1713. https://www.researchgate.net/publication/387457163.

Thuan, N., Tran, N., and Quyen, T. (2024). Generating erd and ddl scripts from vietnamese natural language text by using a multi-phase approach. In *Proceedings of the International Conference on Information Systems Design and Support*.

Togatorop, P. R., Simanjuntak, R. P., Manurung, S. B., and Silalahi, M. C. (2021). Generating entity relationship diagram from requirement specification using natural language processing for indonesian language. In *J-Icon: Jurnal Komputer dan Informatika*, volume 9, pages 196–206. Institut Teknologi Del.

Zala, A., Lin, H., Cho, J., and Bansal, M. (2023). Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. https://arxiv.org/abs/2310.12128.