Towards an LLM-based Tool for Automated Database Design

Predrag Divljan[1], Drazen Brdjanin[2]

[1]*Communications Regulatory Agency, Mehmeda Spahe 1, 71000 Sarajevo, Bosnia and Herzegovina*
[2]*University of Banja Luka, Faculty of Electrical Engineering, Patre 5, 78000 Banja Luka, Bosnia and Herzegovina*

Abstract
Large Language Models (LLMs) are revolutionizing many fields, offering great potential for data modeling as well. This paper presents an early prototype of an LLM-based tool for automated (conceptual) database design. The implemented online web-based tool applies a zero-shot LLM prompting approach, where the users may select one of three different state-of-the-art LLMs. The implemented tool enables automatic synthesis of the target conceptual database model represented by a PlantUML model. The preliminary evaluation results imply very high effectiveness of LLM-based CDM synthesis and significant potential for further applications.

Keywords
class diagram, conceptual database model, LLM, PlantUML

1. Introduction

Database design has long been recognized as a very important research area. The common design process begins with a conceptual design phase, resulting in a CDM (*Conceptual Database Model*). This phase is inherently complex and typically requires multiple iterations to finalize the target CDM. Unlike the subsequent design steps that are usually just straightforward automated transformations of the CDM, conceptual design is typically performed manually, and its automation is very desirable.

Although the idea of automating CDM design dates back to the early 1980s [1], there is still no tool able to automatically generate a complete and correct CDM, regardless of the input used. The existing approaches primarily take *textual specifications* (cf. [2]), although there are some alternative approaches taking *models* (e.g. [3]), or even *speech* (e.g. [4]) as input for automation.

The emergence of LLMs (such as ChatGPT) has brought very promising opportunities to enhance CDM design [5]. Being trained on extensive and diverse datasets, LLMs are very successful in tasks such as generating meaningful context-aware text or generating executable code. Their ability to process and interpret intricate NL (*Natural Language*) inputs makes them very suitable for data modeling, where the transformation of textual descriptions into structured output is essential.

Although LLMs are capable of automatically generating structured outputs, such as JSON representation of data models (e.g. [6]), their usage in automated database design has not yet reached the level of a tool able to generate a complete and correct target database schema based on user requirements or textual descriptions. This fact motivates our research aimed at developing an LLM-based tool for automated database design, which will: (1) be able to automatically derive a CDM from a textual specification, (2) enable forward engineering of the target database based on the automatically generated CDM. In this paper we present an early online web-based prototype of the target tool, which is currently able to perform the first step – automatic derivation of a CDM represented by a PlantUML class diagram.

The paper is structured as follows. After this introductory section, the second section provides a brief overview of the related work. The third section presents the approach and the implemented tool, while the fourth section illustrates its usage. The final section concludes the paper and provides future work intentions.

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France
pdivljan@gmail.com (P. Divljan); drazen.brdjanin@etf.unibl.org (D. Brdjanin)
0009-0004-7010-163X (P. Divljan); 0000-0002-0212-6569 (D. Brdjanin)

2. Related work

The notion of automatic CDM derivation from textual specifications dates back to the early 1980s, when Chen proposed a set of heuristic rules to translate unstructured NL sentences (English) into an E-R diagram [1], followed by a lot of research resulting in different approaches and numerous tools [2].

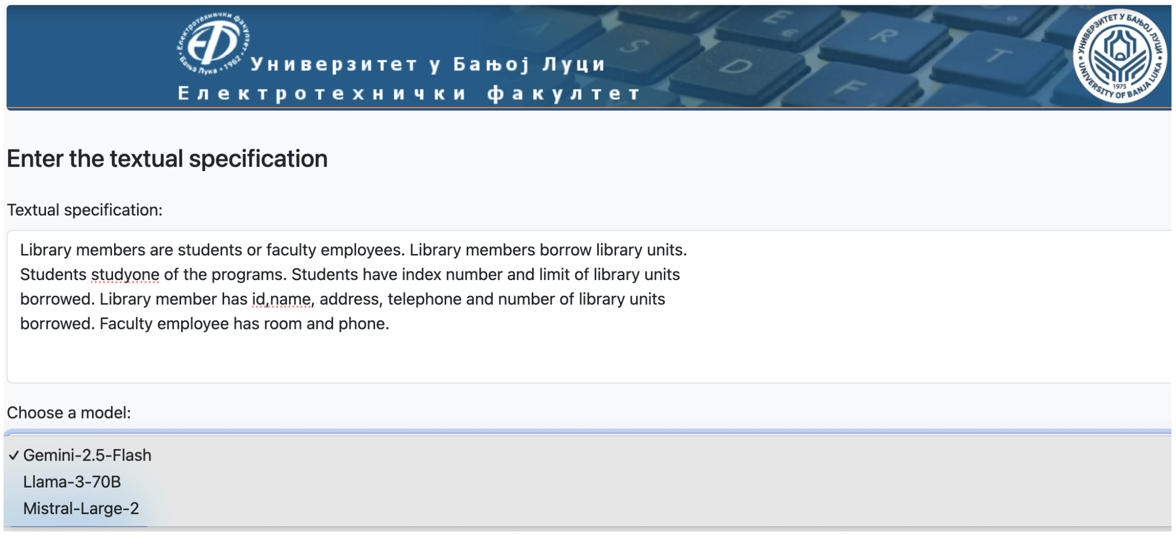
The existing approaches are mainly *linguistics-based* – they apply NLP (*Natural Language Processing*) techniques to derive CDMs from NL text. The existing tools (e.g., ER-Converter [7] and LIDA [8]) typically support a single source NL and do not provide multilingual support. Only TexToData [9] enables automatic CDM derivation from textual specifications in different source NLs.

Recently, the employment of LLMs in data modeling tasks has been explored. The majority of papers (e.g., [6, 10, 11, 12]) just present results of experimental studies aimed at assessing the capability of LLMs for automated derivation of data models from textual specifications. Only a very small number of them (e.g., [13]) present an implemented LLM-based tool for automated data model design, but the generated models do not represent CDMs. The generated data models are typically represented by PlantUML class diagrams (e.g., in [6, 12, 13]), although some alternative representations are also used, such as JSON in [6] or EBNF in [10].

To the best of our knowledge, there is still no LLM-based tool for automatic CDM design, and our tool constitutes the first online LLM-based CDM generator.

3. LLM-based CDM generator

The LLM-based CDM generator is implemented as an SPA (*Single-page Application*) Java Spring Boot web application.[1] It enables automatic creation of PlantUML scripts and diagrams based on input textual specification and the user-selected LLM, as illustrated by the screenshot shown in Fig. 1. It is possible to choose between three LLMs, which are currently employed in the system: Gemini-2.5-Flash, Llama-3-70b, and Mistral-Large-2. These LLMs were selected for their architectural diversity, high effectiveness, and high efficiency scores in automatic CDM generation.[2] Besides the employed LLMs, which are completely free of charge, additional LLMs can be easily employed.



Enter the textual specification

Textual specification:

Library members are students or faculty employees. Library members borrow library units. Students studyone of the programs. Students have index number and limit of library units borrowed. Library member has id,name, address, telephone and number of library units borrowed. Faculty employee has room and phone.

Choose a model:

✓ Gemini-2.5-Flash
Llama-3-70B
Mistral-Large-2

Figure 1: Screenshot of the implemented LLM-based CDM generator in action

[1]Available on: https://database-ai.onrender.com/
Source code available on: https://github.com/pdivljan81/database_AI

[2]We selected these three LLMs after a preliminary assessment of 16 state-of-the-art LLMs. Due to the paper length limitation, we are not able to provide all assessment results, but give only a list of the assessed LLMs: GPT-4, GPT-4o, GPT-4o-Mini, GPT-O1-Preview, GPT-O1-Mini, Gemini-2.5-Pro, Gemini-2.5-Flash, Claude-3.5-Sonnet, Mistral-Large-2, Codestral, Mixtral-8x22B, Llama-3.1-405B-Instr, Llama-3.2-90B-Vision, Llama-3.1-Nemotron, Qwen2.5-72B-Instr, WizardLM-2-8x22B.

The entire process of CDM design, from the input textual specification to the target PlantUML class diagram, is illustrated by the sequence diagram shown in Fig. 2.

Based on the input parameters (textual specification and selected LLM) and a carefully tailored LLM prompt (not visible to the users), the CDM generator creates an API request and forwards it to the corresponding API service[3] of the selected LLM.

Special attention was paid to tailoring the LLM prompt, whereby a *zero-shot prompting*[4] approach is applied. The rules specified for the LLM prompt when generating PlantUML models serve to ensure precise, consistent, and strict adherence to the input description without additional interpretations or assumptions. The selected LLM must use only the information explicitly given in the textual specification, and it is not allowed to introduce prior knowledge, extensions, or typical practices that are not clearly stated. Each class, including subclasses, must contain at least one attribute of its own, and the generation of empty classes or classes without attributes is strictly prohibited. For classes that do not inherit a primary key (PK) operation, it is mandatory to generate exactly one PK operation with the same attribute as the distinct attribute within the class. All other attributes are also explicitly stated. All relationships between classes are displayed only as inheritances or associations with possibly marked end multiplicities and association name, only if they are explicitly defined. It is never allowed to use associations to represent inheritance. Enumerations may be used only when the description clearly and completely states all allowed values. In all other cases, entity types must be represented as concrete classes.[5]

In response, the API service of the selected LLM returns the generated PlantUML script, which is automatically post-processed by the CDM generator if needed. Post-processing is the automatic correction, adjustment, or filtering of generated results after their initial creation, in order to ensure correctness, consistency, or usability according to defined additional rules in the code. Then, the PlantUML script is sent to the external online PlantUML service[6] that generates and returns the corresponding class diagram. Finally, the script and the diagram are shown in the browser.[7]

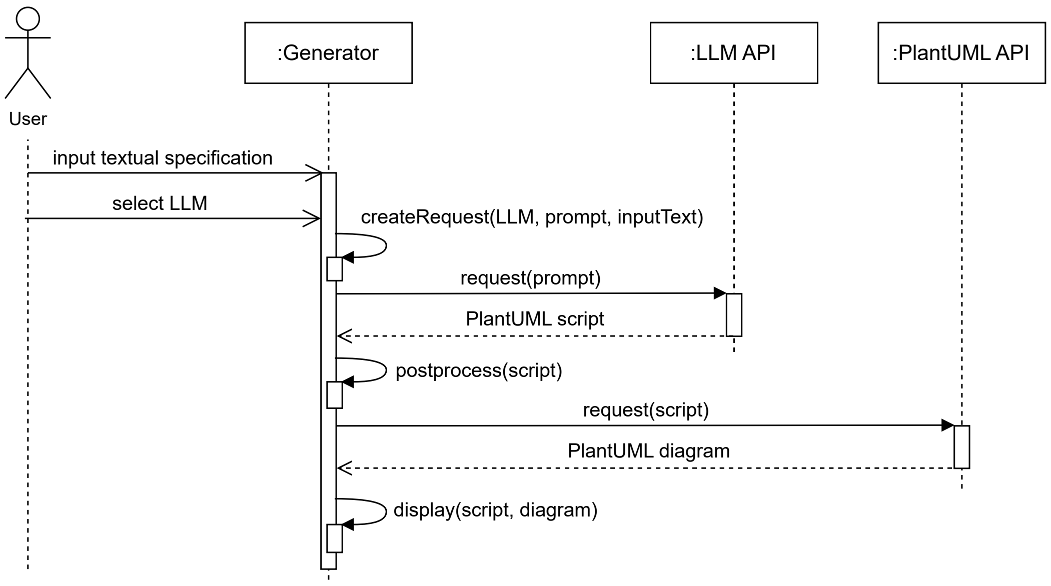


Figure 2: High-level interaction overview in LLM-based CDM design

[3]API service for Gemini-2.5-Flash: https://generativelanguage.googleapis.com/v1/models/gemini-2.5-flash
API service for Llama-3-70b: https://api.groq.com/openai/v1/chat/completions
API service for Mistral-Large-2: https://api.mistral.ai/v1/chat/completions

[4]*Zero-shot prompting* means generating output based solely on an instruction, without any examples (cf. [14]).

[5]The complete prompt can be found at: https://github.com/pdivljan81/database_AI/blob/main/prompt.txt.

[6]https://www.plantuml.com/plantuml/uml/

[7]At the moment, users can only import the resulting PlantUML script into other modeling tools to proceed with forward engineering of the target database, so we intend to integrate the generator into the TexToData tool [9], in order to obtain a complete LLM-based database design tool.

4. Illustrative examples of automatic LLM-based CDM design

This section presents two illustrative examples (Mistral-Large-2 is used in both examples) of automatic CDM design by using the implemented tool.

The first example (Fig. 3) provides a simple textual specification in English, and the corresponding CDM automatically generated by the implemented tool (the corresponding PlantUML script is omitted due to the paper length limitation). To illustrate multilingualism, particularly the ability to derive CDM automatically from languages with very complex morphology (such as Slavic and some other languages), we provide another example (Fig. 4), where we use an equivalent specification in Serbian.

Although the presented examples are quite simple, they are very illustrative and show that the proposed approach and implemented tool enable very effective automatic CDM derivation from textual specifications represented in different NLs, even from NLs with very complex morphology.

Library members are students or faculty employees. Library members borrow library units. Students have index number and limit of library units borrowed. Library member has id, name, address, telephone and number of library units borrowed. Faculty employee has room and phone. Library unit has unique tag and available information. Library units are magazines or books. Library unit has name, year and author. Magazines have number. Magazines and books have ISSN. Publisher has name and residence. Publisher publishes library units.

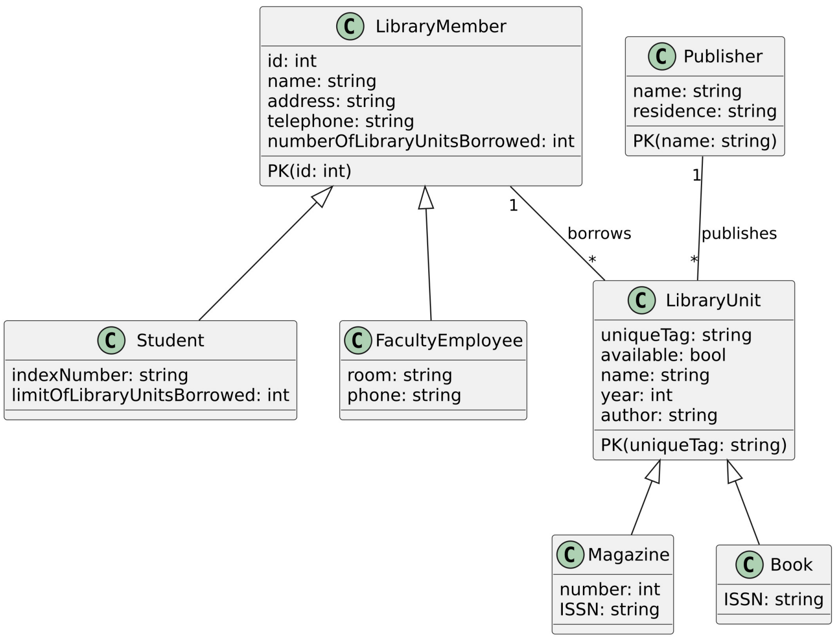


Figure 3: Sample source text in English (left) and corresponding automatically generated CDMs (right)

Članovi biblioteke su studenti ili zaposleni na fakultetu. Članovi biblioteke posuđuju bibliotečke jedinice. Studenti imaju broj indeksa i limit za broj posuđenih bibliotečkih jedinica. Članovi biblioteke imaju id, ime, adresu, broj telefona i broj posuđenih bibliotečkih jedinica. Zaposledni na fakultetu imaju broj sobe i broj telefona. Bibliotečka jedinica ima jedinstvenu oznaku i informaciju o raspoloživosti. Bibliotečke jedinice su časopisi ili knjige. Bibliotečka jedinica ima naziv, godinu izdanja i autora. Časopisi imaju broj. Časopisi i knjige imaju ISSN. Izdavač ima naziv i sjedište. Izdavač objavljuje bibliotečke jedinice.

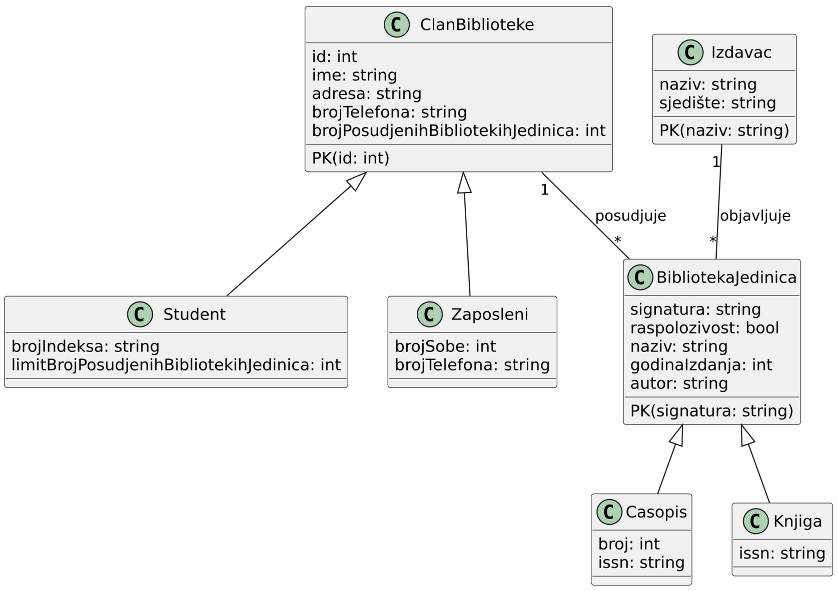


Figure 4: Sample source text in Serbian (left) and corresponding automatically generated CDMs (right)

5. Conclusion

In this paper, we introduced the first online web-based automatic LLM-based CDM generator. The first results show that the LLMs significantly contribute to the automation of CDM design, enabling very efficient and effective CDM derivation from different NLs.

Our future work will focus on further improvements and extensive evaluation of the entire approach, with the main objective to develop and integrate the LLM-based CDM generator into TexToData [9] in order to obtain a complete online LLM-based tool for automated database design.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. Further, the authors used Mistral-Large-2 to generate the PlantUML scripts that are further visualized by the diagrams shown in figures 3 and 4. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Chen, English sentence structure and entity-relationship diagrams, Information Sciences 29 (1983) 127–149.

[2] I.-Y. Song, Y. Zhu, H. Ceong, O. Thonggoom, Methodologies for semi-automated conceptual data modeling from requirements, in: Johannesson, P., et al. (Eds.), Conceptual Modeling, Springer International Publishing, 2015, pp. 18–31.

[3] D. Brdjanin, A. Vukotic, D. Banjac, G. Banjac, S. Maric, Automatic derivation of the initial conceptual database model from a set of business process models, Computer Science and Information Systems 19 (2022) 455–493.

[4] D. Brdjanin, G. Banjac, D. Keserovic, N. Babic, N. Golubovic, Combining speech processing and text processing in conceptual database design, Telfor Journal 16 (2024) 8–13.

[5] V. C. Storey, O. Pastor, G. Guizzardi, S. W. Liddle, W. Maaß, J. Parsons, J. Ralyté, M. Y. Santos, Large language models for conceptual modeling: Assessment and application potential, Data & Knowledge Engineering 160 (2025) 102480.

[6] H.-G. Fill, P. Fettke, J. Köpke, Conceptual modeling and large language models: Impressions from first experiments with chatgpt, Enterprise Modelling and Inf. Sys. Architectures 18 (2023) 1–15.

[7] N. Omar, P. Hanna, P. McKevitt, Heuristics-based entity-relationship modelling through natural language processing, in: Proc. of AICS 2004, 2004, pp. 302–313.

[8] S. Overmyer, L. Benoit, R. Owen, Conceptual modeling through linguistic analysis using LIDA, in: Proc. of ICSE 2001, IEEE, 2001, pp. 401–410.

[9] D. Banjac, M. Matic, N. Cvijanovic, D. Brdjanin, G. Banjac, D. Stojisavljevic, Employing multiple online translation services in a multilingual database design tool, in: J. Tekli, et al. (Eds.), New Trends in Database and Information Systems, Springer Nature Switzerland, 2025, pp. 238–249.

[10] K. Chen, Y. Yang, B. Chen, J. A. Hernández López, G. Mussbacher, D. Varró, Automated domain modeling with large language models: A comparative study, in: MODELS 2023, 2023, pp. 162–172.

[11] I. Reinhartz-Berger, S. J. Ali, D. Bork, Leveraging LLMs for domain modeling: The impact of granularity and strategy on quality, in: J. Krogstie, et al. (Eds.), Advanced Information Systems Engineering, Springer Nature Switzerland, 2025, pp. 3–19.

[12] J. Cámara, J. Troya, L. Burgueño, A. Vallecillo, On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml, Software and Systems Modeling 22 (2023) 1–13.

[13] F. Härer, Conceptual model interpreter for large language models, in: ER Forum 2023, volume 3618, CEUR Workshop Proceedings, 2023.

[14] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, ACM Comput. Surv. 55 (2023) 1–35.