ARTICLE 11

Influence of AI on Database Modeling and Analysis

Nenad Jukić
Loyola University Chicago

Benjamin Arnold
Mintel Group Ltd.

Sippo Rossi
Hanken University

Svetlozar Nestorov
Loyola University Chicago

Generative AI is impacting many areas of information systems and computer science, and database modeling and use is no exception to this trend. As in most cases with the use of generative AI, one of the main questions is how to engage in a balanced use that enhances and improves productivity without sacrificing the quality of work. In this article, we will illustrate several issues stemming from this question as it pertains to the modeling and use of databases.

First, we will demonstrate through three scenarios how generative AI can result in both desirable and undesirable outcomes in the database modeling phase. With a similar approach, we then discuss database analysis by providing examples of correct and incorrect AI-generated SQL code. Next, we will, through examples, illustrate appropriate and inappropriate ways for a student to use AI when studying databases. Lastly, we conclude the article with a discussion about the influence of generative AI on how practitioners work with databases and what needs to be taken into consideration in professional use of generative AI.

Generative AI and Database Modeling

First, let us consider the use of generative AI in database design. For discussion purposes, consider the following simplified scenario. A database designer wants to create an ER (Entity Relationship) diagram for a database that will store data about employees and projects in Company XYZ. Traditionally, this involves collecting the requirements for entities Employee and Project, including the requirements for their attributes and any relationships between them. Using generative AI, it is possible to create an ER diagram that contains some version of entities Employee and Project and a version of a relationship between them, based on the following prompt: "Create an entity relationship diagram about employees and projects." This is illustrated by Figure 1.

We will now consider several possible scenarios stemming from this example.

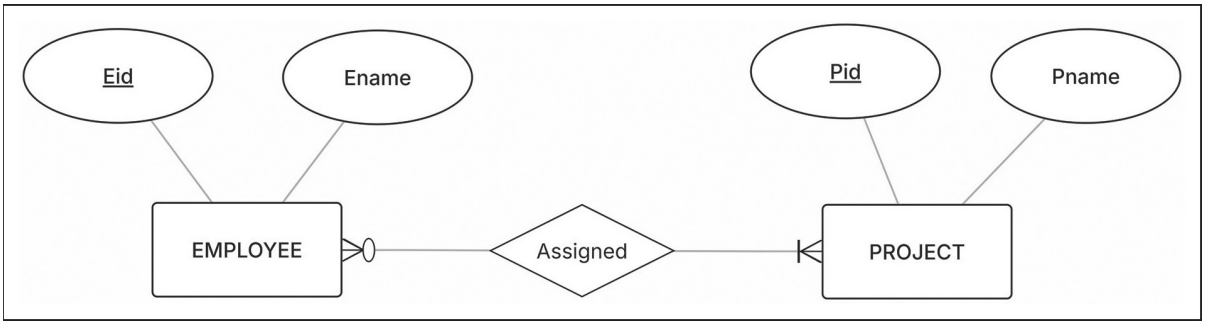

Figure 1 AI-generated ER diagram for Company XYZ.

Scenario 1

The ER diagram in Figure 1 matches perfectly the employees and projects set up in Company XYZ. The ER diagram can now be converted to the corresponding relational schema and implemented as a database. Scenario 1 is an example of successfully using generative AI in the database design process.

Scenario 2

The ER diagram in Figure 1 does not match the actual employees and projects set up in Company XYZ. Contrary to what is shown in Figure 1, for each employee, Company XYZ tracks employee ID, employee name, and employee birthdate; and for each project, Company XYZ tracks project ID, project name, and project description. Moreover, in Company XYZ, each employee is assigned to exactly one project, and each project has between one and many employees assigned to it. Also, each project has one employee who supervises the project, and each employee can supervise one project or none. The database designer in charge of creating and approving this diagram is aware of the discrepancies between what was generated and how the diagram should look like (missing one attribute in entity PROJECT and one attribute in entity EMPLOYEE, incorrect cardinalities in the Assigned relationship, missing relationship Supervises), and he or she can now change the ER diagram, accordingly, as shown in Figure 2.

As the database designer is fully aware of all the correct details (i.e., requirements) and he or she used that knowledge to modify and improve the generated ER diagram, Scenario 2 is another example of successfully using generative AI to aid in the database design process.

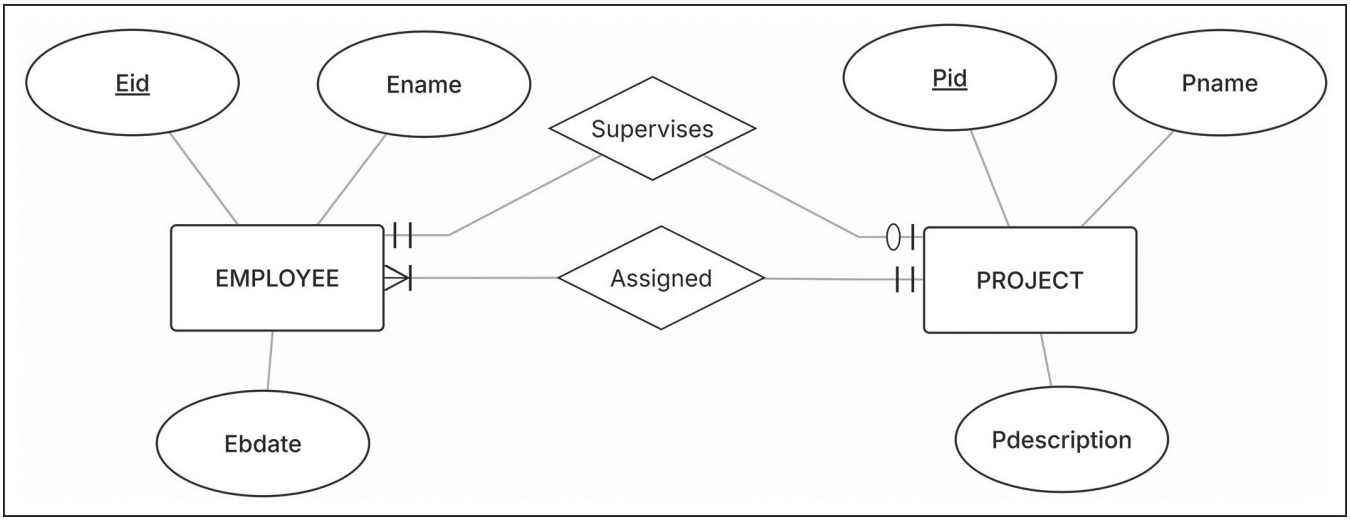

Figure 2 Modified AI-generated ER diagram for Company XYZ.

Scenario 3

The ER diagram in Figure 1 does not match the actual employees and projects set up in Company XYZ. Contrary to what is shown in Figure 1, for each employee, Company XYZ tracks employee ID, employee name, and employee birthdate; and for each project, Company XYZ tracks project ID, project name, and project description. Moreover, in Company XYZ, each employee is assigned to exactly one project, and each project has between one and many employees assigned to it. Also, each project has one employee who supervises the project, and each employee can supervise none or one project.

In contrast to the previous scenario, this time the database designer in charge of creating and approving this diagram is not aware of the discrepancies between what was generated and how the diagram should look. The designer has not been provided with full insight into the company details, and yet he or she is tasked with creating the ER diagram and the subsequent database. Faced with deadlines and the need to produce something, and in the absence of fully worked-out requirements, the designer decides to proceed with the ER diagram from Figure 1 and creates an incorrect database based on it. This, of course, is not an example of successfully using generative AI in the database design process.

The purpose of these three scenarios is to illustrate both the capabilities and limitations of using generative AI in the database design process. There are other possible variations of these scenarios, but for our general discussion, these three scenarios are sufficient.

All three general scenarios shown are possible. As illustrated, they may or may not result in successful development of databases. The key is, as it is with the traditional database design, the availability of complete and correct requirements. If a designer has access to complete and correct requirements, he or she can use generative AI and either verify the correctness of the generated result (in Scenario 1) or modify the generated result (in Scenario 2).

However, when faced with incomplete or incorrect requirements (which is, unfortunately, an all-too-common situation), the designer will have a difficult time producing a quality ER diagram and a database that functions as desired, whether generative AI is used or not. In fact, using generative AI could be counterproductive in such circumstances. Traditionally, designers are aware of assumptions and guesswork that they engage in when manually creating ER diagrams without proper requirements. They can try to investigate and verify some of the missing requirements and gradually improve the accuracy and completeness of their ER diagram. On the other hand, if they resort to generative AI to quickly create something without proper requirements, they are just creating faulty work faster. Even more dangerous is the possibility of not realizing that the generated diagram is faulty before spending a lot of effort developing a back and front end based on it.

Another issue to consider is productivity. In both successful scenarios (Scenario 1 and 2), the designer did not have to create the initial generated diagram but did have to put in effort to verify it. In Scenario 2, he or she also had to modify the generated diagram. Depending on the usability and quality of the software and the designer's level of skill and experience, the verification/modification could take more time for the designer than if he or she created the diagram himself or herself. Furthermore, switching from an active designer to a more passive verifier role introduces a risk of more subtle mistakes, made by the AI, going unnoticed.

There is no doubt that when provided with high-quality (accurate and complete) requirements, generative AI tools will be able to produce perfectly usable database diagrams and schemas. However, producing perfectly usable database diagrams and schemas based on high-quality requirements was never an issue or a problem even before the advent of generative AI tools. The most critical and time-consuming aspect of database design always was (and still is) coming up with the high-quality requirements. This involves the proper engagement of all constituents of the database project: designers, clients, management, users, and so on. This is not to say that generative AI could not be of assistance in that process. But the true engagement of all necessary participants will remain pivotal.

Generative AI and Database Analysis

In this part of the article, we will focus on using generative AI in assisting with the process of writing SQL queries. Using generative AI while creating SQL queries is already a standard industry practice. However, it is imperative that the analyst in charge of creating SQL queries has significant knowledge and experience with the relational database model and the SQL syntax.

Let us first consider the necessity of understanding the relational model. To clearly and reliably specify requirements for SQL queries, the access to the relational schema of the database that is being queried and the understanding of it are vital. Consider the relational schema for the HAFH Realty Company shown in Figure 3.59 in Chapter 3 and Figure 5.50a in Chapter 5, repeated here as Figure 3 (for convenience).

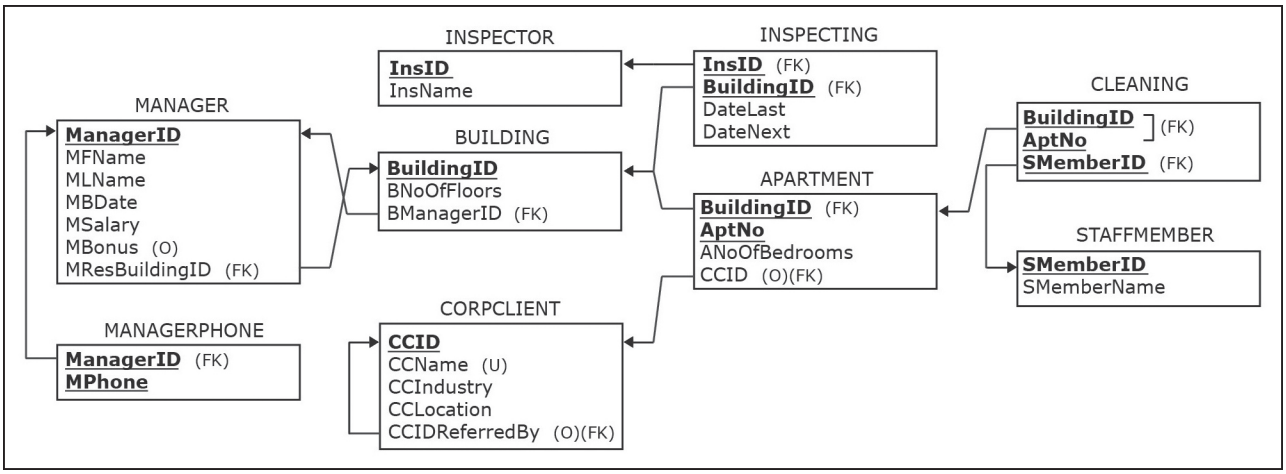

Figure 3 HAFH Realty Company database—relational schema.

The metadata shown in the schema explains what kind of SQL queries *could be issued* and what kind of queries are *impossible*. For example, we see that we *could create a query* that displays the ID and salary for each manager that manages buildings with five or more floors. The relational schema indicates that the database keeps track of managers and their salaries, and of buildings and the number of floors for each building. The relational schema also indicates that each building is associated with its manager.

As another example, we can see that we *cannot create a query* that displays the ID and salary for each staff member managed by managers hired within the last five years. The relational schema shows that there is no information in this database about staff members' salaries and managers' dates of hire. Moreover, there is no information in this database about which managers manage which staff members. However, if asked to produce a query such as the one just described, AI could easily generate a seemingly correct-looking SQL script that references columns or even tables that do not actually exist in the database.

Therefore, in addition to understanding the relational model, the database analysts should have the knowledge and experience with the SQL syntax, even when generative AI is used to create SQL queries. To illustrate, we will discuss AI-generated SQL queries for the HAFH database. As an experiment, we used one of the popular generative AI tools; we first provided it with the HAFH relational schema, and then we asked it to generate a number of SQL queries. In a substantial majority of cases, the AI-generated SQL code was correct. As an example of correctly generated SQL code, consider the following Query A. Query A text was given as a prompt in the generative AI tool, and the query was generated correctly.

```
Query A text: Create an SQL query that displays the BuildingID, AptNo,
              and ANoOfBedrooms for all apartments that have more than
              one bedroom.
Query A generated by an AI tool (correctly):
   select   buildingid, aptno, anoofbedrooms
   from     apartment
   where    anoofbedrooms > 1
```

However, there were instances when the generative AI tool made mistakes. For example, consider Query B. We first show the text of the query and the correct code.

```
Query B text: Create an SQL query that displays the total amount HAFH
              spends on manager salaries (as TotalSalary) and the total
              amount HAFH spends on manager bonuses (as TotalBonus).
Query B correct version:
   select   sum(msalary) as totalsalary, sum(mbonus) as totalbonus
   from     manager;
```

When Query B text was given as a prompt in the generative AI tool (at the time of this writing), the query was generated incorrectly.

```
Query B generated by an AI tool (incorrectly):
   select   sum(ssalary) as totalsalary, sum(sbonus) as totalbonus
   from     staffmember
   where    sposition = 'Manager'
   and      buildingid in  (select buildingid
                            from building
                            where bowner = 'HAFH');
```

Even though in the majority of cases the AI tool used proper tables and columns and generated correct SQL queries for the HAFH database, in this case (Query B), the query was completely off target. The mistakes in generated SQL code include the following: columns ssalary, sbonus, sposition, and bowner do not exist anywhere in the HAFH database; table manager should have been used instead of table staffmember; the nested query is a meaningless hallucination.

This example illustrates that there is a possibility of AI tools generating incorrect SQL queries. In such cases, it is imperative that the database analysts recognize the mistakes and are able to correct them using their knowledge and experience with the SQL syntax.

Learning Versus Professional Use

The use of generative AI is already becoming mainstream among professional database designers and analysts. Highly skilled database designers and users are usually well equipped to recognize and improve usable and useful generated content and filter out incorrect or off-target generated content. While doing so, they rely heavily on their knowledge of database design and use skills and methods (such as ER modeling, relational modeling, and SQL).

On the other hand, using generative AI while learning those database design and analysis skills and methods should be done carefully and in a way that does not impede and minimize the learning process.

For discussion purposes, consider the following scenario. Students A, B, and C are new learners of SQL, and student A studies the traditional way, while B and C use generative AI while studying. Their instructor has given them lectures on basic SQL DML commands (i.e., SELECT FROM, WHERE, JOIN, GROUP BY, HAVING, etc.) followed by examples of various SQL queries. After that, their instructor has assigned multiple homework assignments requiring creation and running of dozens of SQL queries of varying complexity, utilizing all previously presented SQL commands.

Student A—Creates queries manually, runs them, and debugs and corrects the code when needed. Eventually, he or she gets most or all of the homework correct. Through this process, he or she gradually learns the logic of commands, increases level of skill and confidence, and eventually becomes proficient.

Student B—Heavily relies on generative AI for producing the SQL code. He or she is focused on getting the results and quickly accepts the generated code without closely inspecting it or understanding it. He or she gets most or all of the homework correct but misses out on true learning of the language.

It is important to avoid giving students a false sense of learning. Student B could have genuinely believed that he or she was actually learning a skill due to producing mostly correct results. However, in reality, it will not be possible for him or her to engage in professional use of generative AI. Eventually, he or she will be expected to fully understand and often modify the generated code, which will not be possible without actually knowing the fundamentals of how the code functions.

This is not to say that there is no room for using generative AI in the database learning process. Consider Student C.

Student C—Creates his or her queries manually and then compares and verifies those queries versus queries created using generative AI. While doing so, he or she observes and identifies any errors made by either creator, thus both honing database skills and preparing for a professional career where AI will be used as a tool. Moreover, he or she may find generative AI useful as an on-hand tutor. After comparing his or her own SQL queries versus queries created using generative AI, he or she can then question the AI tool as to the reasoning behind its choices. He or she is still developing hands-on understanding and, instead of offloading learning to AI, expanding his or her understanding through curiosity and exploration.

Considerations for Professional Use of Generative AI

Most exercises used in an introductory database course are relatively short and simple, and thus can be solved easily by current AI models. Meanwhile, in a professional setting, queries can become complex and long, easily exceeding dozens of lines of code. Such queries are much harder for AI to produce, and as of 2026, even the leading AI models are incapable of consistently producing correct complex SQL queries even when given an accurate text prompt. Nevertheless, the professional use of generative AI is quickly advancing.

While generative AI has a growing user base among professional database designers and practitioners, a foundational understanding of database design and query syntax remains essential to professional development and success. Here we briefly consider several aspects of generative AI use in the context of understanding the principles of database design and analysis.

Prompt engineering: The practice of designing, structuring, and refining how one instructs inputs to a generative AI. It is true that the more adept one is at prompting an AI, the more likely he or she is to receive valuable results. A foundational understanding of database design and SQL is necessary to get the best results from AI.

Hallucination: Without a fundamental understanding of the questions to be asked—the requirements for design or query—a user is likely to experience situations where the AI generates responses such as in Scenarios 2 and 3 or Query B. Even with careful prompt engineering, users must carefully review outputs to mitigate hallucination. These are activities that one well versed in design principles and SQL syntax will perform more effectively.

Model selection and reasoning: At the time of this article, companies providing generative AI tools have been consistently releasing new versions of their AI models with varying degrees of increased performance and quality. Reasoning models, which take extra time to process a prompt and plan multistep tasks, as an example, can provide notable improvements to

response quality. Regardless of increased quality, the need for creating complete and quality requirements, and possessing a fundamental knowledge of how databases are created and analyzed will not dissipate.

Agents: Beyond individual model performance, there is increasing development on what is known as an agentic architecture for generative AI. These systems consist of multiple AI processes, each explicitly prompted to perform specific activities, and in some cases, granted access to specific reference data, to perform specialized functions. Such architectures have the potential to support database practitioners by guiding them through requirements gathering, prompting clarifying questions, and encouraging higher-quality specifications. While such systems are feasible, practitioners with strong design expertise will ultimately work more efficiently and produce higher-quality outcomes.

Conclusion

In this article, we outlined the opportunities and challenges related to using generative AI when designing databases and writing SQL queries. While advances in the capabilities of AI have been substantial, the importance of developing and maintaining skills in database design and writing SQL code has not diminished. As AI tools become a part of professional database design and use, a strong foundation in requirements collection, conceptual modeling, and SQL is still needed to effectively and safely use AI in the context of databases.

Generative AI and its use with database systems are rapidly evolving and quickly changing. In this article, we have discussed the current situation. We did not want to engage in predictions, speculations, or hallucinations (pun intended) about the future, but we are aware that an article like this one may have to be expanded, revised, or rewritten in the not-so-distant future.