Using DeepSeek LLM AI System on Database Concepts and Design

Putsadee Pornphol
Department of Digital Technology
Phuket Rajabhat University
Phuket, Thailand
putsadee.p@pkru.ac.th

Suphamit Chittayasothorn
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
suphamit.ch@kmitl.ac.th

***Abstract*—**The use of Large Language Model (LLM) artificial intelligence (AI) systems in learning environments is ubiquitous. They are used by learners for self-studying, assignments, and exam preparation. Practitioners use them as junior assistants. Major concerns include the correctness and the completeness of the answers generated by the AI systems. In this paper, we explore how well an LLM AI system DeepSeek, answers database concepts and design questions. Selected questions on database design and frequently misunderstood topics are asked. Related knowledge is explained. The answers are reviewed and marked to provide an assessment on its use in the subject area.

Keywords—DeepSeek, LLM AI, Database Design, ER Model, Relational Database

I. INTRODUCTION

Large Language Models (LLM) [1,2] represent a transformative advancement in artificial intelligence. These models are built on deep learning architectures, particularly transformers, which analyze and predict linguistic patterns. Trained on massive datasets comprising text from various sources such as books, websites, and articles, LLM systems are very good at tasks like text generation, summarization, translation, and question-answering. Their ability to comprehend context and answer questions in various applications has made them applicable across industries including education.

DeepSeek [3] is an emerging tool from China which employs a Mixture-of-Experts (MoE) architecture that allows for efficient scaling. From an economic standpoint, DeepSeek's emergence poses a challenge to established AI competitors. One of the key differentiators of DeepSeek is its open-source nature. This allows for collaborative development and customization by a global community of researchers and engineers, potentially leading to fast iteration and improvement.

In this paper, due to its popularity among students and practitioners, and free of charge, DeepSeek is chosen to answer questions on database concepts, design, and programming. Carefully selected questions are used as inputs to the system. These questions are frequently misunderstood questions and have been used in database-related examinations. The answers are then evaluated, and marks are also given. The motivation behind this work is to demonstrate the capability of the system in the scope of database principles, analysis, and design.

II. THE TEST QUESTIONS

The questions which are asked as prompts to the DeepSeek system are as follows.

1. A hospital would like to have an information system for patients' food preparation. Patient's personal details are to be recorded and used. Treatments and illness related to each patient are recorded. Prohibited food for those treatments and illness are also recorded. Moreover, each patient's personal preferences (vegetarian, Muslim, low fat, etc.) are also recorded. Using the Entity Relationship Model (Peter Chen Model), design 5NF relational database schemas for the above application and explain clearly why you think your design is in 5NF. (30 Marks)
2. Explain the 3-level database architecture in detail and show how the architecture is implemented using the relational database model. (20 Marks)
3. In the database area, what is a conceptual schema model? (10 Marks)
4. What is a relation? Show two possible representations of a relation. (10 Marks)
5. Describe the relational database model. What is a relational DBMS? (20 Marks)
6. Given the following relational schema: EMP (E#, ENAME, ADDRESS, SALARY, DEPT). Is this schema in 5NF? Why? (10 Marks)

III. THE KNOWLEDGE REQUIRED FOR THE FIRST QUESTION ON THE ER MODEL.

The first question requires the Entity Relationship (ER) [4] design knowledge. The students are expected to draw an ER diagram, transform it to relational database schemas, and further normalize to achieve the 5NF relational database schemas. The main idea of the ER model is to model real-world objects and the ways they are related. The two basic constructs of the Entity Relationship Model (ERM), as the name implied, are the entity and the relationship. Entities are objects of interest in the application domains. They represent real-world objects which take part in the application. An information system can be thought of as a computerized system which simulates the real-world application. All objects involved in such application therefore need to be properly represented in the system and the ERM refers to them as entities. Each entity has one or more attributes which are properties of entities. Relationships are simply relationships among entities. Note that the term “relationship” is very similar to the term “relation” but these two terms cannot be interchanged. A relation is a subset of the Cartesian product of domains. It is the main data structure of the relational database model and must not be confused with the relationship between entities in the ERM. The concept of entities can be applied to both concrete, or tangible objects and abstract, or intangible objects. It is important to be aware that the term entity is actually equivalent to the term object instance in the object paradigm. Hence an entity is formally called an entity instance. Entity instances which have the same set of attributes can be grouped together to be the same entity type. The concept of entity type corresponds to the concept of object class in the object paradigm.

The term attribute is a common English term. It is therefore used by other data models including the relational database model. However, in the relational database model, the value of an attribute must be atomic. In the ER model, both atomic and non-atomic attributes are allowed. Simple attributes which have only one value for each entity instance is very common in the ER model. However, composite, and multivalued attributes are also allowed in the ER model. Attributes whose values are unique are called the key attribute of the entity type. The hardest part of drawing an ER diagram is the identification of entity types. This task seems to be natural. However, it is not. Beginners often confuse entity types with processes and reports. Remember processes are procedures or algorithms and are implemented by application programs. Reports are either input or output of the applications, but we are designing databases which are beneath the applications. Entity types must represent objects of interest; the main characters which are involved in the applications. These entity types will later be transformed into some machine accessible logical data structures. Entity instances will be represented by some database instances such as tuples. It is therefore important to correctly declare entity types from the given application domains.

From the given case study, students or LLM AI systems should be able to identify entity types, related them and identify relationship types, add cardinalities, and transform the finished ER diagram to 1NF relational database schemas. Since the ERM and other similar conceptual models do not express relationships between attributes, functional dependencies (FDs), multivalued dependencies (MVDs), and join dependencies (JDs) cannot be obtained from an ER diagram and higher normal form of relational database schemas such as 3NF [5], BCNF [6], or 5NF [7] cannot be obtained and further normalization is required. This first question is there for a challenging one.

IV. The Knowledge Required for the Second Question on the 3-Level Database Architecture.

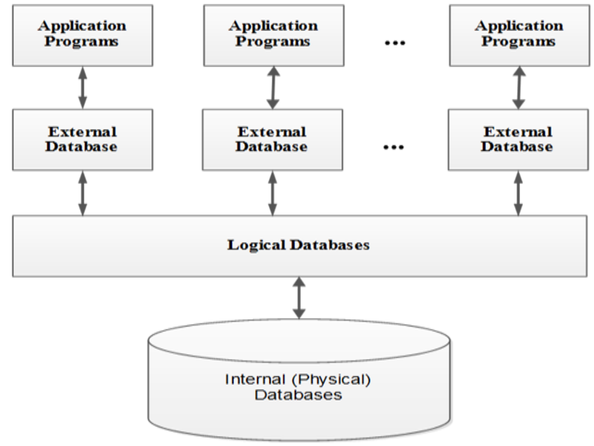

Fig. 1. The 3-level database architecture.

The external level, logical level, and physical level database structures form the 3-level database architecture or the 3-schema database architecture. In the database area, the term “schema” technically means a type or a structure. A relational database schema is a table structure, excluding the data rows. The term logical database schema therefore refers to the structure or data definition that describes the structure of the logical database. An external schema is a subset of the logical schema. It is the structure or data definition of the database as seen by the application programs. A physical or internal schema is the structure or data definition of the physical database such as files and indexes.

According to the 3-schema architecture in Figure 1, application programs and users which are shown on the uppermost level submit database requests to retrieve, insert, delete, and modify the database. The programs see external-level data according to the external schemas. Their requests are therefore based on the external schemas.

Requests on the external-level data are translated to requests on logical databases. This is called the external/logical mapping. The requests on logical-level data are then further translated into requests on physical-level data such as indexes and files. This is called the logical/internal mapping; it is actually the query optimization process which is previously mentioned.

Note that even though this architecture is the simplified version of the standard database architecture issued by ANSI-SPARC (American National Standards Institute, Standards Planning and Requirements Committee) Architecture [8], it is only a recommended architecture. The ANSI/SPARC database architecture refers to the middle database level as the Conceptual Views, not the “Logical databases” as shown in Figure 1. This term “conceptual” in the architecture leads to many confusions among academics and students. Students and LLM AI systems should know that the middle level is the working logical database structure, not a conceptual model likes the Entity Relationship Model. To further illustrate this important 3-level architectural topic. An example based on the relational database model is shown in Figure 2.

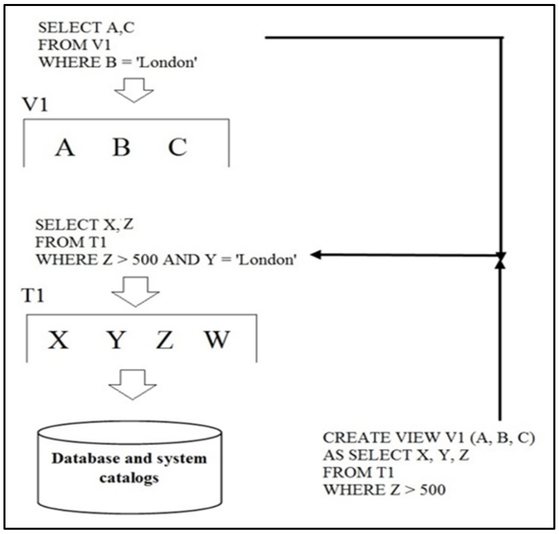

Fig. 2 An Example of the 3-level database implementation using the relational database model.

In this Figure 2, there is a logical database table T1 with the columns X, Y, Z, and W. Suppose there is an application program that would like to access only the first three columns of T1, and rows that meet a certain condition. For some reasons, the application would also like to refer to the columns X, Y, Z as A, B, C respectively. In SQL, a view is a virtual table which is defined based on a real table (or base table) or even on other views. The view is the SQL implementation of the external database concept. The table is the SQL implementation of the logical database concept.

In this case, a view V 1)A,B,C) can be created based on the table T1. The view V1 definition in SQL is shown below:

```
CREATE VIEW V 1)A, B, C)
AS SELECT X, Y, Z
FROM T1
WHERE Z > 500
```

After the successful execution of this command, a view is created. When a view is created, only the view definition is kept in the system area of the database. This area is generally called the system catalogs or system tables. There is no redundant copy of data for the view created. However, some DBMSs may allow the creation of views which store redundant snapshot portions of base tables. Such a view is called a materialized view. This option must be specified when the view is created.

V. THE KNOWLEDGE REQUIRED FOR THE THIRD QUESTION ON RELATIONS

The term relation, according to its mathematical definition, is *a subset of the Cartesian product of domains* .According to the Codd's foundation papers [9] a relation in relational database is also defined in the same way :

Given sets $D_1, D_2, \ldots, D_n$ domains which are not necessarily distinct, r is a relation on these n sets if it is a set of n-tuples each of which has its first element from D_1, its second element from D_2, and so on .More concisely, r is a subset of the Cartesian product D_1 x D_2 x …x D_n where D_i is called the ith domain of r. As defined here, r is said to have degree n .Thus, a relation r is also a set of n-tuples)$a_1, a_2, \ldots, a_n$ (where each $a_i \in D_i$.

According to [10], instead of using the corresponding position in the value sequence of a tuple to determine a domain, an alternative way is to associate with each tuple value not only its domain, but also its distinct role called its attribute. The n distinct attributes of a relation of degree n distinguish the n roles of the domains on which the relation is defined. A relation then comprises a set of tuples, each of which has the same set of attributes. Based on this concept, the order of the attributes of a relation is not significant. Each value of an attribute is associated with only one domain.

As mentioned above, a relation is the data structure of the relational database model. It is not a part of the ERM. It is not the same as the relationship in the ERM. It is also not a link or relationship between tables. In fact, it is the table itself, with a certain set of properties that the students or the LLM AI should know.

VI. THE KNOWLEDGE REQUIRED FOR THE FOURTH AND FIFTH QUESTIONS ON THE RELATIONAL DATABASE MODEL AND RDBMS

A database model comprises three main parts. The first one is the logical database structures which represent data objects and relationships among them. These are the data structures as seen by application developers or users. In the case of the relational database model, its only data structure is relation, as defined in the previous section. The second one is the integrity constraints on the data structures. To ensure data integrity at the logical data structure level, the relational database model provides two structural constraints namely, the entity integrity, and the referential integrity. The Entity Integrity: Primary key values must not be null. The Referential Integrity: Foreign key values must match primary key values or be null. The third part is the data manipulation languages that work on the logical database structures. The relational database model has two original data manipulation languages namely the relational algebra, and the relational calculus. The relational algebra is a procedural language used internally by relational DBMSs. The relational calculus is a more declarative language which was expected to be used by programmers and end-users. A relational complete language is defined as a language which is at least as powerful as the relational algebra or the relational calculus. SQL [11] is a relational complete language. An RDBMS is a database management system software that supports the relational database model. The term relational database actually refers to the relations, or the tables as opposed to the RDBMS software.

VII. THE KNOWLEDGE REQUIRED FOR THE SIXTH QUESTION ON THE 5NF TABLE

This question requires the concept of relational database normalization and the correct understanding of the 5NF and join dependencies. Basically, there are two kinds of relational database tables, the ones that cannot be further split, and the one that can be split. A relational table which can be split may be kept as it is. There is no point to split all the tables that can be split. Only those tables which have insert, delete, and update problems should be split. Said otherwise, we should split the relational table structures only when it does not satisfy a desired normal form.

For those table that cannot be split, there is nothing to do with them. The only question left, and in fact the only point of performing the normalization process is: for a relational table that can be split, should it be kept as it is, or should it be split? Our answer is: if it is already good, keep it. So, what is good? Good here means the best normal form that the design team understands. In our case, we aim at the fifth normal form (5NF). This highest level of normal form refers to the join dependency (JD).

Definition: Relational table structure T satisfies the join dependency (JD): * (A, B, C,.., N) if and only if T is equal to the join of its projections on A, B, C,…, N, where A, B, C, . . . N are subsets of the set of attributes of T.

The concept of join dependency is in fact very simple: what does it mean by "if T is equal to the join of its projections on A, B, C,.., N" ? The answer is: T is equal to the join of its projections on A, B, C,…, N, means T can be split into A, B, C,…, N. So, a join dependency is the ability to split a relational table. It is as simple as that.

Definition: A relational table structure T is in projection-join normal form (PJ/NF) or the fifth normal form (5NF) if and only if every join dependency in T is implied by the candidate keys of T.

An immediate consequence of this definition is that, if a relational table structure cannot be split, it is already in 5NF, since it has no JDs. The only remaining question is: if a relational table structure can be split, should we leave it as it is, or split it? The answer is also very simple: if the small tables after the split always have a candidate key of the big table, then leave the big table as it is.

Consider the sixth question: Given the following relational schema: EMP (E#, ENAME, ADDRESS, SALARY, DEPT). Is this schema in 5NF? Why?

This simple EMP relational schema (table structure) has E# as the primary key. E# -> ENAME, E# -> ADDRESS, and E# -> SALARY) so this table structure can be split into smaller ones but always with the primary key E# in each small projection after the split. According to the 5NF definition, EMP is already in 5NF and therefore should be kept as it is since it is already in 5NF. There is no need to split it.

VIII. Experimental Results From DeepSeek

Question 1: A hospital would like to have an information system for patients' food preparation. Patient's personal details are to be recorded and used. Treatments and illness related to each patient are recorded. Prohibited food for those treatments and illness are also recorded. Moreover, each patient's personal preferences (vegetarian, Muslim, low fat, etc.) are also recorded.

Using the Entity Relationship Model (Peter Chen Model), design 5NF relational database schemas for the above application and explain clearly why you think your design is in 5NF.

Partial Answer to Question 1:

To design a 5NF (Fifth Normal Form) relational database schema for the hospital's patient food preparation system, we need to ensure that the database is free from redundancy and anomalies by decomposing the tables into the smallest possible units of related data. Below is the Entity-Relationship (ER) model and the corresponding relational schema, followed by an explanation of why it satisfies 5NF.

Entities and Relationships:

1. Patient: Stores patient personal details.
   Attributes: PatientID (PK), Name, DOB, Gender, ContactInfo.
2. Treatment: Stores treatments administered to patients.
   Attributes: TreatmentID (PK), TreatmentName.
3. Illness: Stores illnesses diagnosed for patients.
   Attributes: IllnessID (PK), IllnessName.
4. ProhibitedFood: Stores food items prohibited for specific treatments or illnesses.
   Attributes: ProhibitedFoodID (PK), FoodItem, TreatmentID (FK), IllnessID (FK).
5. FoodPreference: Stores patient-specific food preferences.
   Attributes: PreferenceID (PK), PreferenceType (e.g., vegetarian, Muslim, low fat).
6. PatientTreatment: Links patients to their treatments.
   Attributes: PatientID (FK), TreatmentID (FK).
7. PatientIllness: Links patients to their illnesses.
   Attributes: PatientID (FK), IllnessID (FK).
8. PatientPreference: Links patients to their food preferences.
   Attributes: PatientID (FK), PreferenceID (FK).

Explanation of 5NF

1. Atomicity: Each table represents a single entity or relationship, and all attributes are atomic (indivisible).
2. No Redundancy: The schema avoids redundancy by separating entities (e.g., Patient, Treatment, Illness) and relationships (e.g., PatientTreatment, PatientIllness).
3. Lossless Decomposition: The schema ensures that no information is lost when joining tables. For example, joining PatientTreatment and Treatment will correctly retrieve all treatments for a patient.
4. Dependency Preservation: All functional dependencies are preserved. For example, ProhibitedFood depends on both Treatment and Illness.
5. No Join Dependency: The schema is free from join dependencies because all tables are in their smallest possible form. For example, PatientPreference captures the relationship between patients and their food preferences without requiring additional joins to infer relationships.

Question 2: Explain the 3-level database architecture in detail and show how the architecture is implemented using the relational database model.

Partial Answer to Question 2:

The 3-level architecture is implemented in relational databases as follows:

1. External Level

Implemented using views in SQL.

Views are virtual tables that provide a customized representation of the data stored in the base tables.

Example:

```
CREATE VIEW PatientView AS
SELECT PatientID, Name, DOB, Gender
FROM Patient;
```

2. Conceptual Level

Implemented using base tables and constraints in SQL. Base tables represent the logical structure of the database. Example:

```
CREATE TABLE Patient (
   PatientID INT PRIMARY KEY,
   Name VARCHAR(100),
   DOB DATE,
   Gender CHAR(1)
);

CREATE TABLE Treatment (
   TreatmentID INT PRIMARY KEY,
   TreatmentName VARCHAR(100)
);

CREATE TABLE PatientTreatment (
   PatientID INT,
   TreatmentID INT,
   PRIMARY KEY (PatientID, TreatmentID),
   FOREIGN KEY (PatientID) REFERENCES
Patient(PatientID),
   FOREIGN KEY (TreatmentID) REFERENCES
Treatment(TreatmentID)
);
```

3. Internal Level

Managed by the Database Management System (DBMS). Includes physical storage details such as file organization, indexing, and data compression.

Base tables for the conceptual level, and

Physical storage structures managed by the DBMS for the internal level.

Question 3: What is a relation? Show two possible representations of a relation.

Partial Answer to Question 3:

A relation in the context of relational databases is a fundamental concept that represents a table with rows and columns. It is based on the mathematical concept of a relation from set theory. A relation consists of:

1. Attributes (Columns): These define the properties or fields of the relation. Each attribute has a name and a domain (data type).
2. Tuples (Rows): These represent individual records or instances of the relation. Each tuple contains values corresponding to the attributes.

A relation is often denoted as

R(A1,A2,…,An)R(A1,A2,…,An), where:

- RR is the name of the relation.
- A1,A2,…,AnA1,A2,…,An are the attributes.

Properties of a Relation

1. Atomic Values: Each cell in the table contains a single, indivisible value.
2. Unique Tuples: No two tuples in a relation are identical.
3. Unordered Tuples: The order of tuples in a relation is not significant.
4. Unordered Attributes: The order of attributes in a relation is not significant (though it is fixed in the schema).

Two Possible Representations of a Relation

1. Tabular Representation

The most common way to represent a relation is as a table. For example, consider a relation Patient with attributes PatientID, Name, DOB, and Gender:

PatientID	Name	DOB	Gender
1	John Doe	1990-05-15	M
2	Jane Smith	1985-08-22	F
3	Alice Brown	1978-12-10	F

• Attributes: PatientID, Name, DOB, Gender.

• Tuples: Each row represents a tuple, e.g., (1, "John Doe", "1990-05-15", "M").

2. Set-Theoretic Representation

A relation can also be represented as a set of tuples, where each tuple is an ordered list of attribute values. For example, the same Patient relation can be written as:

Patient={(1,John Doe,1990-05-15,M),(2,Jane Smith,1985-08-22,F),(3,AliceBrown,1978-12-10,F)}Patient={(1,John Doe,1990-05-15,M),(2,Jane Smith,1985-08-22,F),(3,Alice Brown,1978-12-10,F)}

- Each tuple is enclosed in parentheses.
- The order of tuples in the set does not matter.

Question 4: In the database area, what is a conceptual schema model?

Partial Answer to Question 4:

In the context of databases, a conceptual schema model is a high-level, abstract representation of the entire database structure. It defines the overall logical organization of the data, including the entities, attributes, relationships, and constraints, without specifying how the data will be physically stored or how it will be accessed by users. The conceptual schema serves as a bridge between the external level (user views) and the internal level (physical storage).

Key Features of a Conceptual Schema Model

1. Entity Definition: Identifies the entities (e.g., Patient, Treatment, Illness) and their attributes.
2. Relationships: Defines the relationships between entities (e.g., a Patient receives a Treatment).
3. Constraints: Specifies rules such as primary keys, foreign keys, and domain constraints (e.g., PatientID must be unique).
4. Data Independence: Abstracts away implementation details, ensuring that changes to the physical storage or user views do not affect the conceptual schema.
5. Comprehensive View: Provides a unified view of the entire database, integrating all user perspectives.

Purpose of a Conceptual Schema Model

1. Design Clarity: Helps database designers and stakeholders understand the structure and relationships of the data.
2. Communication: Acts as a communication tool between database designers, developers, and end-users.

3. Foundation for Implementation: Serves as the basis for creating the logical schema (e.g., tables in a relational database) and the physical schema (e.g., file storage structures).

4. Data Integrity: Ensures that the database is designed to maintain consistency and enforce constraints.

Question 5: Describe the relational database model. What is a relational DBMS?

Partial Answer to Question 5:

Relational Database Model

The relational database model is a data management approach that organizes data into tables (also called relations) consisting of rows and columns. It is based on the mathematical concept of relations from set theory and was first introduced by Edgar F. Codd in 1970. The relational model is the foundation of modern database systems and is widely used due to its simplicity, flexibility, and robustness.

Key Concepts of the Relational Database Model

1. Tables (Relations):

A table is a collection of related data organized into rows and columns.

Each table represents an entity (e.g., Patient, Treatment), and each row represents an instance of that entity (e.g., a specific patient).

Columns represent attributes (e.g., PatientID, Name, DOB).

2. Tuples (Rows):

A tuple is a single row in a table, representing a record or instance of the entity.

Each tuple contains values for each attribute in the table.

3. Attributes (Columns):

An attribute is a named column in a table, representing a property of the entity.

Each attribute has a domain, which defines the set of valid values it can hold (e.g., integer, string, date).

4. Primary Key:

A primary key is a unique identifier for each tuple in a table.

It ensures that no two rows have the same value for the primary key attribute(s).

5. Foreign Key:

A foreign key is an attribute (or set of attributes) in one table that refers to the primary key of another table.

It establishes relationships between tables.

6. Relationships:

Relationships define how tables are connected to each other. Common types of relationships include one-to-one, one-to-many, and many-to-many.

7. Constraints:

Constraints enforce rules on the data to ensure integrity and consistency. Examples include primary key constraints, foreign key constraints, and domain constraints.

Question 6: Given the following relational schema: EMP(E#, ENAME, ADDRESS, SALARY, DEPT). Is this schema in 5NF? Why?

Partial Answer to Question 6:

To determine whether the given relational schema EMP(E#, ENAME, ADDRESS, SALARY, DEPT) is in Fifth Normal Form (5NF), we need to analyze the schema in the context of normalization principles, particularly focusing on join dependencies and lossless decomposition.

What is 5NF?

A relation is in Fifth Normal Form (5NF) if:

1. It is in Fourth Normal Form (4NF) (i.e., it has no multi-valued dependencies).

2. It cannot be further decomposed into smaller relations without losing information or introducing redundancy.

3. It has no join dependencies that are not implied by its candidate keys.

In simpler terms, 5NF ensures that the relation is free from redundancy and cannot be broken down further without losing the ability to reconstruct the original data through joins.

Analysis of the Schema

The schema EMP(E#, ENAME, ADDRESS, SALARY, DEPT) has the following attributes:

- E#: Employee ID (primary key).
- ENAME: Employee name.
- ADDRESS: Employee address.
- SALARY: Employee salary.
- DEPT: Department to which the employee belongs.

Step 1: Check for Functional Dependencies

- The primary key is E#, which uniquely identifies each employee.
- The functional dependencies are:

E# → ENAME, ADDRESS, SALARY, DEPT.

Since all attributes depend solely on the primary key (E#), the schema is in Second Normal Form (2NF) and Third Normal Form (3NF).

Step 2: Check for Multi-Valued Dependencies

- There are no multi-valued dependencies in this schema because all attributes are functionally dependent on the primary key (E#).
- Therefore, the schema is in Fourth Normal Form (4NF).

Step 3: Check for Join Dependencies

- A join dependency exists if the relation can be decomposed into smaller relations and then recombined using a natural join without losing information.
- In this schema, there is no indication of any join dependencies because all attributes are directly dependent on the primary key (E#).
- The schema cannot be decomposed further without losing the ability to reconstruct the original data.

Conclusion

The schema EMP(E#, ENAME, ADDRESS, SALARY, DEPT) is in 5NF because:

1. It is in 4NF (no multi-valued dependencies).

2. It cannot be further decomposed without losing information.

3. It has no join dependencies that are not implied by its candidate key (E#).

Why is it in 5NF?

- The schema represents a single entity (Employee) with all attributes functionally dependent on the primary key (E#).
- There is no redundancy or unnecessary decomposition.
- The schema is already in its simplest and most efficient form.

If the schema had additional relationships or dependencies (e.g., employees working in multiple departments or having multiple addresses), it might not be in 5NF. However, based on the given schema, it satisfies all the conditions for 5NF.

IX. Marking Answers from DeepSeek.

The answers from DeepSeek are very good. We mark them as if they were answers from exam questions. The knowledge required to answer the questions are extracts from [12]. DeepSeek obtains 86/100 in total, which is likely to be an A. The first question on the database design using ER diagram is good. The ER diagram produce is correct. Entity types and relationship types are specified correctly. It does not draw a visual diagram but provides descriptions and easy-to-understand schemas. Mapping from the ER diagram to relational database schemas is also correct. However, the explanation of join dependency is not perfect. In some tables, there are still some join dependencies which are implied by a candidate key, but DeepSeek reports that there are no join dependencies. It seems to refer to cyclic dependencies when it refers to join dependencies. In fact, there are many join dependencies which are not cyclic dependencies. We give it 25/30 for this question.

The answer to the question 2 on the 3-schema database architecture is also very good. It gives details explanations with illustrated examples using the case study from the question 1. DeepSeek correctly describes the conceptual schema in this context using relational database views and tables as examples. We give it a full mark of 20/20.

The answer to the question 3 on relation is also very good. Even though DeepSeek does not use the mathematical definition of a relation, the answer and examples are correct. We give it a full mark of 10/10.

The answer to the question 4 on conceptual schema is good. However, DeepSeek does not emphasis that there are two different kinds of conceptual schema, the high level one used for human communication, such as the Entity Relationship Model and the low level one used for programming, such as the relational database model. We give it 8/10.

The answer to the question 5 on the relational database model and RDBMS is acceptable. However, this one can be improved. DeepSeek does not include the data manipulation languages of the relational database model, the relational algebra, and the relational calculus in the answer. It does not mention about relational complete languages. It mentions about relationships which connect tables together. This is not correct. There is no such data structure in the relational database model. Relationships belong to the Entity relationship Model. We therefore give it 15/20 for this question.

The answer to the question 6 is very good. DeepSeek correctly answers that EMP (E#, ENAME, ADDRESS, SALARY, DEPT) is in 5NF. However, it states a property that a 5NF relation cannot be further decomposed without losing information is not correct. The main point is further decomposition with at least a candidate key or not. The EMP table here is the case. We therefore give it 8/10. Conclusions

This paper verifies the use of The DeepSeek LLM AI system as a tutorial assistant or knowledge source for self-studying students and practitioners. At present, it is a free of charge system. Six carefully selected questions on relational database analysis and design are submitted to DeepSeek as prompts. The results are very good to excellent. Given a case study, database design using the Entity Relationship Model by DeepSeek gives correct results. Other questions which are relatively hard to answer even by experienced practitioners are answered. Even though they are not perfect, they are acceptable. We also test the questions on ChatGPT [13] and Copilot [14] free versions. The results from these tests are comparable to the results from DeepSeek. The use of these LLM AI systems as learning assistants in database principles, analysis, and design shows good potentials. Learners are advised to consult human experts or attend formal classes for better understanding of the topics.

References

[1] Information technology, Database languages, Part 2 Foundations, SQL, ISO/IEC 9075-1:2011

[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. arXiv:1901.02860

[3] DeepSeek-AI 2024. https://arxiv.org/abs/2412.19437

[4] P. P. Chen, "The Entity–Relationship Model – Toward A Unified View of Data," ACM Transactions on Database Systems, Association for Computing Machinery, NY, pp. 9-36, March 1976.

[5] Codd, E.F. "Further Normalization of the Data Base Relational Model," (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24–25, 1971.) IBM Research Report RJ909 (August 31, 1971). Republished in Randall J. Rustin (ed.), Data Base Systems: Courant Computer Science Symposia Series 6. Prentice-Hall, 1972.

[6] Codd, E. F. "Recent Investigations into Relational Data Base Systems," IBM Research Report RJ1385 (April 23, 1974). Republished in Proc. 1974 Congress (Stockholm, Sweden, 1974). , N.Y.: North-Holland, 1974. [9] Ronald Fagin, "Normal forms and relational database operators," in Proceeding SIGMOD '79 Proceedings of the 1979 ACM SIGMOD international conference on Management of data, pp. 153-160 Boston, MA, May 1979.

[7] Ronald Fagin, "Normal forms and relational database operators," in Proceeding SIGMOD '79 Proceedings of the 1979 ACM SIGMOD international conference on Management of data, pp. 153-160 Boston, MA, May 1979.

[8] ANSI/X3/SPARC Study Group on Data Base Management Systems: (1975), Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2

[9] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM vol. 13, June 1970.

[10] C.-C. Yang, Relational Databases. Englewood Cliffs, New Jersey: Prentice-Hall, 1986.

[11] Information technology, Database languages, Part 2 Foundations, SQL, ISO/IEC 9075-1:2011

[12] Suphamit Chittayasothorn, Relational Database Systems Language, Conceptual Modeling and Design for Engineers, Limited 500 hard copies published, not available online, Nutcha Printing, Bangkok, 2017.

[13] OpenAI. 2024. ChatGPT (March 2 version). OpenAI. https://openai.com.

[14] Microsoft Copilot, GPT-4. (2025). Personal communication