

**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

B.Tech. -Computer Science & Business Systems

Index

S.No	Regd. No.	Page No.
1	221FA20013	2
2	221FA20019	
3	221FA20028	
4	221FA20001	18
5	221FA20015	
6	221FA20026	
7	221FA20030	
8	221FA20016	37
9	221FA20014	
10	221FA20025	
11	221FA20005	
12	221FA20020	53
13	221FA20022	
14	221FA20029	
15	221FA20034	
16	221FA20006	68
17	221FA20010	
18	221FA20018	
19	221FA20023	
20	221FA20002	79
21	221FA20008	
22	221FA20021	
23	221FA20033	
24	221FA20003	94
25	221FA20004	
26	221FA20009	
27	221FA20027	
28	221FA20007	109
29	221FA20012	
30	221FA20017	
31	221FA20032	

A FIELD PROJECT REPORT ON

ER - DIAGRAM

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

INDURTHI TEJOONEELA (221FA20013)

BHASHYAM JEEVANA (221FA20019)

SHAIK ROOHI (221FA20028)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)

School of Computing & Informatics

Vignans' Foundation for Science, Technology and Research (Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April-2024



VIGNAN'S

Foundation for Science, Technology & Research
(Deemed to be University)

Established under UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "ENTITY RELATIONSHIP DIAGRAM" being submitted by INDURTHI TEJOONEEL -221FA20013, BHASHYAM JEEVANA-221FA20019, SHAIKROOIII-221FA20028 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**ENTITY RELATIONSHIP DIAGRAM**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

INDURTHI TEJOONEELA (221FA20013)

BHASHYAM JEEVANA (221FA20019)

SHAIK ROOHI (221FA20028)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech

SEMESTER : I

DEPARTMET : CSE CSBS

SECTION : 2A

BATCH : 01

Submitted by:

INDURTHI TEJOONEELA (221FA20013)

BHASHYAM JEEVANA (221FA20019)

SHAIK ROOHI (221FA20028)

TABLE OF CONTENTS

Name of the content

1. Abstract.....	6
2. Introduction.....	7
3. Entity Relationship Diagrams.....	8-9
3.1.What is ER diagram?.....	8
3.2.Why use ER diagram?.....	8
3.3.Components of ER diagram.....	9-15
3.3.1. Entities.....	10
3.3.2. Attributes.....	11
3.3.3. Relationships.....	13
4. Conclusion.....	15
5. References.....	16

Abstract

This paper explores the significance of Entity-Relationship (ER) Diagrams in database design, emphasizing their role in representing the structure and relationships between entities within a system. ER diagrams offer a visual approach to database modelling, simplifying the understanding and communication of database architectures among team members. The study delves into the key components of ER diagrams, including entities, attributes, and relationships, while also highlighting their application in various domains. By analysing their strengths and limitations, this research aims to provide insights into the best practices for creating effective ER diagrams, contributing to improved database design processes and better system performance.

Introduction

Entity-Relationship Diagrams (ER Diagrams) are a crucial part of database design, providing a visual representation of the relationships between various entities in a system. These diagrams help database designers, developers, and stakeholders to understand the structure of a database in a more intuitive and simplified manner. By representing entities as rectangles and relationships as diamonds, ER diagrams map out the interactions and associations within a system, ensuring that the database's structure is aligned with the actual needs of the organization or project.

ER diagrams also facilitate communication among team members, providing a common language to discuss database designs and requirements. This clear visualization helps in identifying potential issues such as redundant data, missing relationships, or incorrect associations early in the development process. As a foundational tool in conceptual data modelling, ER diagrams lay the groundwork for creating relational databases that efficiently store, retrieve, and manage data.

The significance of this database design lies in its ability to streamline user account management, enhance product categorization, and ensure certified shops maintain their standards. By addressing these core areas, the system not only supports current operational needs but also lays the groundwork for future expansions and feature integrations. As ecommerce continues to evolve, the adaptability and robustness of the database system will play a crucial role in maintaining a competitive edge, providing users with a reliable and efficient platform for their shopping experiences. This report is structured to guide the reader through each phase of the database design process, from conceptual modeling to practical implementation. By the end of this document, readers will gain a comprehensive understanding of the methodologies employed, the challenges encountered, and the solutions devised to create a cohesive and functional database system. The ensuing sections provide a detailed exploration of each component, underscoring the importance of meticulous planning and strategic execution in database development.

Entity Relationship Diagram – ER Diagram in DBMS

An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

What is an Entity Relationship Diagram (ER Diagram)?

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

Facts about ER Diagram Model:

ER model allows you to draw Database Design

It is an easy to use graphical tool for modeling data

Widely used in Database Design

It is a GUI representation of the logical structure of a Database

It helps you to identifies the entities which exist in a system and the relationships between those entities

Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

Helps you to define terms related to entity relationship modeling

Provide a preview of how all your tables should connect, what fields are going to be on each table

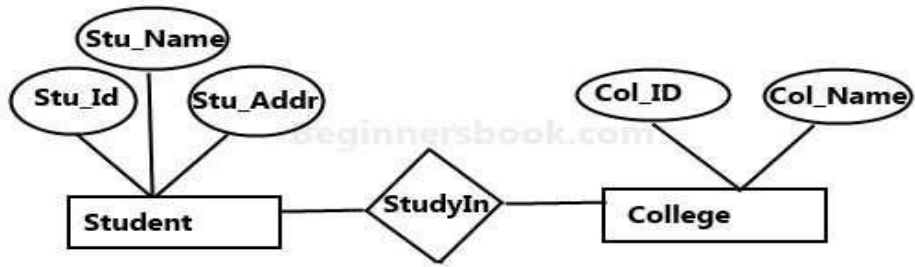
Helps to describe entities, attributes, relationships

ER diagrams are translatable into relational tables which allows you to build databases quickly

ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications

A simple ER Diagram:

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Sample E-R Diagram

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section (Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

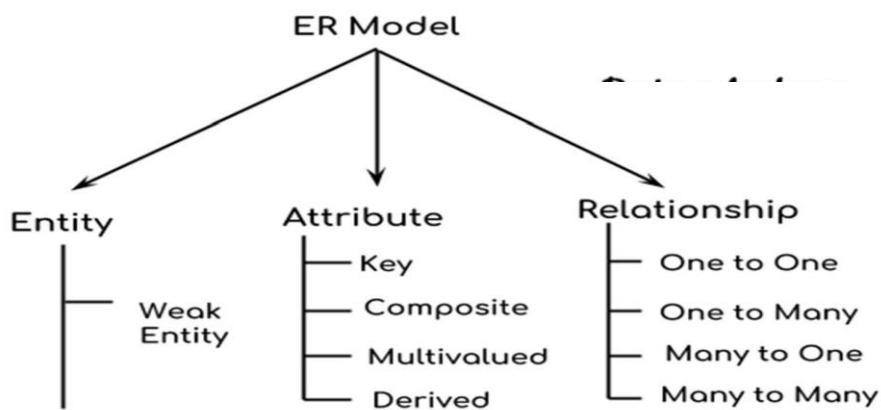
Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

Double Lines: Total participation of an entity in a relationship set

Components of a ER Diagram



Components of ER Diagram

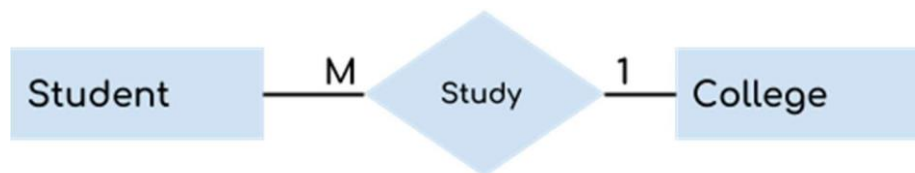
As shown in the above diagram, an ER diagram has three main components:

1. Entity
2. Attribute
3. Relationship

1. Entity

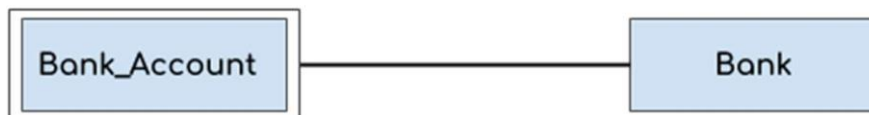
An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.

For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.



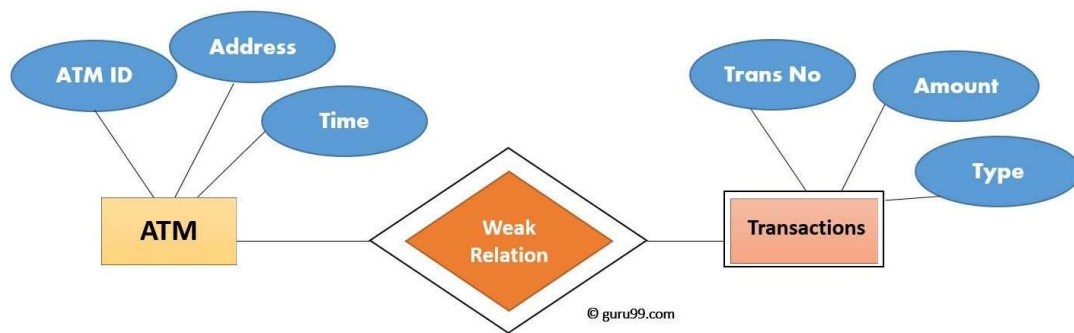
Weak Entity:

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



Weak Entities

A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.



In above example, "Trans No" is a discriminator within a group of transactions in an ATM. Let's learn more about a weak entity by comparing it with a Strong Entity

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

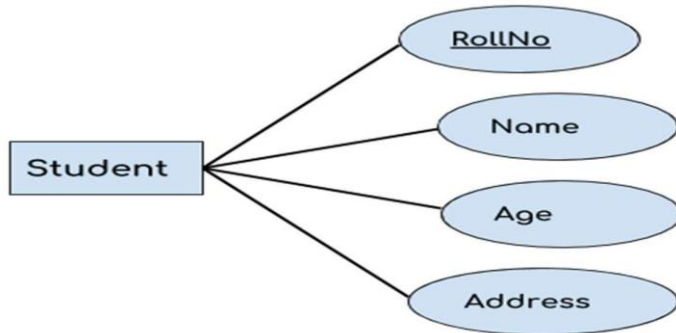
2. Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

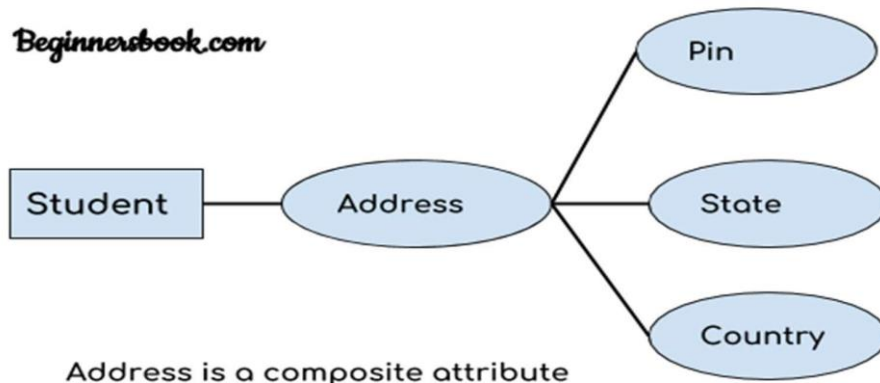
1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

1. Key attribute:

A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.



2. Composite attribute: An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.



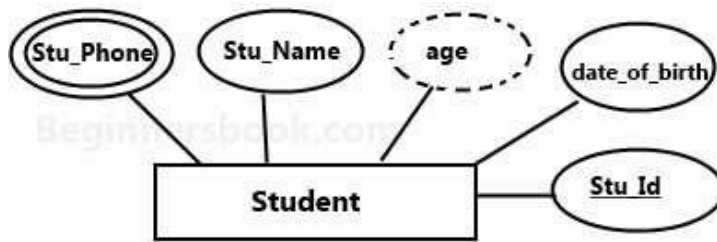
3. Multivalued attribute:

An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

4. Derived attribute:

A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

E-R diagram with multivalued and derived attributes:



3. Relationship

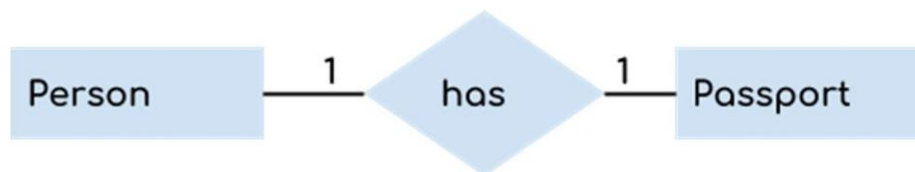
Cardinality: Defines the numerical attributes of the relationship between two entities or entity sets.

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of cardinal relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One Relationship

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.



2. One to Many Relationship

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.



3. Many to One Relationship

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.



4. Many to Many Relationship

When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.



Total Participation of an Entity set

A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. For example: In the below diagram each college must have at-least one associated Student.

CONCLUSION

The **ER Diagram** project has been instrumental in providing a clear and structured approach to designing a relational database. By identifying the key entities, their attributes, and relationships, we developed a logical model that accurately reflects the data requirements of a well-structured system. This diagram served as a valuable communication tool, allowing stakeholders to visualize the data architecture and ensuring alignment across the design process.

Moreover, the ER diagram played a critical role in minimizing data redundancy and ensuring efficient database performance. It provided a strong foundation for future implementation while allowing room for scalability and easy maintenance. Overall, this project demonstrates the power of the ER diagram as a fundamental step in designing efficient and flexible database systems.

References

1. R. Elmasri, S. Navathe, Fundamentals of Database Systems. 2nd ed., Benjamin/Cummings, Redwood City, CA., 1993.
2. Michael Kushner, Il-Yeol Song, and Kyu-Young Whang, "A Comparative Study of Three Object-Modeling Methodologies," Systems Development Management, 1994, 34-03-40 (1-22). (Also in Data Base Management, 26-01-10).
3. Janet Lind, Il-Yeol Song, and E.K. Park, "Object-Oriented Analysis: A Study in Diagram Notations," Journal of Computer and Software Engineering, Vol. 3, No. 1 (Winter 1995), pp. 133-165.
4. K. R. Dittrich, W. Gotthard, and P. Lockemann, "Complex Entities for Engineering Applications", in Proceedings of the International Conference on the ER Approach, NorthHolland, (1987), pp. 421-440.
5. C. Parent and S. Spaccapietra, "About entities, complex objects and object-oriented data models," in Information System Concepts: An In-depth Analysis, ED.. Falkenberg and P.Lindgreen (eds.), North-Holland, 1989, pp. 193-223.
6. P. P-S. Chen, "The entity-relationship model-toward a unified view of data," ACM Transactions on Database Systems, 1,1 (March 1976), pp. 9-36.
7. D. Reiner, M. Brodie, and G. Brown, et al. (eds.). "The Database design and evaluation workbench (DDEW) project at CCA." Database Engineering, 7,4 (1985).
8. T. J. Teorey, Database Modeling and Design: The Entity-Relationship Approach. Morgan Kauffmann, San Mateo, CA. 1991.
9. H. Korth and A. Silberschatz, Database System Concepts. 2nd ed., McGraw-Hill, New York, N.Y., 1991.

A FIELD PROJECT REPORT ON

ENTITY RELATIONSHIP DIAGRAM

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

YAYAVARAM NAGA

SUBRAHMANYA VISHAL (221FA20001)

NIKHIL

REDDY BHARGAV (221FA20015)

NADENDLA GOPICHANDU (221FA20026)

KAMINENI MANASA (221FA20030)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)

School of Computing & Informatics

Vignan's Foundation for Science, Technology and Research (Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April-2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Entity Relationship Diagram**” being submitted by YAYAVARAM NAGA SUBRAHMANYA VISHAL NIKHIL -221FA20001, REDDY BHARGAV-221FA20015, NADENDLA GOPICHANDU-221FA20026, KAMINENI MANASA-221FA20030 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Entity Relationship Diagram**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

YAYAVARAM	NAGA	
SUBRAHMANYA	VISHAL	(221FA20001)
NIKHIL		
REDDY BHARGAV		(221FA20015)
NADENDLA		(221FA20026)
GOPICHANDU		
KAMINENI MANASA		(221FA20030)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech
SEMESTER : I
DEPARTMET : CSE CSBS
SECTION : 2A
BATCH : 02

Submitted by:

YAYAVARAM NAGA
SUBRAHMANYA VISHAL (221FA20001)
NIKHIL
REDDY BHARGAV (221FA20015)
NADENDLA
GOPICHANDU (221FA20026)
KAMINENI MANASA (221FA20030)

TABLE OF CONTENTS

Name of the contents

1. Abstract.....	22
2. Introduction.....	23
3. Introduction to ER diagram.....	24
4. Main components of ER diagram.....	24
5. Entities.....	25-26
5.1.Strong entity.....	25
5.2.Weak entity.....	26
6. Attributes.....	26
7. Relationships.....	27-31
7.1.Degree of relationship.....	28
7.2.Mapping constraints.....	29
8. Converting ER model to relational model.....	31
9. Advantages of ER model.....	33
10. Disadvantages of ER model.....	33
11. Conclusion.....	34
12. References.....	35

Abstract

This paper explores the significance of Entity-Relationship (ER) Diagrams in database design, emphasizing their role in representing the structure and relationships between entities within a system. ER diagrams offer a visual approach to database modelling, simplifying the understanding and communication of database architectures among team members. The study delves into the key components of ER diagrams, including entities, attributes, and relationships, while also highlighting their application in various domains. By analysing their strengths and limitations, this research aims to provide insights into the best practices for creating effective ER diagrams, contributing to improved database design processes and better system performance.

Introduction

Entity-Relationship Diagrams (ER Diagrams) are a crucial part of database design, providing a visual representation of the relationships between various entities in a system. These diagrams help database designers, developers, and stakeholders to understand the structure of a database in a more intuitive and simplified manner. By representing entities as rectangles and relationships as diamonds, ER diagrams map out the interactions and associations within a system, ensuring that the database's structure is aligned with the actual needs of the organization or project.

ER diagrams also facilitate communication among team members, providing a common language to discuss database designs and requirements. This clear visualization helps in identifying potential issues such as redundant data, missing relationships, or incorrect associations early in the development process. As a foundational tool in conceptual data modelling, ER diagrams lay the groundwork for creating relational databases that efficiently store, retrieve, and manage data.

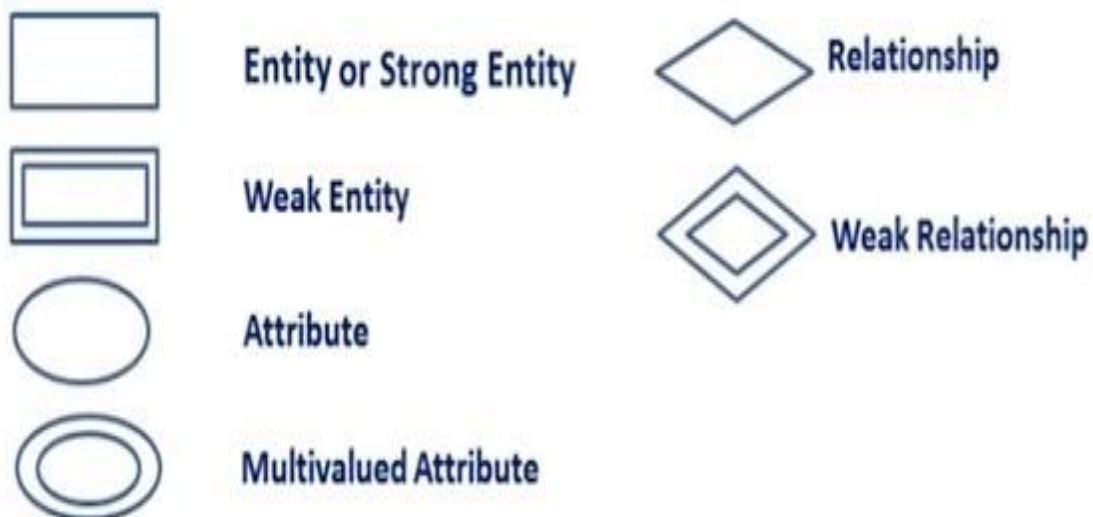
In modern organizational environments, efficient management of employee time cards is crucial for accurate payroll processing, performance tracking, and resource allocation. Traditional paper-based systems are often prone to errors, delays, and difficulties in data retrieval and analysis. To address these challenges, digitizing time card submissions and approvals using a robust database system is essential. This report details the design of such a system, focusing on capturing essential information about employees, managers, and time cards. By leveraging relational database principles, the system ensures data integrity, security, and scalability, thereby supporting the company's operational needs effectively.

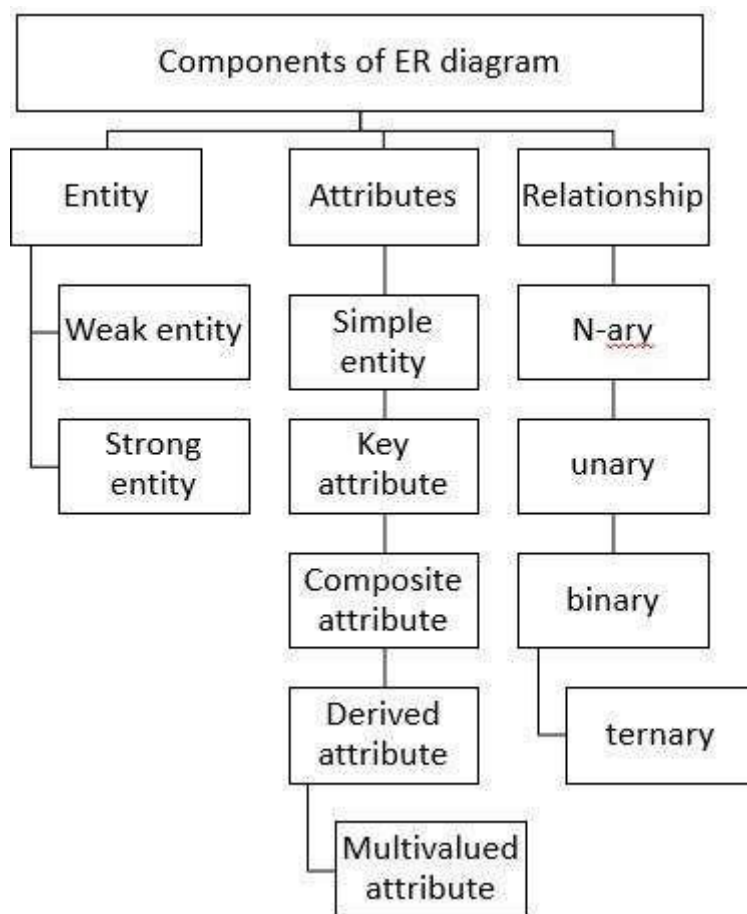
- ER model stands for an **Entity-Relationship** model. It is a high-level data model.
- The Entity Relationship model was proposed by **Peter Chen** in 1976.
- ER model is a logical representation of an enterprise data. ER model is a diagrammatic representation of logical structure of database.
- E-R model describes relationship among entities and attributes.
- Entity Relationship Diagrams are the best tools to communicate within the entire system.
- These diagrams are the graphical representation of the flow of data and information.
- These diagrams are most commonly used in business organizations to make data travel easy.
- This conceptual database model is an effective way of communicating with the individuals at all the levels.

The most common use of this diagram is to present the relation of the various tables present in a database.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes





1. Entity

It may be an object, person, place or event that stores data in a database. In E-R diagram an entity is represented in rectangle form. For example, students, employees, managers, etc.

The entity is pictorially depicted as follows:



Entity set

It is a collection of entities of the same type which share similar properties. For example, a group of students in a college and students are an entity set.

Entity is characterised into two types as follows:

- a. Strong entity set
- b. Weak entity set

- a. **Strong entity set:** The entity types which consist of key attributes or if there are enough attributes for forming a primary key attribute are called a strong entity set. It is represented by a single rectangle.

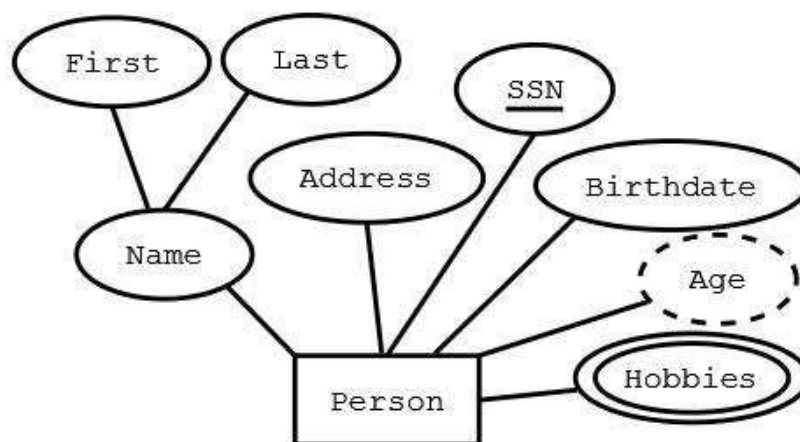


- b. **Weak entity set:** An entity does not have a primary key attribute and depends on another strong entity via foreign key attribute. It is represented by a double rectangle.



2. Attributes

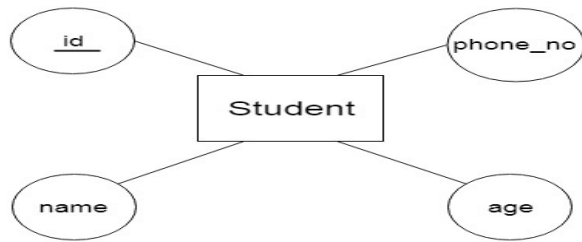
These are the data characteristics of entities or data elements and data fields.



Types of attributes

The types of attributes in the Entity Relationship (ER) model are as follows:

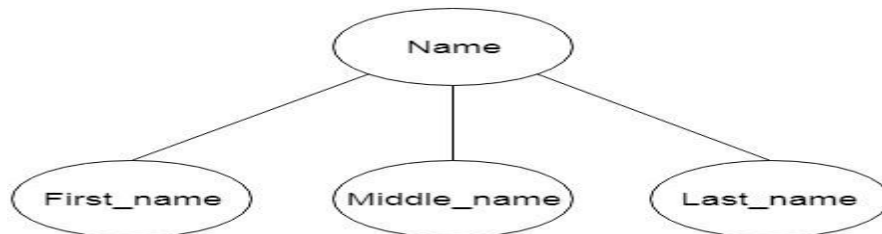
1. **Single value attribute** – these attributes contain a single value. For example, age, salary etc.
2. **Key Attributes-** The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



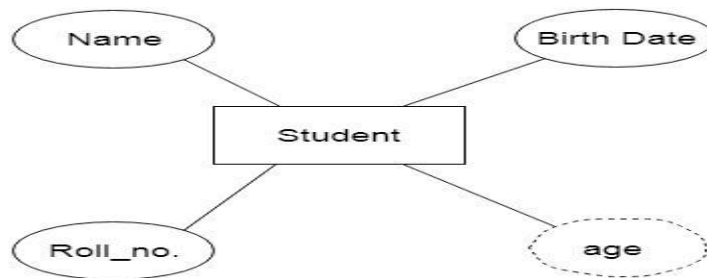
3. **Multivalued attribute** – they contain more than one value of a single entity. For example, phone numbers, Email_Ids, etc.



4. **Composite attribute** – the attributes which can be further divided. For example, **Name** consists of First name, Middle name, last name

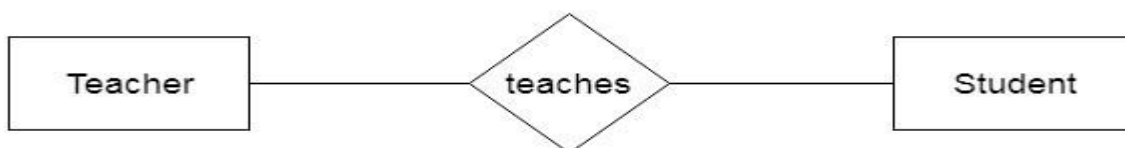


5. **Derived attribute** – the attribute that can be derived from other attributes. For example, age can be derived based on DoB.



3. Relationships

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

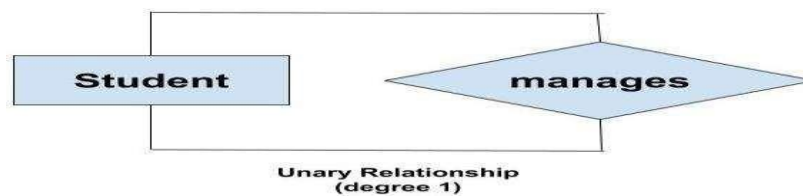


Degree of Relationship: A relationship where a number of different entities set participate is called a degree of a relationship.

It is categorised into the following:

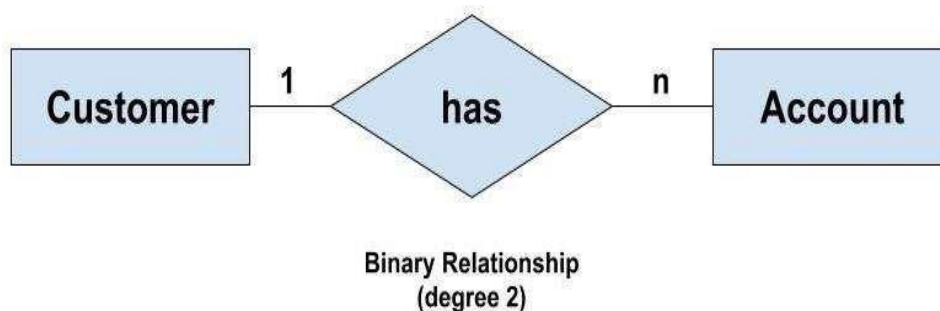
1. **Unary Relationship:** A unary relationship exists when both the participating entity type are the same. When such a relationship is present we say that the degree of relationship is 1.

For example, suppose in a classroom, we have many students who belong to a particular club-like dance club, basketball club etc. and some of them are club leads. So, a particular group of student is managed by their respective club lead. Here, the group is formed from students and also, the club leads are chosen from students. So, the ‘Student’ is the only entity participating here. We can represent this relationship using the E-R diagram as follows:



2. **Binary Relationship:** A binary relationship exists when exactly two entity type participates. When such a relationship is present we say that the degree is 2. This is the most common degree of relationship. It is easy to deal with such relationship as these can be easily converted into relational tables.

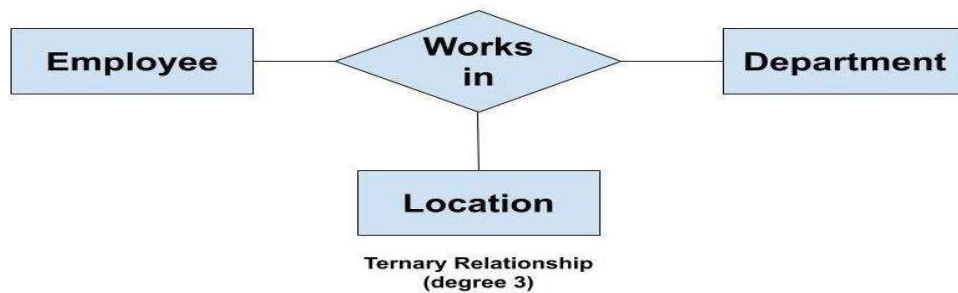
For example, we have two entity type ‘Customer’ and ‘Account’ where each ‘Customer’ has an ‘Account’ which stores the account details of the ‘Customer’. Since we have two entity types participating we call it a binary relationship. Also, one ‘Customer’ can have many ‘Account’ but each ‘Account’ should belong to only one ‘Customer’. We can say that it is a one-to-many binary relationship.



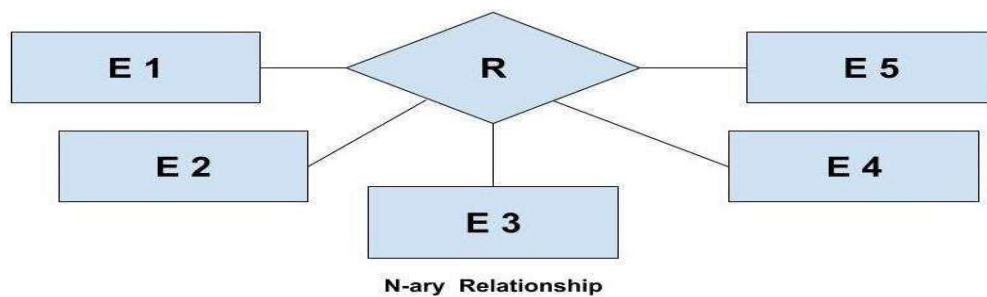
3. **Ternary Relationship:** A ternary relationship exists when exactly three entity type participates. When such a relationship is present we say that the degree is 3. As the number of entity increases in the relationship, it becomes complex to convert them into relational tables.

For example, we have three entity type ‘Employee’, ‘Department’ and ‘Location’. The relationship between these entities are defined as an employee works in a department,

an employee works at a particular location. So, we can see we have three entities participating in a relationship so it is a ternary relationship. The degree of this relation is 3.



4. **n-ary Relationship:** An N-ary relationship exists when 'n' number of entities are participating. So, any number of entities can participate in a relationship. There is no limitation to the maximum number of entities that can participate.



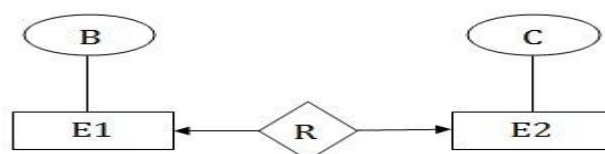
Mapping Constraints

- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.
- It is most useful in describing the relationship sets that involve more than two entity sets.

For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities.

- a. One-to-One (1:1)
- b. One-to-Many (1:M)
- c. Many-to-One (M:1)
- d. Many-to-Many (M:M)

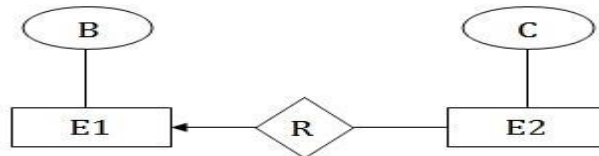
a. One-to-One Relationship: When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.



For example: A female can marry to one male, and a male can marry to one female.



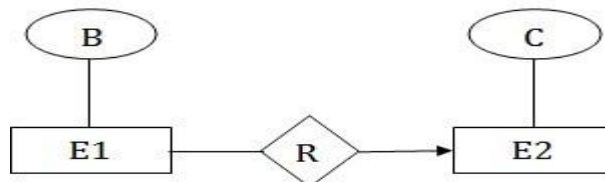
b. One-to-many relationship: When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.



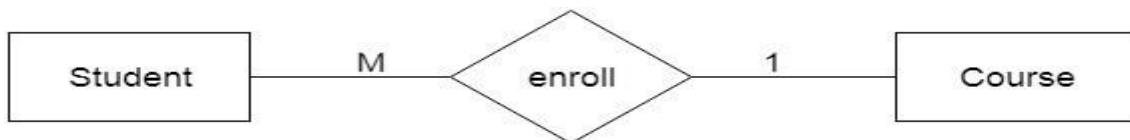
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



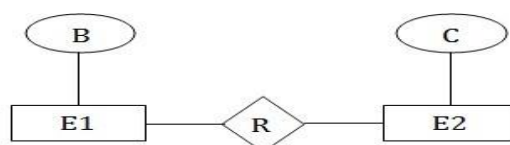
c. Many-to-one relationship: When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.



For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship: When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.



For example, Employee can assign by many projects and project can have many employees.



Comparison between Strong and Weak Entity Set

Strong Entity Set	Weak Entity Set
1. Strong entity set always has a primary key.	1. It does not have enough attributes to build a primary key.
2. It is represented by a rectangle symbol	2. It is represented by a double rectangle symbol.
3. It contains a Primary key represented by the underline symbol.	3. It contains a Partial Key which is represented by a dashed underline symbol.
4. The member of a strong entity set is called as dominant entity set.	4. The member of a weak entity set called as a subordinate entity set.
5. Primary Key is one of its attributes which helps to identify its member.	5. In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
6. In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	6. The relationship between one strong and a weak entity set shown by using the double diamond symbol.
7. The connecting line of the strong entity set with the relationship is single.	7. The line connecting the weak entity set for identifying relationship is double.

Converting E-R model into relational model

A given ER model can be converted into Relational model. A Relational model includes Relations, Tuples, Attributes, Keys, and Foreign keys.

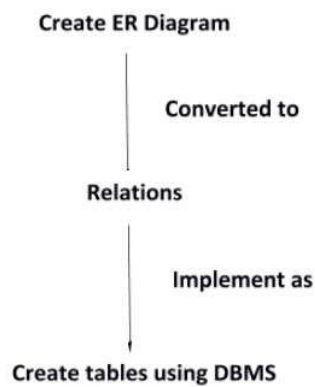
- Relation is a table made from tuples.
- A Tuple is a row of data.
- An Attribute is a characteristic of the relation.

There is a direct mapping between ER model and Relational model.

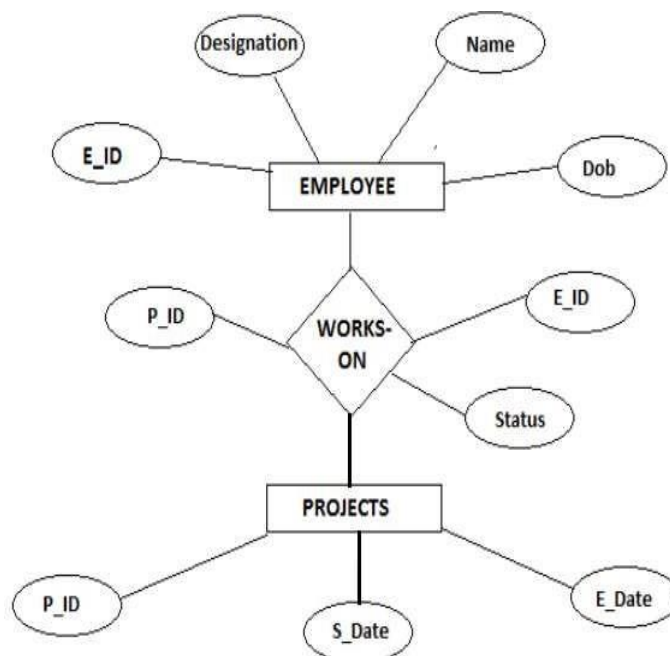
Rules of converting ER model to Relational Model:

- Entity type is converted to a Relation table.
- 1:1 or 1: N relationship type is converted to foreign key.
- M: N relationship type is converted to a relation with two foreign key.
- Simple attribute converted to an attribute.
- Value set converted to a domain.
- Key attribute converted to a primary key.

Overall transformation summary is as follows:



Consider the following **example**:



Now for the above example we can create three relations:

- Employee
- Works_On
- Projects

Transform attributes to fields:

- Employee will have E_ID, Name, Designation and Dob.
- Works_On will have E_ID, Status and P_ID.
- Projects will have P_ID, S_Date and E_Date.

Now we can create tables in DBMS.

Advantages of E-R Model

1. Conceptually E-R model is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.

2. Better Visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.

3. Effective communication tool: It is an effective communication tool for database designer.

The clear representation of the data listed under proper headings and tables results in the effective flow of information and communication.

4. Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.

5. Easy conversion to any data: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

6. Straightforward relation representation: Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straightforward.

Disadvantages of E-R Model

1. Limited constraints and specification: The constraints and specifications are limited.

2. Loss of information content: Some information be lost or hidden in ER model.

3. Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.

4. No representation of data manipulation: It is difficult to show data manipulation in ER model.

5. No industry standard for notation.

CONCLUSION

In conclusion, Entity-Relationship (ER) diagrams are a valuable tool for visually representing and understanding the structure of databases. By modeling entities, attributes, and relationships, ER diagrams provide a clear and concise blueprint for database design. They are particularly useful in the early stages of database development, facilitating communication between stakeholders and ensuring that the database accurately captures the information requirements of the system.

ER diagrams offer several advantages, including their ability to capture complex relationships, promote understanding among team members, and serve as a foundation for database implementation. However, it is important to note that ER diagrams have limitations, such as their potential for becoming overly complex and their difficulty in representing certain types of data. Nevertheless, when used effectively, ER diagrams can be a powerful asset in database design and development.

REFERENCES

Books:

- **Conceptual Database Design: An Entity-Relationship Approach** by Carlo Batini, Sergio Ceri, and Shamkant B. Navathe: A classic textbook that provides a comprehensive overview of ER modeling and database design.
- **Database Systems: Design, Implementation, and Management** by Elmasri and Navathe: A widely used textbook that covers ER diagrams as part of database design principles.
- **Data Modeling Techniques: A Practical Guide to Relational and Object-Oriented Database Design** by David Barker: A practical guide to ER modeling and other data modeling techniques.

Online Resources:

- **Vertabelo:** An online database modeling tool that offers tutorials and resources on ER diagrams, including notation, best practices, and examples.
- **Lucidchart:** Another popular online diagramming tool that provides templates and guides for creating ER diagrams.
- **GeeksforGeeks:** A technical blog with articles on various database concepts, including ER diagrams and their applications.
- **Tutorials Point:** A website offering tutorials and courses on database management systems, including ER modeling.

A FIELD PROJECT REPORT ON

FUNCTIONAL DEPENDENCIES AND NORMAL FORMS

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

KOLA VARUN KUMAR REDDY	(221FA20016)
MALINENI SRIHARI	(221FA20014)
CHITIMIREDDY SAIKUMAR REDDY	(221FA20025)
BANDARU LAKSHMI CHARITHA	(221FA20005)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)
School of Computing & Informatics
Vignan's Foundation for Science, Technology and Research (Deemed to be University)
Vadlamudi, Guntur, Andhra Pradesh-522213, India
April-2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Functional dependencies and Normal forms**” being submitted KOLA VARUN KUMAR REDDY-221FA20016, MALINENI SRIHARI-221FA20014, CHITIMIREDDY SAIKUMAR REDDY-221FA20025, BANDARU LAKSHMI CHARITHA-221FA20005 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans’ Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Functional dependencies and Normal forms**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

KOLA VARUN KUMAR REDDY	(221FA20016)
MALINENI SRIHARI	(221FA20014)
CHITIMIREDDY SAIKUMAR REDDY	(221FA20025)
BANDARU LAKSHMI CHARITHA	(221FA20005)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech
SEMESTER : I
DEPARTMET : CSE CSBS
SECTION : 2A
BATCH : 03

Submitted by:

KOLA VARUN KUMAR REDDY	(221FA20016)
MALINENI SRIHARI	(221FA20014)
CHITIMIREDDY SAIKUMAR REDDY	(221FA20025)
BANDARU LAKSHMI CHARITHA	(221FA20005)

TABLE OF CONTENTS

Name of the contents

1. Abstract.....	41
2. Introduction.....	42
3. Introduction to functional dependencies.....	43-44
3.1. Types of functional dependencies.....	43
4. Anomalies in DBMS.....	45
5. Normalization.....	45-49
5.1 First Normal Form.....	46
5.2 Second Normal Form.....	47
5.3 Third Normal Form.....	48
6. Conclusion.....	50
7. References.....	51

Abstract

This study investigates the role of functional dependencies in relational database design, focusing on their importance in ensuring data integrity and reducing redundancy. Functional dependencies describe the relationship between attributes within a relation, offering a framework for organizing data in an optimal manner. The research highlights the impact of FDs on the normalization process, which transforms unstructured tables into well-formed relations. Through a detailed analysis of functional dependencies and their use in database normalization, this paper provides practical insights into optimizing database design and preventing common anomalies like update, insert, and delete anomalies.

Introduction

Functional dependencies (FDs) play a fundamental role in the design and optimization of relational databases. A functional dependency occurs when the value of one attribute (or set of attributes) in a relation uniquely determines the value of another attribute. Understanding these dependencies is essential in ensuring database normalization, which reduces data redundancy and enhances data integrity. In essence, functional dependencies form the backbone of how relationships are structured in a relational model, making them a key concept for anyone involved in database management.

By identifying and analysing functional dependencies, database designers can break down complex tables into smaller, more manageable ones, ensuring that each relation is in its optimal form. This process helps prevent anomalies during data insertion, deletion, and updating, and results in a more efficient and reliable database system. Functional dependencies thus provide a blueprint for creating databases that perform well and scale easily, with a consistent and structured representation of data.

In relational database management systems (RDBMS), the organization and integrity of data are paramount. Functional dependencies serve as fundamental constraints that define how attributes within a table relate to one another. Understanding and accurately modeling these dependencies are critical for effective database normalization—a systematic process aimed at reducing data redundancy and enhancing data integrity.

Functional Dependencies

The attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.

For example: Suppose we have a student table with attributes: Stu_Id, Stu_Name, Stu_Age. Here Stu_Id attribute uniquely identifies the Stu_Name attribute of student table because if we know the student id we can tell the student name associated with it. This is known as functional dependency and can be written as $\text{Stu_Id} \rightarrow \text{Stu_Name}$ or in words we can say Stu_Name is functionally dependent on Stu_Id.

Formally:

If column A of a table uniquely identifies the column B of same table then it can represented as $A \rightarrow B$ (Attribute B is functionally dependent on attribute A)

Types of Functional Dependencies

- Trivial functional dependency
- non-trivial functional dependency
- Multivalued dependency
- Transitive dependency

Trivial functional dependency

The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.

Symbolically: $A \rightarrow B$ is trivial functional dependency if B is a subset of A.

The following dependencies are also trivial: $A \rightarrow A$ & $B \rightarrow B$

For example: Consider a table with two columns Student_id and Student_Name.

$\{\text{Student_Id}, \text{Student_Name}\} \rightarrow \text{Student_Id}$ is a trivial functional dependency as Student_Id is a subset of $\{\text{Student_Id}, \text{Student_Name}\}$. That makes sense because if we know the values of Student_Id and Student_Name then the value of Student_Id can be uniquely determined.

Also, $\text{Student_Id} \rightarrow \text{Student_Id}$ & $\text{Student_Name} \rightarrow \text{Student_Name}$ are trivial dependencies too.

Non-Trivial functional dependency

If a functional dependency $X \rightarrow Y$ holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

For example:

An employee table with three attributes: emp_id, emp_name, emp_address.

The following functional dependencies are non-trivial:

emp_id → emp_name (emp_name is not a subset of emp_id)

emp_id → emp_address (emp_address is not a subset of emp_id)

On the other hand, the following dependencies are trivial:

{emp_id, emp_name} → emp_name [emp_name is a subset of {emp_id, emp_name}]

Completely non trivial FD:

If a FD X→Y holds true where X intersection Y is null then this dependency is said to be completely non trivial function dependency.

Multivalued dependency

Multivalued dependency occurs when there are more than one **independent** multivalued attributes in a table.

For example: Consider a bike manufacture company, which produces two colors (Black and white) in each model every year.

bike_model	manuf_year	color
M1001	2007	Black
M1001	2007	Red
M2012	2008	Black
M2012	2008	Red
M2222	2009	Black
M2222	2009	Red

Here columns manuf_year and color are independent of each other and dependent on bike_model. In this case these two columns are said to be multivalued dependent on bike_model. These dependencies can be represented like this:

bike_model →→ manuf_year

bike_model →→ color

➤ Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let's discuss about anomalies first then we will discuss normal forms with examples.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let's take an example to understand this.

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete anomaly: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

Normalization

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp name	emp_address	emp mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF** as the rule says “each attribute of a table must have atomic (single) values”, the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

emp_id	emp name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

Table is in 1NF (First normal form)

No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher id	Subject	teacher age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher id	teacher age
111	38
222	38
333	40

teacher_subject table:

teacher_id	Subject
------------	---------

111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}...so on

Candidate Keys: {emp_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp state	emp city	emp district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

CONCLUSION

In this project, the concepts of functional dependencies and normal forms played a crucial role in optimizing the database design. By understanding how certain attributes in a table depend on others, we were able to identify functional dependencies that ensured consistency in the data. This understanding allowed us to decompose tables effectively and eliminate anomalies, such as redundancy and update issues.

Through the process of normalization, we structured the database into higher normal forms (1NF, 2NF, 3NF, and beyond), ensuring that the design minimized redundancy and enhanced data integrity. This not only improved query performance but also made future database maintenance more efficient. Overall, the application of functional dependencies and normalization was key to developing a well-organized, efficient, and scalable database.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.
3. Connolly, T. M., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
4. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
6. Korth, H. F., & Silberschatz, A. (2002). *Database System Concepts*. McGraw-Hill.
7. O'Neil, P., & O'Neil, E. (2009). *Database: Principles, Programming, and Performance*. Morgan Kaufmann.
8. Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
9. Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, 1(1), 9-36.
10. Firebird Project. (n.d.). "Firebird SQL." Retrieved from <https://firebirdsql.org/>
11. MySQL. (n.d.). "MySQL Documentation." Retrieved from <https://dev.mysql.com/doc/>
12. PostgreSQL. (n.d.). "PostgreSQL Documentation." Retrieved from <https://www.postgresql.org/docs/>
13. Ambler, S. W. (2003). *The Object Primer: Agile Model-Driven Development*. Cambridge University Press.

A FIELD PROJECT REPORT ON

FUNCTIONAL DEPENDENCIES AND RELATIONAL SCHEMA

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

YEDURU AARADHYA (221FA20020)

RAVELA JYOTHI KAMBIKA (221FA20022)

CHINTALA LOKESH (221FA20029)

INTURI GAYATHRI (221FA20034)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)

School of Computing & Informatics

Vignans's Foundation for Science, Technology and Research (Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April-2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Functional dependencies and Relational Schema**” being submitted by YEDURU AARADHYA-221FA20020, RAVELA JYOTHI KAMBIKA-221FA20022, CHINTALA LOKESH-221FA20029, INTURI GAYATHRI-221FA20034 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Functional dependencies and Relational Schema**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

YEDURU AARADHYA (221FA20020)

RAVELA JYOTHI KAMBIKA (221FA20022)

CHINTALA LOKESH (221FA20029)

INTURI GAYATHRI (221FA20034)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech
SEMESTER : I
DEPARTMET : CSE CSBS
SECTION : 2A
BATCH : 04

Submitted by:

YEDURU AARADHYA	(221FA20020)
RAVELA JYOTHI KAMBIKA	(221FA20022)
CHINTALA LOKESH	(221FA20029)
INTURI GAYATHRI	(221FA20034)

TABLE OF CONTENTS

Name of contents

1. Abstract.....	57
2. Introduction.....	58
3. Definition of Functional Dependencies.....	59-60
3.1.Types of functional dependencies.....	59
4. Decomposition.....	60-63
4.1.Properties of decomposition.....	60
5. Anamoly.....	64
6. Normalization.....	64
7. Conclusion.....	65
8. References.....	66

Abstract

This paper explores the relationship between functional dependencies and the design of relational schemas in databases. Functional dependencies define how attributes in a relation are connected, and they play a pivotal role in database normalization, where large, complex tables are decomposed into smaller, more efficient ones. The study highlights the importance of identifying functional dependencies in reducing data redundancy and avoiding update, insert, and delete anomalies. By analysing their role in shaping relational schemas, the paper provides a comprehensive understanding of how functional dependencies contribute to the creation of robust, optimized database designs.

Introduction

Functional dependencies (FDs) are an essential concept in relational database theory, defining how attributes within a relation are related. They serve as rules that govern the relationships between different attributes, where one attribute (or a combination of attributes) can uniquely determine another. Understanding these dependencies is crucial when designing a relational schema because they influence how data is structured, stored, and accessed. Without properly identifying functional dependencies, databases risk being inefficient, prone to anomalies, and difficult to maintain.

Relational schemas, which describe the organization of data into tables (relations), are directly shaped by functional dependencies. They form the framework for database normalization, a process used to eliminate redundancy and ensure data integrity. Through normalization, complex relations are broken down into smaller, well-organized tables based on the functional dependencies that exist among their attributes. A well-designed relational schema reduces redundancy, prevents anomalies, and ensures that the database operates efficiently, making functional dependencies a foundational element in database design.

Normalization involves decomposing a table into smaller, related tables without loss of data, ensuring that each table adheres to specific normal forms. This process not only streamlines data storage but also optimizes query performance and simplifies maintenance. This report delves into the application of functional dependencies and normalization techniques on the `DISK_DRIVE` relation, demonstrating the transition from an unnormalized schema to a fully normalized database structure.

Functional Dependencies

- A functional dependence is a **constraint between two sets of attributes** from the database.
- It plays major role in **differentiating** good database design from bad database design.
- Functional dependency is a type of constraint that is generalization of notion of key.
- Bad database design
 - Repetition of information
 - Inability to represent certain information
- Good database
 - Avoid redundant data
 - Ability to represent all information and relationship among attributes.

Definition

- A function dependency denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subset of R specifies a constraints on the possible tuples that can form a relation state r of R.
- The value of component **Y depends on values of component X**
- Abbreviation for functional dependencies is **FD or f.d.**
- The set of attributes X is called the left-hand side of the FD, and Y is called the right-hand side.

Example

- Suppose we have a student table with attributes: Stu_Id, Stu_Name, Stu_Age.
- Here **Stu_Id attribute uniquely identifies the Stu_Name** attribute of student table because if we know the student id we can tell the student name associated with it.
- This is known as functional dependency.
- Can be written as **Stu_Id->Stu_Name** or in words we can say Stu_Name is functionally dependent on Stu_Id.

Types of Functional Dependency

- Functional Dependency has three forms:
 - Trivial Functional Dependency
 - Non-Trivial Functional Dependency

Trivial Functional Dependency

It occurs when B is a subset of A in:

$$A \rightarrow B$$

Example

We are considering the same <Department> table with two attributes to understand the concept of trivial dependency.

The following is a trivial functional dependency since DeptId is a subset of DeptId and DeptName

$$\{ \text{DeptId, DeptName} \} \rightarrow \text{Dept Id}$$

Non –Trivial Functional Dependency

It occurs when B is **not a subset** of A in:

$$A \rightarrow B$$

Example

$$\text{DeptId} \rightarrow \text{DeptName}$$

The above is a non-trivial functional dependency since DeptName is not a subset of DeptId.

An employee table with three attributes: emp_id, emp_name, emp_address. The following functional dependencies are non-trivial:

emp_id → emp_name (emp_name is not a subset of emp_id) emp_id → emp_address
(emp_address is not a subset of emp_id)

Non-loss Decomposition

What is decomposition?

- Decomposition is the process of **breaking down in parts or elements**.
- It replaces a relation with a collection of smaller relations.
- It breaks the **table into multiple tables in a database**.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition

Following are the properties of Decomposition,

1. Lossless Decomposition
2. Dependency Preservation
3. Lack of Data Redundancy

1. Lossless Decomposition

- Decomposition must be lossless. It means that the **information should not get lost from the relation** that is decomposed.
- It gives a guarantee that the join will result in the same relation as it was decomposed.
 - Let R be a relation schema and let F be a set of functional dependencies on R.
 - Let 'R1' & 'R2' form a decomposition of R.
 - Let r(R) be a relation with schema R.
 - Decomposition is a losses decomposition, if for legal database instance $\prod_{R1}(\mathbf{r}) \bowtie \prod_{R2}(\mathbf{r}) = \mathbf{r}$
 - If user project r onto R1 & R2, and compute the natural join of the projection results exactly 'r'. Hence no loss, non-loss decomposition.

Example:

Let's take 'E' is the Relational Schema, With instance 'e'; is decomposed into: E1, E2, E3, . . . En; With instance: e1, e2, e3, en, If $e1 \bowtie e2 \bowtie e3 \dots \dots \dots \bowtie en$, then it is called as

'Lossless Join Decomposition'.

- In the above example, it means that, if **natural joins of all the decomposition give the original relation, then it is said to be lossless join decomposition. Example:**

<Employee_Department> Table

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

- Decompose the above relation into two relations to check whether a decomposition is lossless or lossy.
- Now, decompose the relation that is Employee and Department.

Relation 1 : <Employee> Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

Employee Schema contains (Eid, Ename, Age, City, Salary).

Relation 2 : <Department> Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production

D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

Department Schema contains (Deptid, Eid, DeptName).

- So, the above decomposition is a Lossless Join Decomposition, because the two relations contains one common field that is 'Eid' and therefore join is possible.
- Now apply natural join on the decomposed relations.

Employee ⋈ Department

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

Hence, the decomposition is Lossless Join Decomposition.

- If the <Employee> table contains (Eid, Ename, Age, City, Salary) and <Department> table contains (Deptid and DeptName), then it is not possible to join the two tables or relations, because there is no common column between them. And it becomes **Lossy Join Decomposition**.

2. Dependency Preservation

- Dependency is an important constraint on the database.
- Every dependency **must be satisfied by at least one decomposed table**.
- If $\{A \rightarrow B\}$ holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in a same relation.
- This decomposition property can only be **done by maintaining the functional dependency**.
- In this property, it allows to check the updates without computing the natural join of the database structure.
- Set of restriction $F_1, F_2, F_3 \dots F_n$ is the set of dependencies that can be checked efficiently.
 - Let, $F' = F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n$.
 - F' is a set of functional dependencies on schema R but in general $F' \neq F$.
 - The property $F'^+ = F^+$ is a dependency preserving decomposition.
 - $F = \{A \rightarrow B, B \rightarrow C\}$. F^+ is dependency $A \rightarrow C$ even though it is not in F.

1. Lack of Data Redundancy

- Lack of Data Redundancy is also known as a **Repetition of Information**.
- The proper decomposition should not suffer from any data redundancy.
- The careless decomposition may cause a problem with the data.
- The lack of data redundancy property may be achieved by Normalization process.

Normalization

- **Normalization** is a process of organizing the data in database to avoid
 - Data Redundancy
 - Insertion anomaly
 - Deletion anomaly.
 - Update anomaly &
- Normalization divides the **larger table into the smaller table** and links them using relationship.
- The normal form is used to reduce redundancy from the database table.
- **Normalization** is the process of **minimizing redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updating anomalies.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies.

Anomalies

- **Insert anomalies** – When user tried to insert data in a record that does not exist at all.
- **Deletion anomalies** – When user tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else. When user tried to delete a record, which cause deletion of some other data from the table/realation.
- **Update anomalies** – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Example:** Suppose a manufacturing company stores the employee details in a table named employee that has four attributes:
 - emp_id for storing employee's id,
 - emp_name for storing employee's name,
 - emp_address for storing employee's address and
 - emp_dept for storing the department details in which the employee works.

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

- The above table is not normalized. We will see the problems that we face when a table is not normalized.
- **Insert anomaly:** Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.
- **Delete anomaly:** Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.
- **Update anomaly:** In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.
- To overcome these anomalies, we need to normalize the data.

CONCLUSION

In this project, the exploration of **functional dependencies** and their impact on the **relational schema** was essential in creating an efficient database design. By identifying and analysing functional dependencies between attributes, we gained a deeper understanding of how data elements are interconnected. This process allowed us to structure the relational schema in a way that minimized redundancy and eliminated potential update, insertion, and deletion anomalies.

The careful application of these functional dependencies led to a well-normalized relational schema, ensuring that the data is both consistent and easily maintainable. This structured approach provides a foundation for scalable, efficient, and high-performing database systems. Ultimately, this project highlights the importance of leveraging functional dependencies to refine and optimize the relational schema for long-term usability and data integrity.

REFERENCES

1. **Korth, H. F., & Sudarshan, S.** (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
2. **Ramakrishnan, R., & Gehrke, J.** (2013). *Database Management Systems* (3rd ed.). McGraw-Hill Education.
3. **Date, C. J.** (2004). *An Introduction to Database Systems* (8th ed.). Addison-Wesley.
4. **Elmasri, R., & Navathe, S. B.** (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.
5. **Rob, P., & Coronel, C.** (2017). *Database Systems: Design, Implementation, & Management* (13th ed.). Cengage Learning.
6. **Connolly, T. M., & Begg, C.** (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.
7. **Silberschatz, A., Korth, H. F., & Sudarshan, S.** (2011). *Database System Concepts* (6th ed.). McGraw-Hill.
8. **Gonzalez, A. J.** (2016). "A Comparative Study of Library Management Systems." *International Journal of Computer Applications*, 140(3), 22-28.
9. **Zhang, Y., & Yao, W.** (2015). "A Cloud-Based Library Management System." *International Journal of Cloud Computing and Services Science*, 4(2), 123-134.
10. **Chowdhury, G. G.** (2004). *Introduction to Digital Libraries*. Facet Publishing.
11. **Ranganathan, S. R.** (1931). *Colon Classification*. Asia Publishing House.
12. **Bates, M. J.** (2005). "An Introduction to Metadata." *Digital Libraries: Technology and Management*, 4-8.
13. **W3Schools.** (n.d.). "SQL Tutorial." Retrieved from W3Schools SQL
14. **Oracle.** (n.d.). "Oracle Database Documentation." Retrieved from [Oracle Docs](#)
15. **MySQL Documentation.** (n.d.). "MySQL Reference Manual." Retrieved from [MySQL Docs](#)

A FIELD PROJECT REPORT ON

FUNCTIONAL DEPENDENCIES

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

CHINNAPAREDDY	(221FA20006)
GOWTHAM REDDY	
GADE LOKESH	(221FA20010)
NARAYANAM LASYA	(221FA20018)
TADIBOINA SAIDABABU	(221FA20023)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)
School of Computing & Informatics
Vignan's Foundation for Science, Technology and Research (Deemed to be University)
Vadlamudi, Guntur, Andhra Pradesh-522213, India
April-2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Functional dependencies**” being submitted by CHINNAPAREDDY GOWTHAM REDDY- 221FA20006, GADE LOKESH-221FA20010, NARAYANAM LASYA - 221FA20018, TADIBOINA SAIDABABU - 221FA20023 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Functional dependencies**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

CHINNAPAREDDY GOWTHAM REDDY	(221FA20006)
GADE LOKESH	(221FA20010)
NARAYANAM LASYA	(221FA20018)
TADIBOINA SAIDABABU	(221FA20023)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech
SEMESTER : I
DEPARTMET : CSE CSBS
SECTION : 2A
BATCH : 05

Submitted by:

CHINNAPAREDDY GOWTHAM REDDY	(221FA20006)
GADE LOKESH	(221FA20010)
NARAYANAM LASYA	(221FA20018)
TADIBOINA SAIDABABU	(221FA20023)

TABLE OF CONTENTS

Name of contents

1. Abstract.....	72
2. Introduction.....	73
3. Functional Dependency.....	74
4. Example for functional dependency.....	74
5. Rules for functional dependency.....	75
6. Conclusion.....	76
7. References.....	77

Abstract

This paper examines the concept of functional dependencies (FDs) and their importance in relational database design. Functional dependencies define the relationship between attributes, where one attribute uniquely determines another. These dependencies are essential for database normalization, a process that reduces redundancy and prevents data anomalies by organizing data into well-structured relations. The study highlights the role of FDs in maintaining data integrity and their influence on the design of efficient, scalable database systems. By understanding and applying functional dependencies, database designers can create relational schemas that enhance performance and ensure data consistency.

Introduction

Functional dependencies (FDs) are a fundamental concept in relational database design, describing the relationship between attributes within a database table (or relation). A functional dependency exists when one attribute (or a set of attributes) uniquely determines another attribute. For instance, in a table of employees, the employee ID can uniquely determine the employee's name, making it a functional dependency. This concept plays a key role in maintaining the integrity and consistency of data, particularly during the process of database normalization. Without understanding and managing functional dependencies, databases may suffer from redundancy, inefficiency, and various data anomalies.

Functional dependencies are not only theoretical but practical tools that help database designers define how data should be organized. By analysing FDs, designers can identify opportunities to break down large tables into smaller, more focused ones that avoid data redundancy. This process—called normalization—leads to the creation of relations that are easier to maintain, update, and scale. FDs ensure that the database adheres to the principles of integrity and performance, making them critical for developing reliable relational databases. In relational database design, understanding functional dependencies is crucial for ensuring data integrity and optimizing storage. Functional dependencies dictate how attributes relate to one another, influencing the normalization process that minimizes redundancy and prevents anomalies. This report examines a specific set of functional dependencies within a schema $R(A, B, C, D, E, F)$ and addresses several key questions related to attribute closures, super key identification, canonical cover computation, and normalization into Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF). By systematically answering these questions, the report provides a comprehensive analysis of the schema's structure and offers insights into effective database design practices.

Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system. Functional dependency helps you to maintain the quality of data in the database. A functional dependency is denoted by an arrow \rightarrow . The functional dependency of X on Y is represented by $X \rightarrow Y$. Functional Dependency plays a vital role to find the difference between good and bad database design.

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

- Key terms
- Rules of Functional Dependencies
- Types of Functional Dependencies
- Multivalued dependency in DBMS
- Trivial Functional dependency
- Non trivial functional dependency in DBMS
- Transitive dependency
- What is Normalization?
- Advantages of Functional Dependency

Axiom Axioms is a set of inference rules used to infer all the functional dependencies on a relational database.

Decomposition It is a rule that suggests if you have a table that appears to contain two entities which are determined by the same primary key then you should consider breaking them up into two different tables.

Dependent It is displayed on the right side of the functional dependency diagram.

Determinant It is displayed on the left side of the functional dependency Diagram.

Union It suggests that if two tables are separate, and the PK is the same, you should consider putting them together.

Rules of Functional Dependencies

Below given are the Three most important rules for Functional Dependency:

- Reflexive rule –. If X is a set of attributes and Y is_subset_of X, then X holds a value of Y.
- Augmentation rule: When $x \rightarrow y$ holds, and c is attribute set, then $ac \rightarrow bc$ also holds. That is adding attributes which do not change the basic dependencies.
- Transitivity rule: This rule is very much similar to the transitive rule in algebra if $x \rightarrow y$ holds and $y \rightarrow z$ holds, then $x \rightarrow z$ also holds. $X \rightarrow y$ is called as functionally that determines y.

CONCLUSION

In this project, functional dependencies played a critical role in shaping the structure and integrity of the database. By analyzing how certain attributes in a relation are determined by others, we were able to identify functional dependencies that guided the design of an efficient and organized schema. This process helped us ensure data consistency, as well as reduce redundancy and potential anomalies like data duplication or inconsistency.

The identification and proper management of functional dependencies also allowed us to apply normalization techniques, which contributed to the overall improvement of the database's performance and scalability. In conclusion, understanding and leveraging functional dependencies proved to be a vital step in optimizing the relational database design, ensuring both data integrity and efficient operation.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems*.
2. Date, C. J. (2004). *An Introduction to Database Systems*.
3. Rob, P., & Coronel, C. (2015). *Database Systems: Design, Implementation, & Management*.
4. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database System Concepts*.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*.
6. Hoffer, J. A., Venkataraman, R., & Upadhyaya, J. (2016). *Modern Database Management*.
7. Batini, C., & Scannapieco, M. (2016). *Data Quality: Concepts, Methodologies, and Techniques*.
8. Ponniah, P. (2010). *Database Fundamentals: A Complete Introduction to Database Design*.
9. Connolly, T. M., & Begg, C. E. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*.
10. Chandrasekaran, M., & Nair, A. (2021). *Database Management Systems*.
11. Atzeni, P., & Torlone, R. (2015). *Database Systems*.
12. Thomas, J. (2014). *SQL for Data Analytics*.
13. Danylo, O., & Zakharchuk, O. (2017). *Database Management with MySQL*.
14. Rob, P., & Coronel, C. (2016). *Database Design Using Entity-Relationship Diagrams*.
15. Chen, P. P. (1976). *The Entity-Relationship Model: Toward a Unified View of Data*.
16. Schreiber, R. (2019). *Database Systems: Concepts, Languages, Architectures*.
17. Wong, K. C. (2018). *An Introduction to Database Design*.
18. Finkelstein, A. (2017). *Database Systems: A Practical Approach to Design, Implementation, and Management*.
19. Klug, A. (2015). *Database Management Systems*.
20. Melton, J., & Simon, A. R. (2002). *SQL: 1999*.

A FIELD PROJECT REPORT ON

ENTITY RELATIONSHIP DIAGRAM

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

NEELISETTY LAKSHMI SATWIKA	(221FA20002)
DIVI SUKHESH BABU	(221FA20008)
BONTHU NAGA PREETHIKA REDDY	(221FA20021)
SHAIK ASIM SAYEED	(221FA20033)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)
School of Computing & Informatics
Vignans' Foundation for Science, Technology and Research (Deemed to be University)
Vadlamudi, Guntur, Andhra Pradesh-522213, India
April-2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

Act No. 3 of 1987 Act 1988

CERTIFICATE

This is to certify that the field project/IDP entitled "Entity Relationship Diagram" being submitted by NEELISETTY LAKSHMI SATWIKA-221FA20002, DIVI SUKHESH BABU-221FA20008, BONTHU NAGA PREETHIKA REDDY-221FA20021, SHAIK ASIM SAYEED-221FA20033 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Entity Relationship Diagram**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

NEELISETTY LAKSHMI SATWIKA	(221FA20002)
DIVI SUKHESH BABU	(221FA20008)
BONTHU NAGA PREETHIKA REDDY	(221FA20021)
SHAIK ASIM SAYEED	(221FA20033)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech
SEMESTER : I
DEPARTMET : CSE CSBS
SECTION : 2A
BATCH : 06

Submitted by:

NEELISETTY LAKSHMI SATWIKA (221FA20002)
DIVI SUKHESH BABU (221FA20008)
BONTHU NAGA PREETHIKA REDDY (221FA20021)
SHAIK ASIM SAYEED (221FA20033)

TABLE OF CONTENTS

Name of contents

1. Abstract.....	83
2. Introduction.....	84
3. ER Diagrams.....	85
4. ER Models in Database Design.....	86
5. Elements of ER Diagram.....	87-89
5.1.Entity.....	87
5.2.Attributes.....	88
5.3.Relationships.....	89
6. How to draw ER diagrams.....	90
7. Benefits of ER diagram.....	90
8. Conclusion	91
9. References	92

Abstract

This paper explores the significance of Entity-Relationship Diagrams (ERDs) in database design, focusing on their role in representing entities, attributes, and relationships in a structured format. ERDs provide a conceptual model for understanding the relationships between data elements in a system, helping stakeholders visualize and communicate complex database requirements. By analysing the key components of ERDs and their application in various industries, this study highlights how ERDs contribute to the development of efficient, well-structured databases. The research also discusses how ERDs aid in identifying potential design issues early in the development process, thus ensuring data integrity and system scalability

Introduction

An Entity-Relationship Diagram (ERD) is a visual representation of the relationships and interactions between entities within a database system. ERDs are crucial tools in the database design process as they provide a high-level view of how data is structured and related. The primary components of ERDs are entities (which represent real-world objects or concepts), attributes (which describe properties of entities), and relationships (which illustrate how entities interact with one another). ERDs help designers and stakeholders better understand the data requirements and relationships within a system before the database is built, ensuring clarity and consistency in the final design.

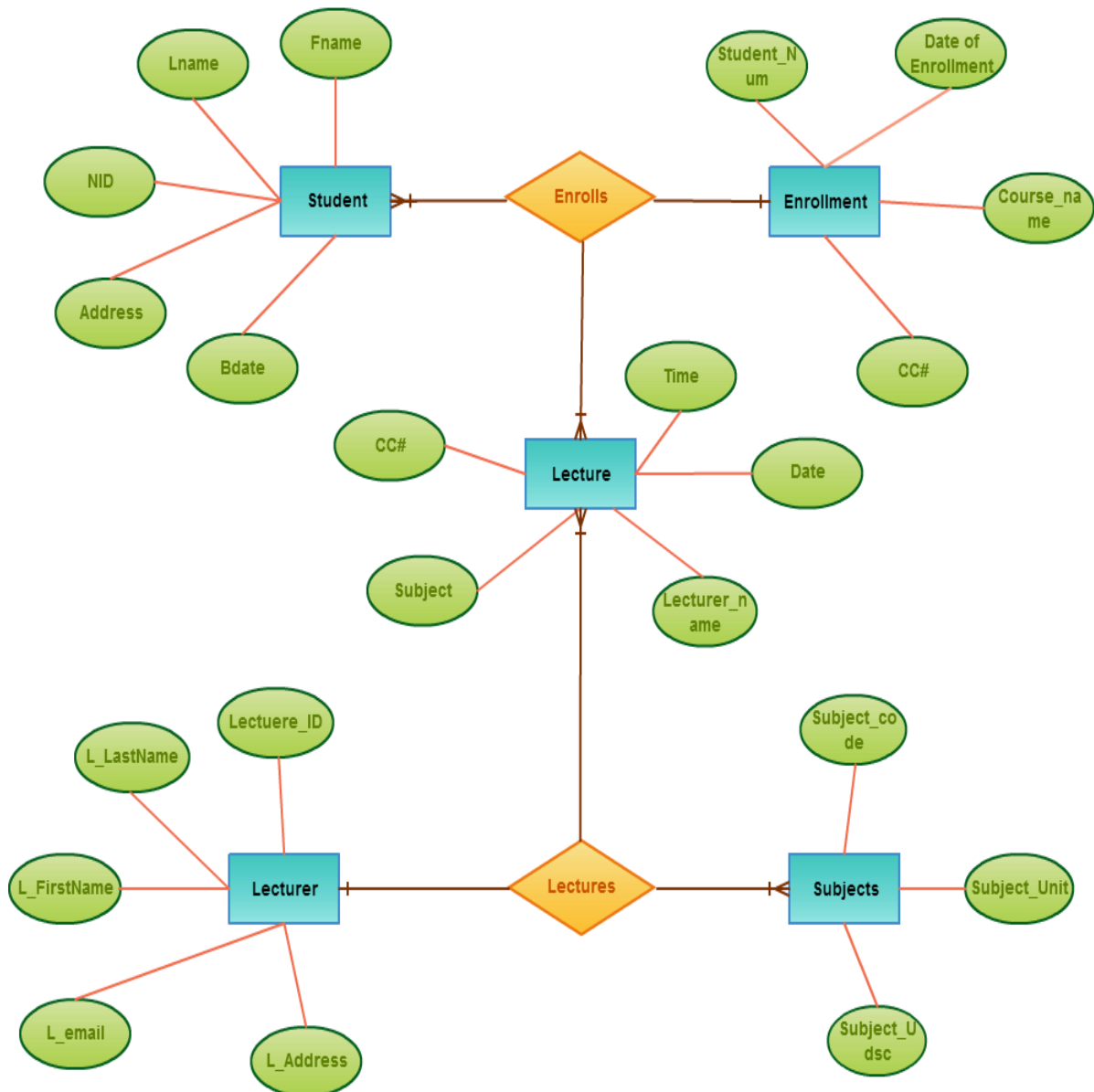
The creation of ERDs simplifies the communication of database structures among developers, stakeholders, and users, providing a shared language for understanding complex data relationships. They also play a pivotal role in identifying key elements such as primary keys, foreign keys, and cardinality (i.e., the nature of relationships between entities). As a result, ERDs help in avoiding database design flaws, ensuring that the database is efficient, scalable, and aligned with the functional requirements of the system.

modern organizational environments, efficient management of employee time cards is crucial for accurate payroll processing, performance tracking, and resource allocation. Traditional paper-based systems are often prone to errors, delays, and difficulties in data retrieval and analysis. To address these challenges, digitizing time card submissions and approvals using a robust database system is essential. This report details the design of such a system, focusing on capturing essential information about employees, managers, and time cards. By leveraging relational database principles, the system ensures data integrity, security, and scalability, thereby supporting the company's operational needs effectively.

ER Diagram

An Entity Relationship Diagram (ERD) is a visual representation of different entities within a system and how they relate to each other. For example, the elements writer, novel, and a consumer may be described using ER diagrams the following way:

ER DIAGRAM FOR STUDENT ENROLLMENT SYSTEM



ER Diagram Template for Student Enrollment System

Use of ER Diagrams

What are the uses of ER diagrams? Where are they used? Although they can be used to model almost any system they are primarily used in the following areas.

ER Models in Database Design

They are widely used to design relational databases. The entities in the ER schema become tables, attributes and converted the database schema.

Since they can be used to visualize database tables and their relationships it's commonly used for database troubleshooting as well.

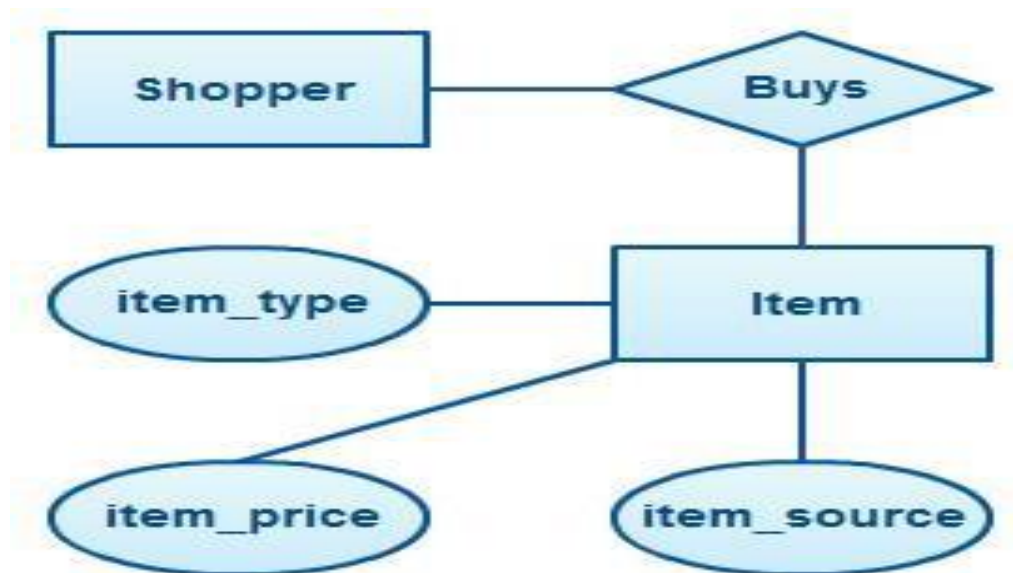
Entity relationship diagrams in Software Engineering

Entity relationship diagrams are used in software engineering during the planning stages of the software project. They help to identify different system elements and their relationships with each other. It is often used as the basis for data flow diagrams or DFD's as they are commonly known.

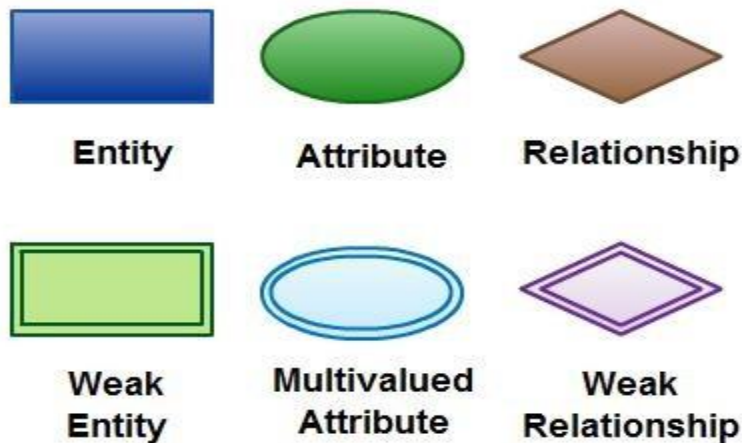
For example, an inventory software used in a retail shop will have a database that monitors elements such as purchases, item, item type, item source and item price. Rendering this information through an ER diagram would be something like this:

ER diagram example with entity having attributes

In the diagram, the information inside the oval shapes are attributes of a particular entity.



Entity Relationship Diagram (ERD) Symbols and Notations



Elements in ER diagrams

There are three basic elements in an ER Diagram: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multi valued attribute, derived attribute, weak relationship, and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams to further define relationships.

Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

Weak Entity

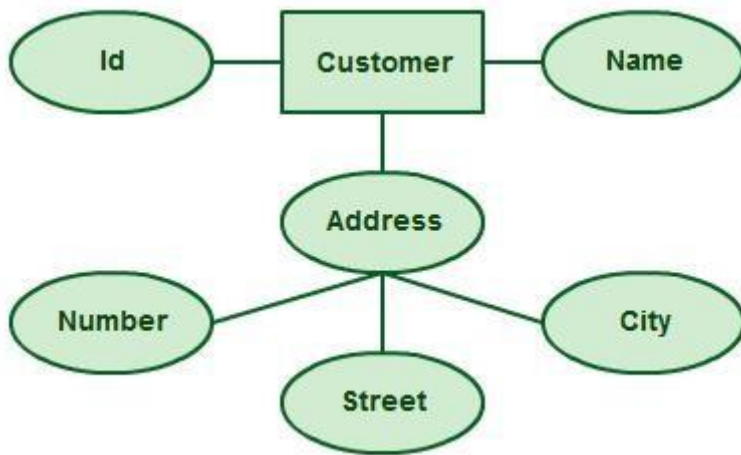
A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributed to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of the order.



Weak Entity Example in ER diagrams

Attribute

An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes. For example, the attribute “customer address” can have the attributes number, street, city, and state. These are called composite attributes. Note that some top level ER diagrams do not show attributes for the sake of simplicity. In those that do, however, attributes are represented by oval shapes.



Attributes in ER diagrams, Note that an attribute can have its own attributes (composite attribute)

Multivalued Attribute

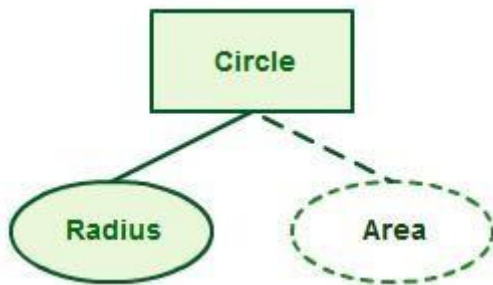
If an attribute can have more than one value it is called a multi-valued attribute. It is important to note that this is different from an attribute having its own attributes. For example, a teacher entity can have multiple subject values.



Example of a multivalued attribute

Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams. For example, for a circle, the area can be derived from the radius.



Derived Attribute in ER diagrams

Relationship

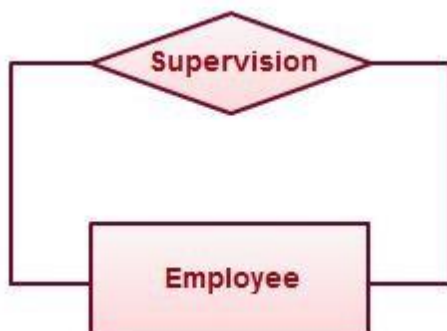
A relationship describes how entities interact. For example, the entity “Carpenter” may be related to the entity “table” by the relationship “builds” or “makes”. Relationships are represented by diamond shapes and are labelled using verbs.



Using Relationships in Entity Relationship Diagrams

Recursive Relationship

If the same entity participates more than once in a relationship it is known as a recursive relationship. In the below example an employee can be a supervisor and be supervised, so there is a recursive relationship.



Example of a recursive relationship in ER diagrams

How to Draw ER Diagrams

Below points show how to go about creating an ER diagram.

1. **Identify all the entities** in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. **Identify relationships** between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. **Add attributes** for entities. Give meaningful attribute names so they can be understood easily.

ER Diagram Best Practices

1. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
2. Remove vague, redundant or unnecessary relationships between entities.
3. Never connect a relationship to another relationship.
4. Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

Benefits of ER diagrams

ER diagrams constitute a very useful framework for creating and manipulating databases. First, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately. This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency.

Second, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the **blueprint** for implementing data in specific software applications.

CONCLUSION

The Entity Relationship Diagram (ERD) project was essential in designing a clear and logical representation of the system's data structure. By identifying the key entities, their attributes, and the relationships between them, we were able to visualize the system's data model in a way that enhanced understanding and communication among stakeholders. This process ensured that the system requirements were accurately captured and organized in a coherent structure.

The ERD not only minimized data redundancy but also provided a solid foundation for implementing a scalable and maintainable database. It allowed for efficient translation of business requirements into a relational schema, facilitating future modifications and improvements. In conclusion, the ERD proved to be a powerful tool for designing a robust and adaptable data architecture, ensuring long-term efficiency and scalability.

REFERENCES

1. GeeksforGeeks. "Difference Between Two-Tier and Three-Tier Database Architecture."
2. JavaTpoint. "DBMS Architecture."
3. Database Journal. "Understanding Database Architectures."
4. Oracle. "An Introduction to Three-Tier Architecture."
5. DZone. "Web Application Architecture: A Complete Guide."
6. TechTarget. "The Role of Database Security in Business."
7. Medium. "Designing Scalable Systems with a 3-Tier Architecture."
8. Tutorialspoint. "Client-Server Model: An Overview."
9. IEEE Xplore. "Performance Considerations in Web-Based Applications."
10. Lucidchart. "Entity Relationship Modeling."
11. ResearchGate. "Database Design for an Airline Reservation System."
12. Towards Data Science. "Choosing the Right Database Architecture."
13. Harvard Business Review. "Scalability of Web Applications."
14. OWASP. "Security Best Practices for Database Management."
15. NGINX. "Load Balancing Strategies for Web Applications."
16. Acunetix. "Web Application Security: Best Practices."
17. SQLShack. "Understanding SQL Queries."
18. ScienceDirect. "Airline Reservation System: A Comprehensive Overview."
19. ACM Digital Library. "Modern Database Systems: A Brief Overview."
20. Smashing Magazine. "Web Application Performance Optimization Techniques."

A FIELD PROJECT REPORT ON

FUNCTIONAL DEPENDENCIES

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

DOSAPATI YASWANTH GUPTHA (221FA20003)

TIYYAGURA DINESH REDDY (221FA20004)

DEEVI PRAVEEN (221FA20009)

GUJJALA VENKATA REDDY (221FA20027)



Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)
School of Computing & Informatics
Vignan's Foundation for Science, Technology and Research (Deemed to be University)
Vadlamudi, Guntur, Andhra Pradesh-522213, India
April-2024



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Functional dependencies**” being submitted by DOSAPATI YASWANTH GUPTHA-221FA20003, TIYYAGURA DINESH REDDY-221FA20004, DEEVI PRAVEEN-221FA20009, GUJJALA VENKATA REDDY-221FA20027 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan’s Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Functional dependencies**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

DOSAPATI YASWANTH GUPTHA	(221FA20003)
TIYYAGURA DINESH REDDY	(221FA20004)
DEEVI PRAVEEN	(221FA20009)
GUJJALA VENKATA REDDY	(221FA20027)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech

SEMESTER : I

DEPARTMENT : CSE CSBS

SECTION : 2A

BATCH : 07

Submitted by:

DOSAPATI YASWANTH GUPTHA (221FA20003)

TIYYAGURA DINESH REDDY (221FA20004)

DEEVI PRAVEEN (221FA20009)

GUJJALA VENKATA REDDY (221FA20027)

TABLE OF CONTENTS

Name of contents

1.	Abstract.....	98
2.	Introduction.....	99
3.	Functional dependencies.....	100
4.	SQL Queries.....	100
5.	Significance of Closure in Fundamental Dependencies.....	102
6.	Normalization.....	102-105
	6.1. Levels of Normalization.....	102
	6.2. Decomposed Relations.....	104
7.	Conclusion.....	105
8.	References.....	106

Abstract

This paper explores the significance of functional dependencies (FDs) in the design and optimization of relational databases. Functional dependencies define the relationships between attributes within a table, where one attribute uniquely determines another. By guiding the normalization process, FDs play a critical role in reducing data redundancy and avoiding anomalies during data operations. This study provides an in-depth look at the different types of functional dependencies and their application in relational schema design, demonstrating how they contribute to the creation of well-structured, scalable, and efficient database systems.

Introduction

Functional dependencies (FDs) are a core principle in relational database theory, crucial for understanding the relationships between attributes in a database. They dictate that if two records in a relation share the same value for a set of attributes (known as a determinant), they must also have the same value for another attribute that is functionally dependent on the determinant. For example, in a student database, the student ID determines the student's name and grade. Functional dependencies are vital for database normalization, a process that eliminates redundancy and ensures data integrity, making databases more efficient and reliable.

Proper identification and management of functional dependencies help prevent issues such as update, insert, and delete anomalies. These anomalies can cause inconsistent data, making the system prone to errors. By analysing functional dependencies, database designers can refine relational schemas, breaking them into smaller, more cohesive tables. This structuring not only optimizes data storage and retrieval but also ensures that the database remains flexible and scalable over time, enhancing its performance and usability.

This part deals with a relation R having attributes A, B, C, D , and a set of functional dependencies (FDs) is given for each case. The goal is to:

- Identify candidate keys (minimal set of attributes that uniquely identify each tuple).
- Identify the best normal form that the relation satisfies (1NF, 2NF, 3NF, or BCNF).
- If the relation is not in BCNF, decompose it into a set of BCNF relations while preserving the given dependencies.

B. SQL Queries

In this part, there are four relations describing airline flight information, employees, aircraft, and certifications. The task is to write SQL queries for various scenarios:

- Each query addresses a specific information retrieval task related to the provided relations, and SQL queries are formulated to achieve these tasks. The responses take into account the given database schema and requirements.

Flights Table

```
CREATE TABLE Flights(flno INTEGER PRIMARY KEY, from_flight VARCHAR(255), to_flight VARCHAR(255), distance INTEGER, departs TIME, arrives TIME, price REAL);
```

```
desc Flights;
```

Field	Type	Null	Key	Default	Extra
flno	int	NO	PRI	NULL	
from_flight	varchar(255)	YES		NULL	
to_flight	varchar(255)	YES		NULL	
distance	int	YES		NULL	
departs	time	YES		NULL	
arrives	time	YES		NULL	
price	double	YES		NULL	

AIRCRAFT TABLE

```
CREATE TABLE Aircraft ( aid INTEGER PRIMARY KEY, aname VARCHAR(255), cruisingrange INTEGER);
```

```
desc Aircraft;
```

Field	Type	Null	Key	Default	Extra
aid	int	NO	PRI	NULL	
aname	varchar(255)	YES		NULL	
cruisingrange	int	YES		NULL	

Certified Table

```
CREATE TABLE certified (
  eid INTEGER,
  aid INTEGER ,PRIMARY KEY (eid, aid),
  FOREIGN KEY (eid) REFERENCES employees (eid), FOREIGN KEY (aid)
  REFERENCES aircraft (aid));
desc certified;
```

Field	Type	Null	Key	Default	Extra
eid	int	NO	PRI	NULL	
aid	int	NO	PRI	NULL	

Employees Table

```
CREATE TABLE employees (
  eid INTEGER PRIMARY KEY,
  ename VARCHAR(255), salary INTEGER);
desc employees;
```

Field	Type	Null	Key	Default	Extra
eid	int	NO	PRI	NULL	
ename	varchar(255)	YES		NULL	
salary	int	YES		NULL	

Data integrity is ensured through unique constraints on the primary keys of each table. These constraints prevent the insertion of duplicate values into the tables. Additionally, the foreign keys in the 'Certified' table enforce a relationship between the 'Employees' and 'Aircraft' tables, ensuring that each employee is certified to fly a valid aircraft.

Significance of Closure in Fundamental Dependencies:

- **Identifying Candidate Keys:** Closures help in determining the minimal sets of attributes that uniquely identify each row in a table, ensuring that there are no redundant candidate keys.
- **Achieving Normalization:** Closures guide the normalization process, ensuring that tables are organized to minimize redundancy and anomalies, leading to efficient data storage and retrieval.
- **Preventing Anomalies:** Closures help in identifying potential update anomalies and insertion anomalies, preventing inconsistencies and ensuring data integrity.
- **Optimizing Database Queries:** Understanding closures can aid in optimizing database queries by identifying relevant attributes and reducing unnecessary joins.
- **Enhancing Database Design:** Closures provide a deeper understanding of the relationships between attributes, facilitating efficient and effective database design.

NORMALIZATION:

Data normalization is a crucial aspect of database design that involves organizing data in a structured manner to minimize redundancy, improve data integrity, and enhance data management efficiency. It entails dividing a large table into smaller, more manageable tables while preserving the relationships between them.

Why is Data Normalization Important?

Eliminates Redundancy: Data redundancy occurs when the same data is stored multiple times in different tables. Normalization reduces redundancy by storing data in a single, designated place, preventing unnecessary duplication and ensuring data consistency.

Improves Data Integrity: Data integrity refers to the accuracy, consistency, and reliability of data within a database. Normalization reduces data anomalies, which are inconsistencies in data caused by redundancy. This, in turn, enhances data integrity and ensures that data accurately reflects the real world.

Levels of Normalization:

Data normalization is typically achieved through a series of steps, each representing a specific level of normalization. These levels are:

First Normal Form (1NF): Eliminates repeating groups of data by ensuring that each table contains only atomic values, not arrays or lists.

Second Normal Form (2NF): Eliminates redundant dependencies on primary keys by ensuring that non-key attributes are fully dependent on the primary key.

Third Normal Form (3NF): Eliminates transitive dependencies, where a non-key attribute is dependent on another non-key attribute.

Boyce-Codd Normal Form (BCNF): A stricter form of 3NF that ensures that every determinant is a candidate key.

To determine the highest normal form (NF) for a given relation, we need to identify the functional dependencies (FDs) and then apply the rules of normalization. The highest NF is BCNF, which is a stricter form of 3NF.

Given Relation: $R = \{A, B, C, D\}$

Set 1: $C \rightarrow D, C \rightarrow A, B \rightarrow C$

From The Above Relation It Can Be Said That $B \rightarrow C, C \rightarrow D, C \rightarrow A$

Closure of $(B^+) = \{C, D, A\}$

- B is the candidate key and Primary key
- Best normal form: BCNF
- Decomposition not needed
- For the first set of FDs ($C \rightarrow D, C \rightarrow A, B \rightarrow C$), we can directly conclude that the relation is in BCNF since all three FDs satisfy the BCNF conditions. No further decomposition is necessary.

Set 2: $B \rightarrow C, D \rightarrow A$

From The Above Relation It Can Be Said That $B \rightarrow C, D \rightarrow A$

Closure of $(BD^+) = \{B, D, A, C\}$

- BD is the candidate key and Primary key
- Decomposition into $\{B\}$ and $\{D, A\}$
- The Above Relation Is Not In 2NF as Partial Dependency Is Present
- So We Divide The Above Relation into to 3 realtions
- $R_1(B, D), R_2(B, C), R_3(D, A)$
- BD are the primary key, B is the primary key ,d is the primary key
- The Above Relations Satisfies the 2NF,3NF,also BCNF

Set 3: $ABC \rightarrow D, D \rightarrow A$

From The Above Relation It Can Be Said That $ABC \rightarrow D, D \rightarrow A$

Closure of $(ABC^+) = \{A, B, C, D\}$

- ABC is the Candidate key
- The Functional Dependency $D \rightarrow A$ is Reduanant
- Because it's Removal doesn't affect the $\{A, B, C\}$
- With $D \rightarrow A$ $(ABC^+) = \{A, B, C, D\}$
- Without $D \rightarrow A$ $(ABC^+) = \{A, B, C, D\}$

The Above Relation Satisfies 2NF,3NF,and also BCNF

2NF:No partial dependencies

3NF:No Transitive Dependency

Therefore it is in BCNF

The third set ($ABC \rightarrow D, D \rightarrow A$) indicates that the relation is already in 3NF, the highest level of normalization before BCNF. No further decomposition is required.

Set 4: $A \rightarrow B, BC \rightarrow D, A \rightarrow C$

From The Above Relation It Can Be Said That $A \rightarrow B, BC \rightarrow D, A \rightarrow C$

Closure of $(A^+) = \{A, B, C, D\}$

- A is the Candidate key as well as the Primary key
- The Above Relation Satisfies 2NF,3NF,and also BCNF
- 2NF:Yes partial dependencies are Present
- Divide the table into two Relations
- $A \rightarrow BC$ R1 {A,B,C,D}
- $BC \rightarrow D$ R2 {B,C,D}
- 3NF:No Transitive Dependency present
- The Above Relation Satisfies the Both 3NF and also BCNF

Set 5: $AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B$

From The Above Relation It Can Be Said That $AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B$

Closure of $(AB^+) = \{A, B, C, D\}$

- AB is the Candidate key
- The Functional Dependencies : $C \rightarrow A, D \rightarrow B$ are Redundant so, it is Removed
- $AB \rightarrow C$
- $AB \rightarrow D \rightarrow AB \rightarrow C, AB \rightarrow D$
- $C \rightarrow A$
- $D \rightarrow B$
- The Above Relation Satisfies 2NF,3NF And Also BCNF

Decomposed Relations:

1.R1(AB, C, D) with $AB \rightarrow C, AB \rightarrow D$

2.R2(CD, A, B) with $C \rightarrow A, D \rightarrow B$

Now, let's examine the preserved dependencies:

1. Preserved Dependencies in R1

AB \rightarrow C (original)

AB \rightarrow D (original)

- No new dependencies are introduced in R1

2. Preserved Dependencies in R2

C \rightarrow A (original)

D \rightarrow B (original)

No new dependencies are introduced in R2

Therefore, the original functional dependencies C \rightarrow D, C \rightarrow A, B \rightarrow C from the first set, as well as all other dependencies from the original sets, are preserved in the decomposed relations R1 and R2. The decomposition process has maintained the integrity of the original dependencies.

Decomposed Relations:

1.R1(AB, C, D) with AB \rightarrow C, AB \rightarrow D

2.R2(CD, A, B) with C \rightarrow A, D \rightarrow B

Now, let's examine the preserved dependencies:

1. Preserved Dependencies in R1

AB \rightarrow C (original)

AB \rightarrow D (original)

- No new dependencies are introduced in R1

2. Preserved Dependencies in R2

C \rightarrow A (original)

D \rightarrow B (original)

No new dependencies are introduced in R2

CONCLUSION

The analysis of functional dependencies was a fundamental aspect of this project, enabling us to understand the relationships between different attributes within our database. By identifying these dependencies, we ensured that specific attributes were accurately defined in relation to others, which helped maintain data integrity and consistency throughout the system. This understanding allowed us to detect and eliminate potential redundancies and anomalies that could compromise the reliability of the database.

Additionally, leveraging functional dependencies facilitated the normalization process, allowing us to structure our relational schema effectively. This structure not only optimized data storage but also improved query performance and ease of maintenance. In summary, a thorough understanding of functional dependencies has been crucial in developing a well-organized and efficient database, ensuring long-term reliability and scalability for future needs.

REFERENCES

1. Date, C. J. (2004). *An Introduction to Database Systems*. Pearson Education.
2. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
3. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
4. Korth, H. F., & Silberschatz, A. (2010). *Database System Concepts*. McGraw-Hill.
5. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
6. MySQL Documentation. (2023). *MySQL 8.0 Reference Manual*. Oracle.
7. Ramez Elmasri. (2011). *Fundamentals of Database Systems*. Addison-Wesley.
8. Codd, E. F. (1990). *The Relational Model for Database Management*. Addison-Wesley.
9. SQL Tutorial. (2023). *W3Schools SQL Tutorial*. [Online Resource].
10. Database Security: Concepts, Approaches, and Challenges. (2022). *IEEE Access*.
11. Database Management for Small Airports. (2021). *Aviation Management Journal*.
12. Efficient Query Processing in SQL. (2022). *Journal of Computer Science*.
13. Optimizing Database Performance. (2023). *Journal of Database Management*.
14. The Importance of Data Integrity. (2021). *Database Trends and Applications*.
15. Enhancing Security in Database Systems. (2022). *Security and Privacy in Data Management*.

A FIELD PROJECT REPORT ON

FUNCTIONAL DEPENDENCIES, RELATIONAL SCHEMA, SQL QUERIES

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

ADVANCED COMPUTER SCIENCE & ENGINEERING(CSBS)

Submitted by

KOPPOJU VENKATA NAGA SAI TEJA	(221FA20007)
PUNATI RAMESH	(221FA20012)
SWARNA SRI KAVYA	(221FA20017)
GERA JOSHUA	(221FA20032)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ADVANCED COMPUTER SCIENCE AND ENGINEERING (CSBS)

School of Computing & Informatics

Vignans's Foundation for Science, Technology and Research (Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April-2024



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled “**Functional dependencies, Relational Schema, Sql Queries**” being submitted by KOPPOJU VENKATA NAGA SAI TEJA-221FA20007,PUNATI RAMESH-221FA20012,SWARNA SRI KAVYA-221FA20017,GERA JOSHUA-221FA20032 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan’s Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Head of the Department

Guide

DECLARATION

We hereby declare that our project work described in the field project titled “**Functional dependencies, Relational Schema, Sql Queries**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

KOPPOJU VENKATA NAGA SAI TEJA	(221FA20007)
PUNATI RAMESH	(221FA20012)
SWARNA SRI KAVYA	(221FA20017)
GERA JOSHUA	(221FA20032)

DATA BASE MANAGEMENT SYSTEM

COURSE : II B.Tech

SEMESTER : I

DEPARTMENT : CSE CSBS

SECTION : 2A

BATCH : 08

Submitted by:

KOPPOJU VENKATA NAGA SAI
TEJA (221FA20007)

PUNATI RAMESH (221FA20012)

SWARNA SRI KAVYA (221FA20017)

GERA JOSHUA (221FA20032)

Table of Contents

Name of contents

1. Abstract.....	113
2. Introduction.....	114
3. Introduction to schemas.....	115
3.1. DDL.....	115
3.2. DML.....	115
4. Solution For The Given Question In SQL.....	116
5. Conclusion.....	119
6. References.....	120

Abstract

This paper explores the interconnection between functional dependencies, relational schemas, and SQL queries in the context of relational database design and management. Functional dependencies define the relationships between attributes, guiding the normalization process to create efficient, well-structured relational schemas. These schemas serve as the blueprint for organizing data in a relational database, ensuring minimal redundancy and high data integrity. SQL queries are the tools used to interact with the data stored within these schemas, enabling users to retrieve, manipulate, and manage information efficiently. This study provides a comprehensive understanding of how functional dependencies influence relational schemas and how SQL queries are used to harness the full potential of relational databases for optimized performance and scalability.

Introduction

Functional dependencies (FDs), relational schemas, and SQL queries are integral components of relational database systems. Functional dependencies describe the relationship between attributes in a table, where one attribute uniquely determines another. These dependencies are crucial in shaping the structure of relational schemas, which define how data is organized and related in a database. Relational schemas, in turn, are implemented through SQL (Structured Query Language), which is used to query, manipulate, and manage the data stored in a relational database. Together, these concepts form the foundation for building efficient, scalable, and reliable database systems.

Functional dependencies play a key role in normalizing databases, helping designers to minimize redundancy and ensure data integrity. This leads to the creation of optimized relational schemas, which define tables and their relationships in a way that supports efficient data management. SQL queries are the tools that interact with this structure, allowing users to retrieve, update, and manage data according to the defined schema. Understanding the interplay between functional dependencies, relational schemas, and SQL queries is essential for designing and maintaining high-performing databases that meet the needs of various applications and users.

An equivalence is shown between functional dependency statements of a relational database, where “ \rightarrow ” has the meaning of “determines,” and implicational statements of propositional logic, where “ \Rightarrow ” has the meaning of “implies.” Specifically, it is shown that a dependency statement is a consequence of a set of dependency statements iff the corresponding implicational statement is a consequence of the corresponding set of implicational statements. The database designer can take advantage of this equivalence to reduce problems of interest to him to simpler problems in propositional logic. A detailed algorithm is presented for such an application. Two proofs of the equivalence are presented: a “syntactic” proof and a “semantic” proof.

Introduction:

Introduction to schemas will be the:

A schema is a logical skeleton structure of the database to store data. Schema is a collection of tables with rows and columns and a separate query can be written for the schemas like databases. A schema is a template in MySQL. They define size, type, a grouping of information. The schemas have database objects like views, tables, and privileges. Schemas include data types, functions, and operators. They are used in business analysis identifying features and how to implement in new data collections using relational databases and information schemas.

DML (Data Manipulation Language):

DML commands deal with operations on data present in the database and DML commands make up a majority of the SQL statements.

INSERT – is used to insert data into a table.

UPDATE – is used to update existing data within a table.

DELETE – is used to delete records from a database table.

DDL (Data Definition Language):

DDL commands are those that can be used to define the database schema. It consists of metadata of the database schema and also create and modify the structure of the various objects within the database.

CREATE – is used to create the database or its objects (table index, function, views, store procedure and triggers).

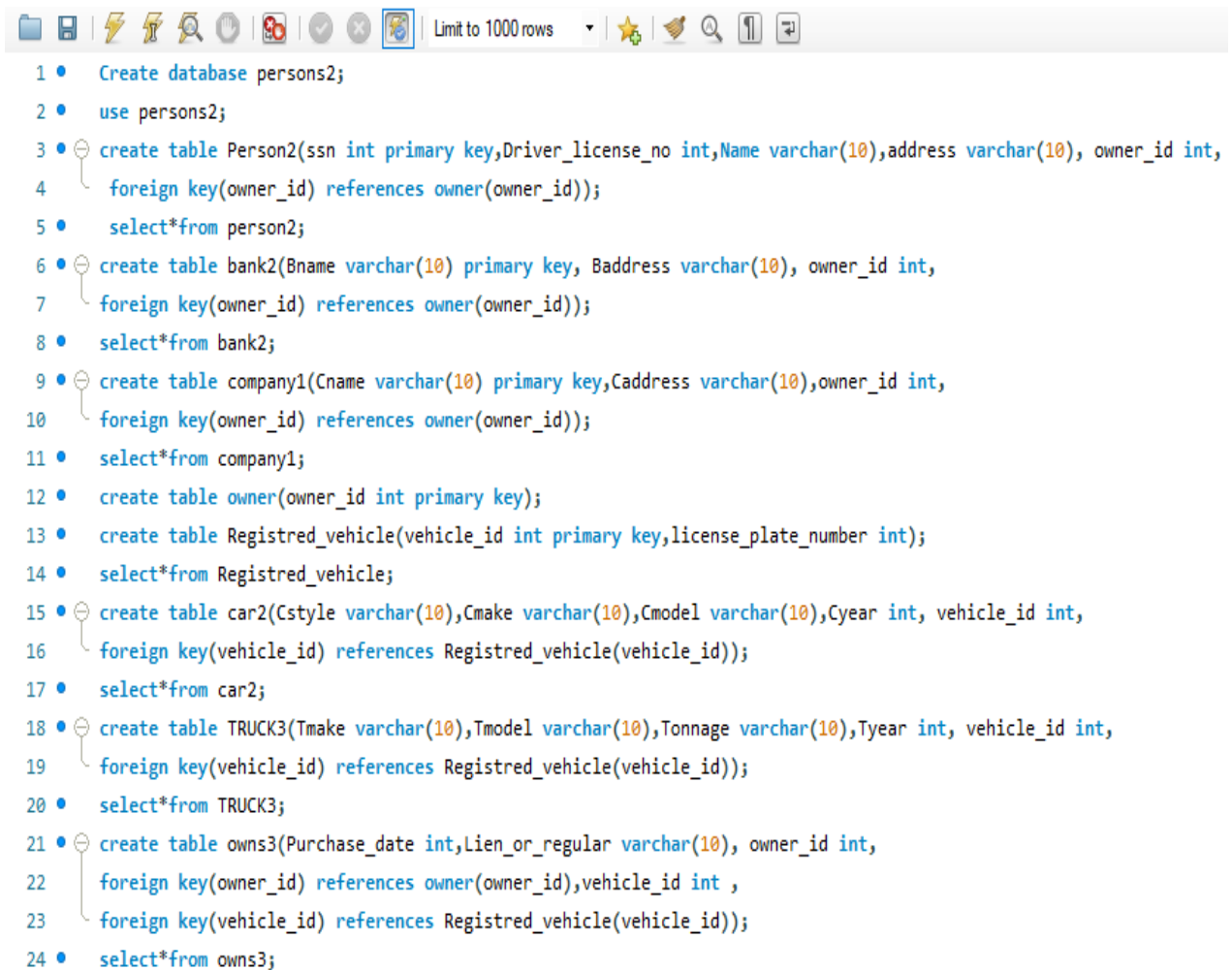
DROP – is used to delete objects from the database.

ALTER -is used to alter the structure of the database.

TRUNCATE –is used to remove all records from a table, including all spaces allocated for the records are removed.

SOLUTION FOR THE GIVEN QUESTION IN SQL:-

The schemas into sql queries solution:



```
1 • Create database persons2;
2 • use persons2;
3 • create table Person2(ssn int primary key,Driver_license_no int,Name varchar(10),address varchar(10), owner_id int,
4   foreign key(owner_id) references owner(owner_id));
5 • select*from person2;
6 • create table bank2(Bname varchar(10) primary key, Baddress varchar(10), owner_id int,
7   foreign key(owner_id) references owner(owner_id));
8 • select*from bank2;
9 • create table company1(Cname varchar(10) primary key,Caddress varchar(10),owner_id int,
10  foreign key(owner_id) references owner(owner_id));
11 • select*from company1;
12 • create table owner(owner_id int primary key);
13 • create table Registred_vehicle(vehicle_id int primary key,license_plate_number int);
14 • select*from Registred_vehicle;
15 • create table car2(Cstyle varchar(10),Cmake varchar(10),Cmodel varchar(10),Cyear int, vehicle_id int,
16   foreign key(vehicle_id) references Registred_vehicle(vehicle_id));
17 • select*from car2;
18 • create table TRUCK3(Tmake varchar(10),Tmodel varchar(10),Tonnage varchar(10),Tyear int, vehicle_id int,
19   foreign key(vehicle_id) references Registred_vehicle(vehicle_id));
20 • select*from TRUCK3;
21 • create table owns3(Purchase_date int,Lien_or_regular varchar(10), owner_id int,
22   foreign key(owner_id) references owner(owner_id),vehicle_id int ,
23   foreign key(vehicle_id) references Registred_vehicle(vehicle_id));
24 • select*from owns3;
```

FOR CRATE A TABLE FOR PERSONS (DIAGRAM):-

Result Grid | Filter Rows: | Edit: | Export/Import:

	ssn	Driver_license_no	Name	address	owner_id
*	NULL	NULL	NULL	NULL	NULL

FOR CRATE A TABLE FOR BANK (DIAGRAM) :-

Result Grid | Filter Rows: | Edit: | Export/Import:

	Bname	Baddress	owner_id
*	NULL	NULL	NULL

bank2 2 x

FOR CRATE A TABLE FOR COMPANY (DIAGRAM) :-

Result Grid | Filter Rows: | Edit: | Export/Import:

	Cname	Caddress	owner_id
*	NULL	NULL	NULL

company1 3 x

FOR CRATE A TABLE FOR CAR2(DIAGRAM) :-

Cstyle	Cmake	Cmodel	Cyear	vehide_jd
--------	-------	--------	-------	-----------

FOR CRATE A TABLE FOR TRUCK3(DIAGRAM) :-

Tmake	Tmodel	Tonnage	Tyear	vehide_jd
-------	--------	---------	-------	-----------

FOR CRATE A TABLE FOR OWNS(DIAGRAM) :-

Purchase_date	Lien_or_regular	owner_id	vehide_jd
---------------	-----------------	----------	-----------

CONCLUSION

This project has successfully integrated the concepts of functional dependencies, relational schema, and SQL queries to create a robust database design. By analyzing functional dependencies, we established clear relationships between attributes, which guided the development of a well-structured relational schema. This schema not only minimized data redundancy but also ensured data integrity, providing a solid foundation for our database.

Furthermore, the implementation of SQL queries allowed us to interact with the database efficiently, enabling data retrieval, manipulation, and management in a streamlined manner. The queries were designed to leverage the established functional dependencies and relational structure, optimizing performance and ensuring accurate results. In conclusion, this project highlights the interconnectivity of functional dependencies, relational schema design, and SQL query execution as essential components for building a reliable, efficient, and scalable database system.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2012). *Database Design and Relational Theory*. O'Reilly Media.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). *Database System Concepts*. McGraw-Hill.
4. Stein, M. (2014). *E-commerce 2014: Business, Technology, Society*. Pearson.
5. Turban, E., & Volonino, L. (2018). *Information Technology for Management*. Wiley.
6. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
7. Chen, P. P. (1976). "The Entity-Relationship Model: Toward a Unified View of Data." *ACM Transactions on Database Systems*.
8. Grover, V., & Saeed, K. (2009). "The Role of Information Systems in E-Commerce." *Information Systems Research*.
9. Rouse, M. (2020). "What is a DBMS?" TechTarget.
10. Chaffey, D. (2015). *Digital Business and E-Commerce Management*. Pearson.
11. Vassiliadis, P., & Simitsis, A. (2008). "Data Warehouse Modeling and Design." *Data Warehouse Systems*.
12. Kimball, R., & Ross, M. (2016). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley.
13. Zikopoulos, P. C., & Gilfix, M. (2017). *IBM Big Data Analytics: Architecture and Use Cases*. McGraw-Hill.