Learning Objectives

After undergoing the topics, the students will be able to:

- Know about the DBMS system and classify the DBMS user
- Describe data models, DBMS architecture
- Describe database languages and interfaces
- Functional dependency and Normalization
- Different types of keys

1. INTRODUCTION

1. **What Is Data?** :

A sequence of characters stored in computer memory or storage. It is nothing, it is the convey of meaning.

1.2 **Information**-organized data **Knowledge**-
information imbued with intelligence.

1.3 **Database:**

A database is a software system which consists relational data used by various applications.

1.4 **Database Management System**:

A Database Management System (DBMS) is a software system that facilitates the process defining, constructing, manages, control and modify database.

1.5 **File Systems:**

• File is uninterrupted, unstructured collection of information
• File operations: delete, catalog, create, rename, open, close, read, write.

1.6 **File Management System Problems:**

• Data redundancy
• Data Access: New request-new program
• Data is not isolated from the access

1.7 **Use of DBMS**

Data independence and efficient access. Reduced application development time. Data integrity and security.
Uniform data administration.

1.8 **ADVANTAGES OF A DBMS:**

1.8.1 **Data independence**: Application programs should be as independent as possible from details of data representation and storage.

1.8.2 **Data integrity and security**: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data.

1.8.3.Data administration: When several users share the data, centralizing the administration of data can or significant improvements.

Data Independence

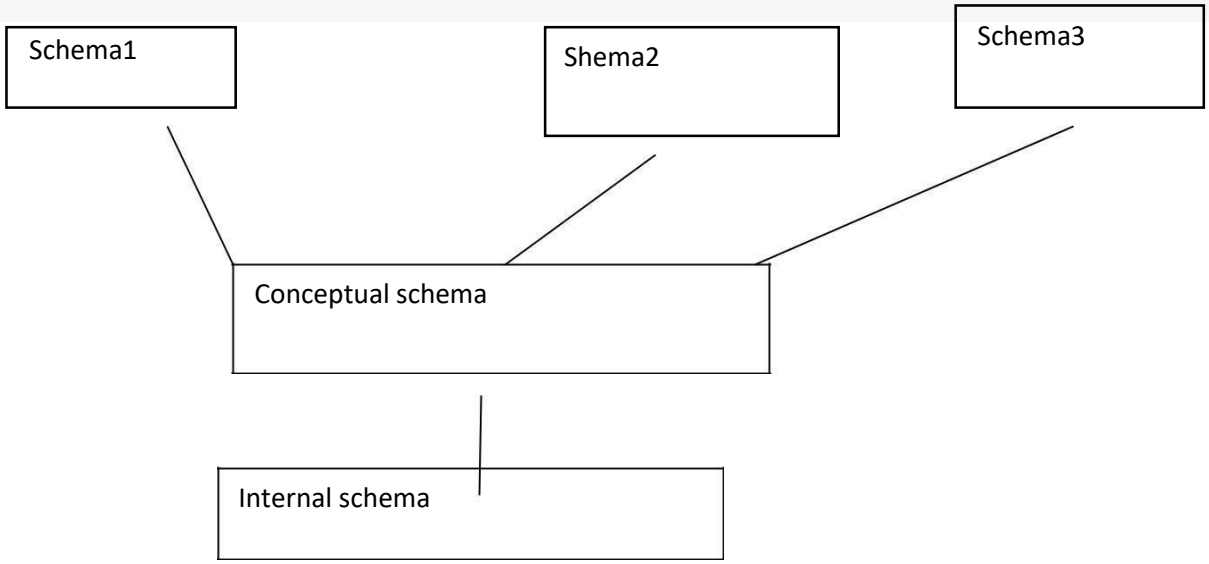
Applications insulated from how data is structured and stored.

Logical data independence: Protection from changes in logical structure of data.

Physical data independence: Protection from changes in physical structure of data if the data as being accessed by only one user at a time.

A schema is a description of a particular collection of data, using a given data model

2.Design of schema



2.1 Physical – view: *How data is stored, how it is accessed, how data is modified, is data ordered, how data is allocated to computer memory*

2.2 Conceptual – view: *The conceptual schema (sometimes called the logical schema) describes the stored data in terms of the data model of the DBMS. In a relational DBMS, the conceptual schema describes all relations that are stored in the database*

2.3 Internal – view: *External schemas, which usually are also in terms of the data model of the DBMS, allow data access to be customized (and authorized) at the level of individual users or groups of users.*

3. Data abstraction

A**bstraction-** is the process by which data and programs are defined with a representation similar in form to its meaning, while hiding away the implementation details.

4.Data Models

a data model is a collection of concepts for describing

- *Data and Data relationships.*
- *Data semantics.*
- *Data constraints.*

***5.Database Administrator**:*

- *A person who has such central control over the system is called a database administrator (DBA)*
- *The DBA creates the original database schema by executing a set of data definition statements in the DDL.*
- *Define Storage structure and access-method.*
- *The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.*
- *Granting of authorization for data access.*
- *Integrity*

DDL-It defines the conceptual schema and also gives some details about how to implement this schema.

DML-It used to manipulate data just like retrival, insert, deletion and modification.

diagrams

Chapter 2 Entity Relationships

2.1 **Entity:** An entity is a “thing” or “object” in the real world that is distinguishable from all other objects. For example, each person in an enterprise is an entity.

2.2 **Entity set:** An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all persons who are customers at a given bank, for example, can be defined as the entity set customer.

2.3 **Attributes:** are descriptive properties possessed by each member of an entity set.

2.3.1 Simple and composite attributes: the attributes are simple if they are not divided into subparts. On the other hand, if it divided into subparts is called as "composite attributes". For example, an attribute name could be structured as a composite attribute consisting of first-name, middle-initial, and last-name.

2.3.2 Single-valued and multivalve attributes: For instance, the loan-number attribute for a specific loan entity refers to only one loan number. Such attributes are said to be single valued. There may be instances where an attribute has a set of values for a specific entity. Consider an employee entity set with the attribute phone-number. An employee may have zero, one, or several phone numbers, and different employees may have different numbers of phones. This type of attribute is said to be multivalve.

2.3.3 Derived attribute: The value for this type of attribute can be derived from the values of other related attributes or entities.

***3.*Relationship Sets:** A relationship is an association among several entities. A relationship set is a set of relationships of the same type.

3.1Mapping Cardinalities: Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

- **One to one**. An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
- **One to many**. An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.
- **Many to one**. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.
- **Many to many**. An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

***4.*Keys:** A key allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

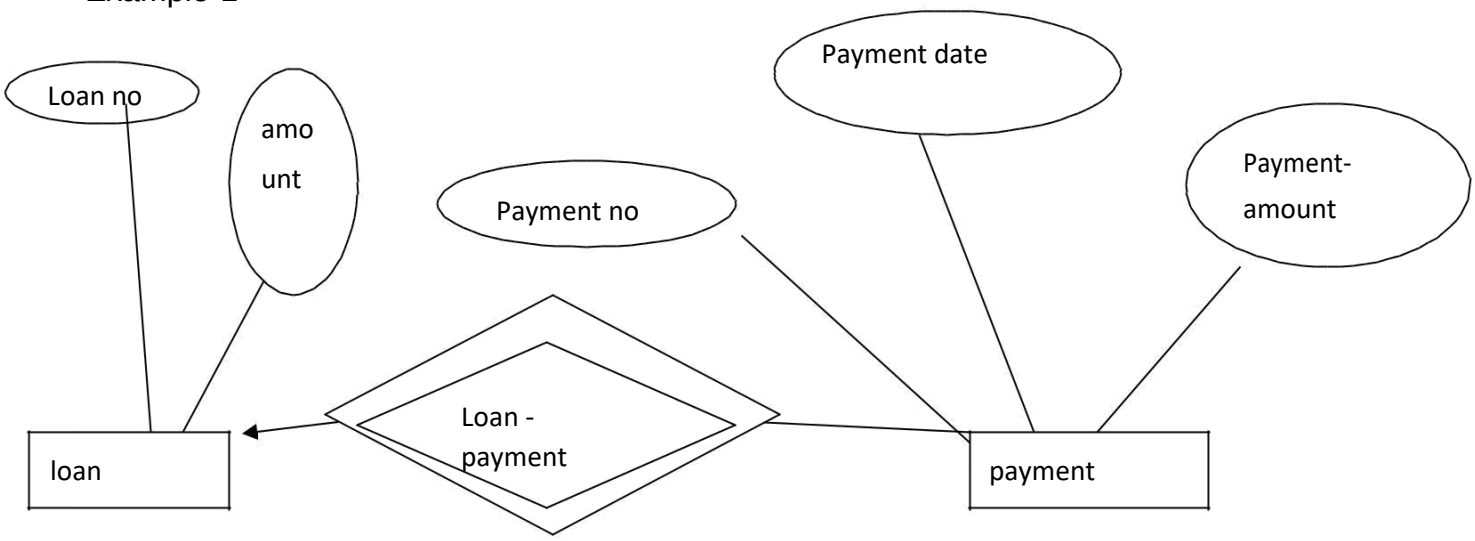
***4.1*Super key:** A super key is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set. For example, the customer-id attribute of the entity set customer is sufficient to distinguish one customer entity from another. Thus, customer-id is a super key. Similarly, the combination of customer-name and customer-id is a super key for the entity set customer. The customer-name attribute of customer is not a super key, because several people might have the same name.

***4.2*Candidate key:** minimal super keys are called candidate keys.

***4.3*Primary key:** which denotes the unique identity is called as primary key. Primary key to denote a candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set. A key (primary, candidate, and super) is a property of the entity set, rather than of the individual entities.

***4.4*Weak Entity Sets:** An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed a weak entity set. An entity set that has a primary key is termed a strong entity set

Example-1



For a weak entity set to be meaningful, it must be associated with another entity set, called the identifying or owner entity set. The relationship associating the weak entity set with the identifying entity set is called the identifying relationship. The participation of the weak entity set in the relationship is called total.

In our example, the identifying entity set for payment is loan, and a relationship loan-payment that associates payment entities with their corresponding loan entities is the identifying relationship. Although a weak entity set does not have a primary key, we nevertheless need a means of distinguishing among all those entities in the weak entity set that depend on one particular strong entity. The discriminator of a weak entity set is a set of attributes that allows this distinction to be made.

Here are the geometric shapes and their meaning in an E-R Diagram

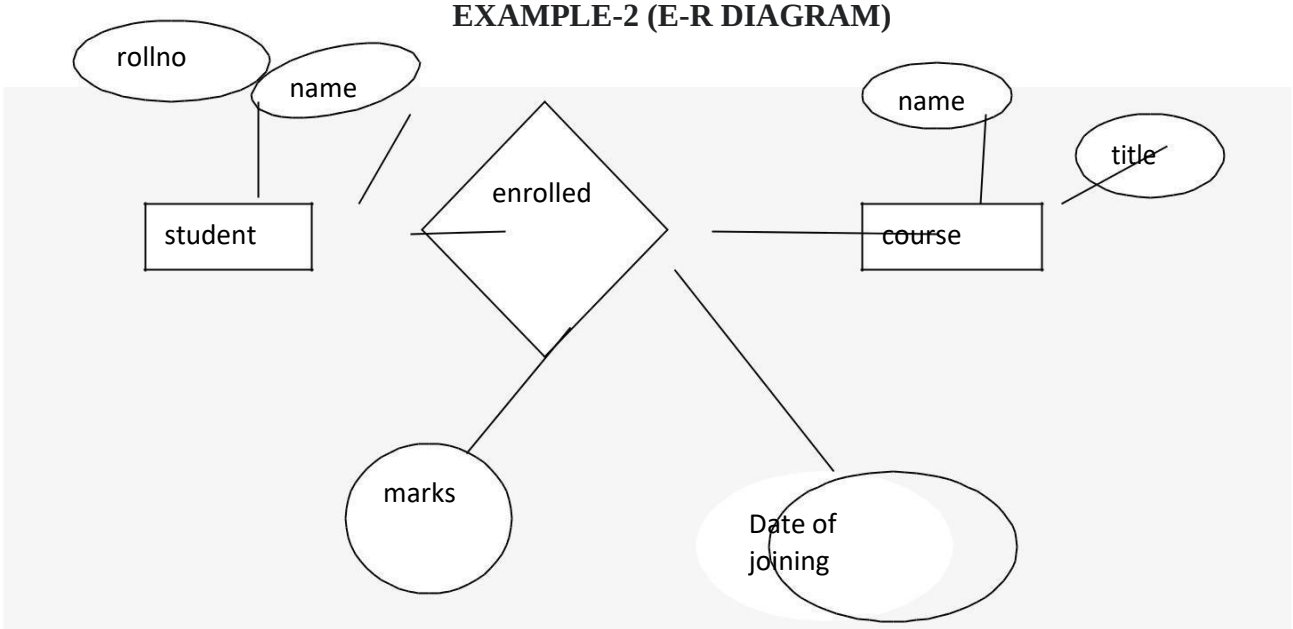
1. In E-R diagrams, a doubly outlined box indicates a weak entity set, and a doubly outlined diamond indicates the corresponding identifying relationship
2. Rectangle: Represents Entity sets.
3. Ellipses: Attributes
4. Diamonds: Relationship Set
5. Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
6. Double Ellipses: Multivalued Attributes

7. Dashed Ellipses: Derived Attributes

8. Double Rectangles: Weak Entity Sets

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

EXAMPLE-2 (E-R DIAGRAM)



3. Normalization of Database

3.1 Functional Dependency

Functional dependency FD is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign → that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

3.2 Armstrong's Axioms

If F is a set of functional dependencies then the closure of F, denoted as F^+, is the set of all functional dependencies logically implied by F. Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

3.3 Reflexive rule If alpha is a set of attributes and beta is subset of alpha, then alpha holds beta.

3.4 Augmentation rule If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.

3.5 Transitivity rule if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

3.6 Additive- if a → b and a → c holds, then a → bc

3.7Projectivity- if a → bc then a → c and a → b

3.8Pseudotransitivity- if a → b and bc → w, then ac → w

4Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible. Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Update and Deletion Anomalies are very frequent if Database is not normalized

4.1 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.

4.2 Deletion anomalies We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.

4.3 Insert anomalies We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

4.4 Trivial Functional Dependency

4.4.1 Trivial If a functional dependency FD X → Y holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.

4.4.2 Non-trivial If an FD X → Y holds, where Y is not a subset of X, then it is called a non- trivial

4.4.3 Completely non-trivial If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completelynon-trivial FD.

5.Normalization Rule

Normalization rule are divided into following normal form.

1. **First Normal Form**
2. **Second Normal Form**
3. **Third Normal Form**
4. **BCNF**

Before normalization we must know the closure of x.

Example of closure

R (A, B, C, D) FDS = {AB→C, B→D} NOW the closure of AB IS AB^+ IS {ABC^+ } =ABCD
MINIMUMCOVER-given a set of FDS F, we say that it is no redundant if no proper subset F' OF F is equivalent of F.

CANONICAL COVER-A set of FD F_C is canonical cover if every FD in F_C SATISFIES the following

1. Each FD in F_C is simple

2. The FD in F_C are left reduced.

3. No FD x→a is redundant

5.1 **First Normal Form**

Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The **Primary key** is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table:

Student	**Age**		**Subject**	
dillip	35		Biology, Maths	

rohan	24	Maths
smita	32	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be:

Student	Age	Subject
dillip	35	Biology
dillip	35	Maths
rohan	24	Maths
smita	32	Maths

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique

5.2 Second Normal Form

Before we learn about the second normal form, we need to understand the following

5.2.1 ***Prime attribute*** An attribute, which is a part of the prime-key, is known as a prime attribute.

5.2.2 Non-prime attribute An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X, for which $Y \rightarrow A$ also holds true.

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the

table that is not

part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

New Student Table following 2NF will be:

Student	Age
dillip	35
rohan	24
smita	32

In Student Table the candidate key will be **Student** column, because all other column i.e. **Age** is dependent on it.

Example

R (A, B, C, D) FDS = {AB→C, B→D} NOW THERE IS A PARTIAL DEPENDANCY B→D, SO IT IS NOT 2NF.CANDIDATE KEY IS AB.

5.3 Third Normal Form

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy

No non-prime attribute is transitively dependent on prime key attribute.

For any non-trivial functional dependency, X → A, then either X is a super key or, A is prime attribute.

EXAMPLE-1

R (A, B, C, D) FDS = {AB→C, B→D} NOW THERE IS A PARTIAL DEPENDANCY B→D, SO IT IS NOT 2NF.CANDIDATE KEY IS AB.SO IT IS NOT

3NF EXAMPLE-2

R(A,B,C,D,E) FDS ={AB→C,C→D,D→E} NOW THERE IS A TRANSITIVE DEPENDANCY D→E ,SO IT IS NOT 3NF.CANDIDATE KEY IS AB.BUT IT IS 2NF

5.4 Boyce-Codd Normal Form

Boyce-Codd Normal Form BCNF is an extension of Third Normal Form on strict terms. BCNF states that

For any non-trivial functional dependency, X → A, X must be a super-key..

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple **overlapping candidate keys is said to be in BCNF.**

EXAMPLE

R (A, B) FDS = {A→B} NOW THERE IS NO PARTIAL DEPENDANCY AND TRANSITIVE DEPENDANCY .CANDIDATE KEY IS A.SO IT IS BCNF BECAUSE NO OVERLAPPING CANDIDATE KEY.

Lossless-A decomposition of relation schema R<S,F> INTO the relation schemes Ri(i<=n) is said to be a lossless join decomposition or simply lossless if for every relation (R) that satisfies the FDs in F,the natural join of the projections R give the original relation R.

Example: Let R (A, B, C) with F = {A→B} R decompose into R1 {A, B} and R2 {A, C}

NOW R is lossless because common attribute A is key of R1

5.5

set of functional dependencies on the attribute in S,R is decomposed into the relation schemes R1,R2 …..Rn with the functional dependencies F1…..Fn then this decomposition of R is dependency preserving if the closure of F'+ is the identical to F^{+}

Example: Let R(A,B,C,D) with F ={A→B,A→C,C→D} R decompose into R1{ABC} with FD{ A→B,A→C} and R2{CD} with FD {C→D}

is dependency preserving.

QUESTIONS

A. Very short answer type questions:

1) What is database and give its common use?
2) Why DBMS is needed?
3) What is naïve user?
4) Define database and DBMS.
5) Give the various advantages of database system.
6) Define database management system interfaces.
7) What is DBA?
8) List any three applications of DBMS.
9) What are end users?
10) What is system analyst?
11) Give advantages and disadvantages of network model.
12) Write any two differences between network and relational model.
13) What is centralized DBMS.
14) Define schema.
15) What is parallel DBMS?
16) Define record based model.
17) What is the difference between schema and subschema?

1) What is DBMS? What are its various characteristics?
2) What are the disadvantages of DBMS?

3) What is the role of database administrator?
4) How database designer is different from DBMS system designer?
5) How will you classify DBMS?
6) What are the characteristics of database?
7) What are the disadvantages of conventional file processing system?
8) Define database schema and instance with example.
9) Explain the three level architecture of DBMS.
10) Explain the two tier architecture of DBMS.
11) Discuss the concept of data independence in brief.
12) Explain various types of DBMS.
13) What is the difference between logical and physical data independence?
14) What is object based model?

C. Long answer type questions.
1) Differtiate between conventional system and database system.
2) Discuss the various types of data models in DBMS.
3) Discuss the various database languages used in DBMS.
4) Compare hierarchical data model, network data model and relational data model in detail.

Output Yields Article, Page, and Subject from the relation book, where subject is database.