Assignment 1:

[1. What are different types of Database users and their roles? Explain the Data independence with example. 2](#_Toc203076610)

[I. End Users: 2](#_Toc203076611)

[II. Application Programmers: 2](#_Toc203076612)

[III. Database Administrators (DBAs): 2](#_Toc203076613)

[Data Independence: 3](#_Toc203076614)

[I. Logical Data Independence: 3](#_Toc203076615)

[II. Physical Data Independence: 4](#_Toc203076616)

[2. What are the components of ER diagram? Explain the function of various symbols use in ER diagram. Construct an ER diagram to store data in a library of your college. 4](#_Toc203076617)

[I. Entity: 4](#_Toc203076618)

[II. Weak 4](#_Toc203076619)

[III. Attribute: 5](#_Toc203076620)

[IV. Key 5](#_Toc203076621)

[V. Multi-valued 5](#_Toc203076622)

[VI. Derived 5](#_Toc203076623)

[VII. Relationship: 5](#_Toc203076624)

[VIII. Weak 5](#_Toc203076625)

[IX. Line: 5](#_Toc203076626)

[X. Cardinality: 5](#_Toc203076627)

[ER diagram for a library in a college: 5](#_Toc203076628)

[I. Identify the entities: 5](#_Toc203076629)

[II. Set 5](#_Toc203076630)

[III. Identify 6](#_Toc203076631)

[I. Set 6](#_Toc203076632)

[IV. Join 7](#_Toc203076633)

[3. What is difference between logical data independence and physical data independence? 7](#_Toc203076634)

[4. Explain Relationship and Relationship sets with example. 8](#_Toc203076635)

[5. Retrieve the TName, SName, SPhone for "ABC" school using SQL from given relation as below. 9](#_Toc203076636)

[6. What are the advantages of using Database Management System over traditional filing system? Explain different data models with example. 10](#_Toc203076637)

[7. Explain the use of primary and foreign key in DBMS with example. What is the role of foreign key? 11](#_Toc203076638)

[8. Define data independence. Explain three-schema architecture. 11](#_Toc203076639)

[9. Consider a banking database with three labels and primary key underlined as given below: 11](#_Toc203076640)

[10. Write both relational algebra and SQL queries: 11](#_Toc203076641)

[11. What do you mean by Schema and Instance in DBMS? Explain both with examples. 11](#_Toc203076642)

1. What are different types of Database users and their roles? Explain the Data independence with example.

A database user is an individual, application, or system that interacts with a database to perform operations such as storing, retrieving, updating, or managing data. Each user is typically assigned specific access rights and privileges based on their role to ensure security and proper data management.

Database users can be categorized based on their level of interaction with the database system:

1. End Users:

End users are database users who interact with database by issuing commands from a terminal through predefined application programs to perform functions like create, retrieve, modify and delete. Example: Bank tellers using an interface to access customer accounts

1. Application Programmers:

Application programmers are database users who develop applications that interact with the database using programming languages and APIs like Access, FoxPro, COBOL, etc. These application programs are used by end users to operate on data. Example: Developers creating e-commerce websites that connect to product databases.

1. Database Administrators (DBAs):

DBAs are database users who maintains the database description in original form. It is responsible for overall control of the database system. Example: Database managers ensuring that the product database runs smoothly, securely, and efficiently—supporting developers, customers, and business operations.

The responsibilities of a DBA are:

1. **Schema definition and modification:**

The creation and modification of the original description of the database structure and the way that structure is reflected by the files of the physical database.

1. **Storage structure and access method definition:**

The DBA determines how data is physically stored on disk, including file organization, portioning and tablespace management. The DBA also chooses the best access methods for query efficiency through indexing strategies like B-tree, hashmap, bitmaps.

1. **Granting authorization for data access:**

Granting access to the database to different users.

1. **Routine maintenance:**

Making backup copies of the database and repairing damage to the database due to hardware or software failures or misuse.

Data Independence:

Data independence refers to the ability to modify the database schema at one level without affecting the schema at the next higher level.

****

Figure 1: Levels of Abstraction

The types of data independence are:

1. Logical Data Independence:

It is the ability to change the conceptual schema without affecting external schemas or application programs. It protects from changes in logical structure of data. Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data. Example: Adding a new entity (table), attribute (column), or relationship to the database without requiring changes to existing applications that don't use these new elements.

1. Physical Data Independence:

It is the ability to change the physical schema without affecting the conceptual schema. It allows tuning of the physical database for efficiency while permitting application programs to run as if no change had occurred. Example: Changing file organizations or storage structures (e.g., from B-trees to hash indexes), changing storage devices, all without requiring changes to the logical database design or applications.

Example:

Consider a university database with a STUDENT table:

* Original schema: STUDENT (ID, Name, Address, Major)
* Logical change: Adding a "Phone\_Number" column - existing applications that don't use phone numbers continue working (logical independence)
* Physical change: Creating an index on the Major field - this improves performance without changing how applications query the data (physical independence)

1. What are the components of ER diagram? Explain the function of various symbols use in ER diagram. Construct an ER diagram to store data in a library of your college.

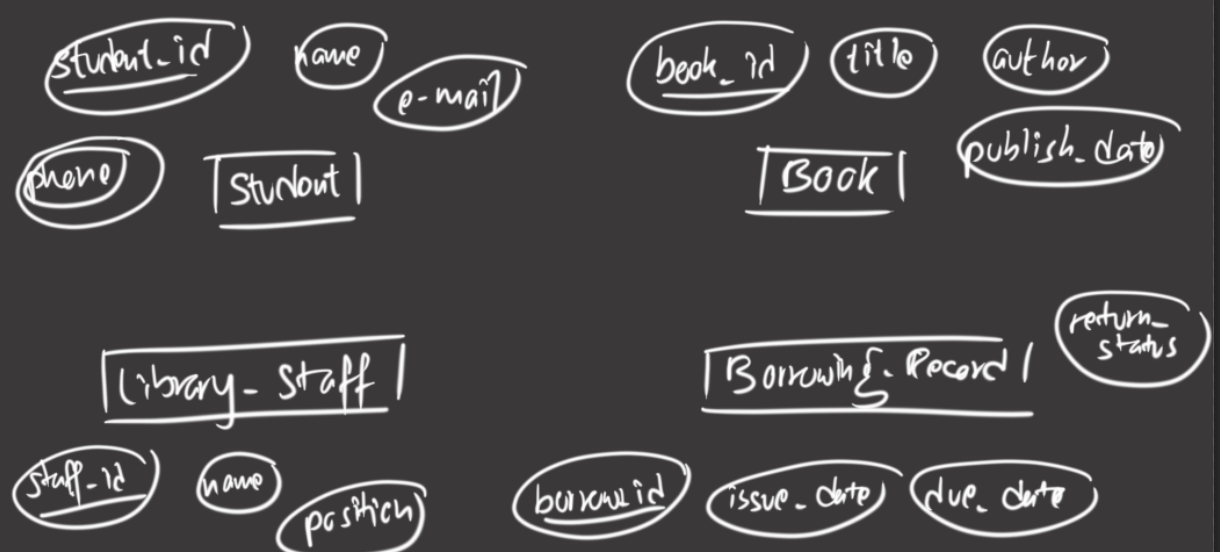
An Entity-Relationship (ER) Diagram is a visual representation of a database’s logical structure. It consists of entities, attributes, relationships, and associativity, each represented by specific symbols.

The various symbols used in ER diagram are:

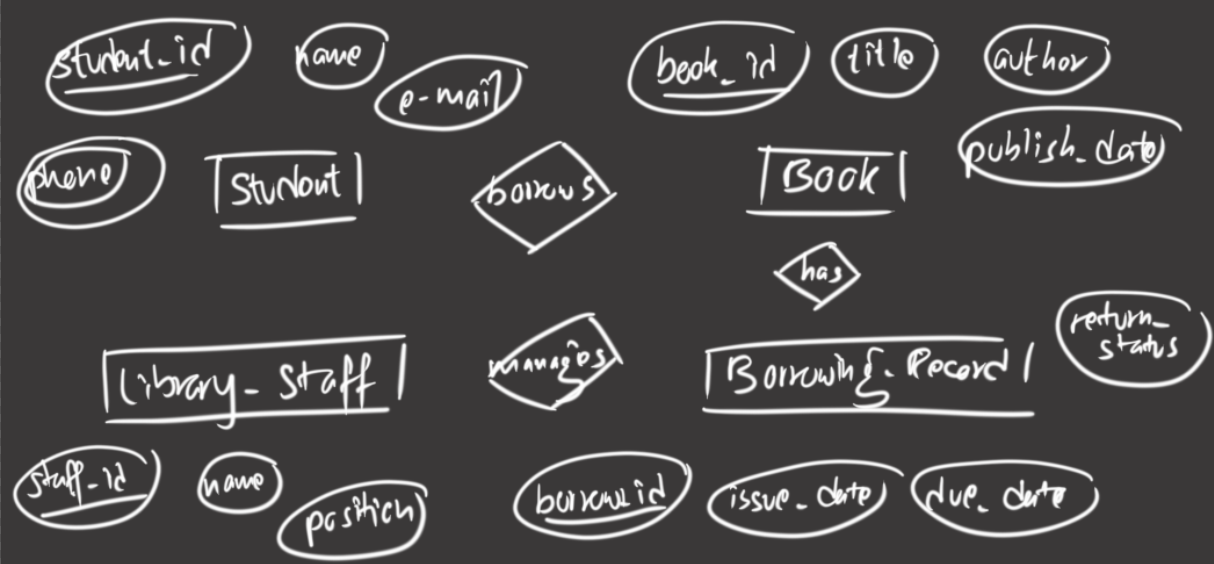
1. Entity: It represents a real-world object. It is the table in a database. Example: Student, Book, Library.
2. Weak Entity: It depends on another entity. It doesn’t have a primary key on its own. Example: Book\_Copy (needs Book).
3. Attribute: It describes an entity. It is the column in database. Example: student\_name, book\_id.
4. Key Attribute: It uniquely identifies an entity. It is also called primary key. Example: student\_id.
5. Multi-valued Attribute: It can have multiple values. Example: phone\_numbers.
6. Derived Attribute: It is computed from other attributes. Example: age (calculated from date\_of\_birth).
7. Relationship: It connects 2/more entities. Example: Borrows (Student->Book).
8. Weak relationship: It exists when a weak entity depends on a strong entity for its identity. Example: ‘Has’ is a weak relationship relating weak entity Book\_Copy with strong entity Book.
9. Line: It links attributes to entities and entities to relationships.
10. Cardinality: It defines how entities relate. Its types are one-to-one (1 Student : 1 Library\_Card), one-to-many (1 student: N books), many-to-one (N Borrowing\_Records : 1 Librarian) and many-to-many (M Student : N Book, a student can borrow many books and a book can be borrowed by many students over time).
11. ER diagram for a library in a college:
12. Identify the entities: Student, Book, Library\_Staff, Borrowing\_Record(weak entity)



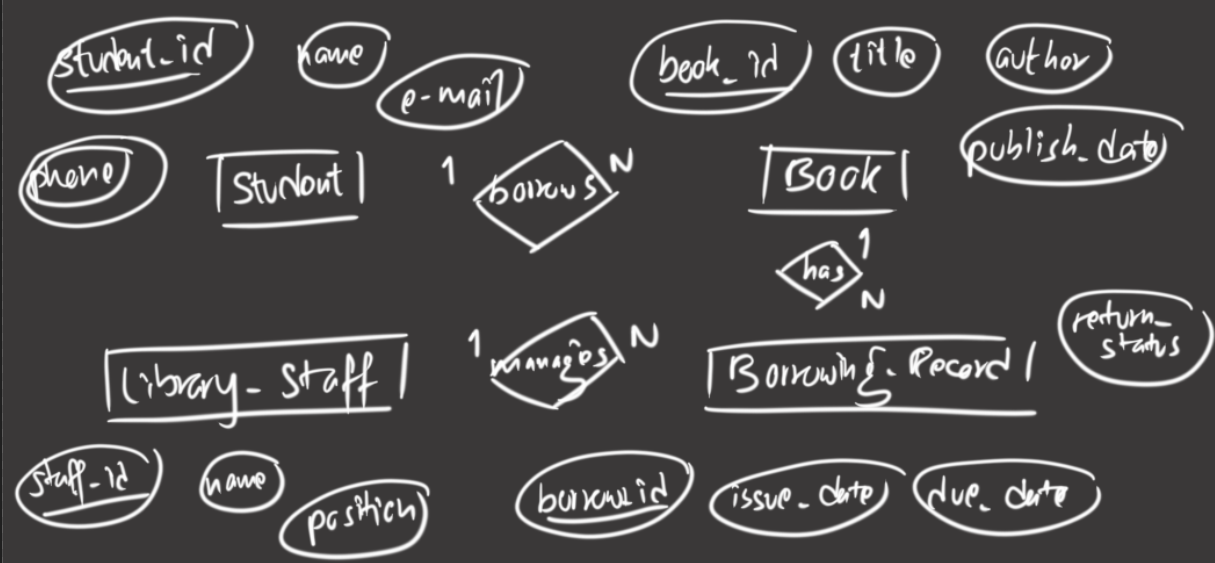
1. Set attributes:
2. Student: student\_id (PK), name, email, phone
3. Book: book\_id (PK), title, author, publish\_date
4. Library\_Staff: staff\_id (PK), name, position
5. Borrowing\_Record: borrow\_id (PK), issue\_date, due\_date, return\_status



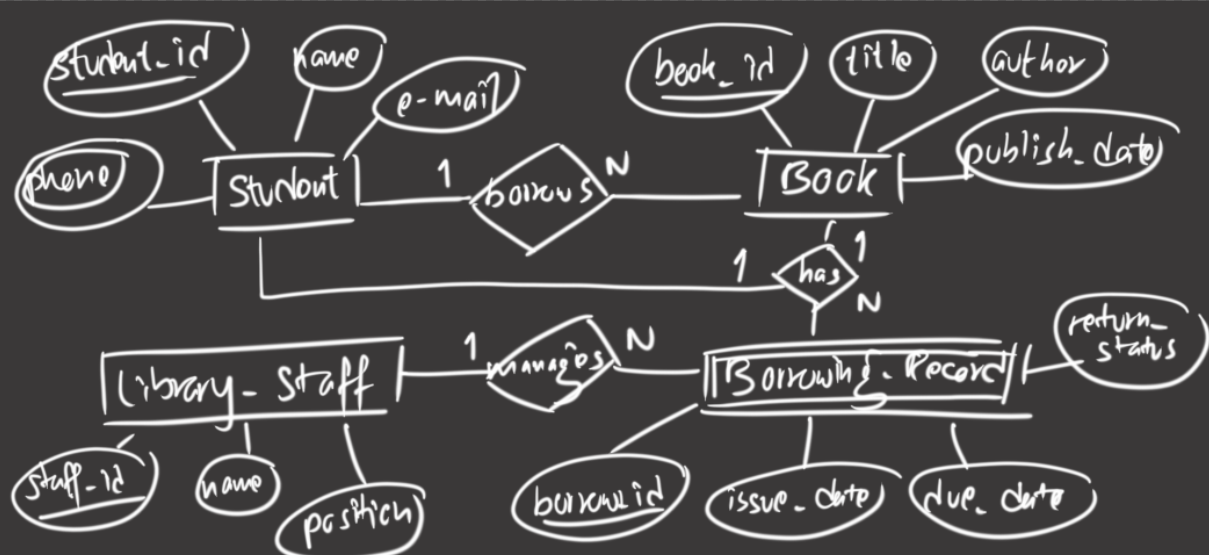
1. Identify relationship:
2. Student borrows Book.
3. Library\_Staff manages Borrowing\_Record
4. Book has Borrowing\_Record
5. Student has Borrowing Record



1. Set cardinality:
2. 1 Student borrows N Book.
3. 1 Library\_Staff manages N Borrowing\_Record
4. 1 Book has N Borrowing\_Record
5. 1 Student has N Borrowing\_Record



1. Join entities with relationship:



1. What is difference between logical data independence and physical data independence?

| **Aspect** | **Logical Data Independence** | **Physical Data Independence** |
| --- | --- | --- |
| **Definition** | Ability to change the conceptual schema without affecting external schemas (user views). | Ability to change the internal schema without affecting the conceptual schema. |
| **Level Affected** | Between conceptual and external levels. | Between internal and conceptual levels. |
| **What Changes?** | Modifications to tables, relationships, or constraints (e.g., adding/removing entities or attributes). | Changes in file structures, storage devices, or access methods (e.g., indexing, hashing). |
| **Impact on Users** | Applications not using the modified schema remain unaffected. | No impact on application programs or user queries. |
| **Purpose** | Protects applications from changes in the logical database design. | Protects the database design from changes in physical storage. |
| **Complexity** | Harder to achieve (may require view definitions). | Easier to achieve (handled by the DBMS). |
| **Example** | Adding a phone\_number column to a Student table without breaking existing apps that don’t use this field. | Switching from B-tree to hash indexing for faster searches without altering table structures. |

1. Explain Relationship and Relationship sets with example.

A relationship defines associationship between 2/more entities in a database. It is represented as a diamond (◇) in ER diagrams. It is defined by cardinality (1:1, 1:N, M:N).

Example:

* Entities: Student, Course
* Relationship: Enrolls
* Meaning: "A Student enrolls in a Course."

**ER Diagram:**

text

+----------+ +----------+

| Student |━━━━━┓ | Course |

+----------+ Enrolls

┗━━━━━┛

A relationship set is a collection of similar relationships between entity sets. It is analogous to a table in a relational database. Each row represents one relationship instance.

Formally it is a mathematical relation on n>=2 (possibly non-distinct) sets. If E1, E2,……..En are entity sets, then a relationship set R is a subset of {( e1, e2,…….., en ) | e1 Î E1, e2 Î E2 ,…….., en Î En } where ( e1, e2,…….., en ) is a relationship.

Example: University Database  
Entity Sets:

Student = {S1, S2, S3}

Course = {C101, C102}

Relationship Set Enrolls:

| **Student** | **Course** | **Enrollment\_Date** | **Grade** |
| --- | --- | --- | --- |
| S1 | C101 | 2023-09-01 | A |
| S2 | C101 | 2023-09-01 | B+ |
| S3 | C102 | 2023-09-05 | A- |

This table captures:

* S1 and S2 enrolled in C101.
* S3 enrolled in C102.

1. Retrieve the TName, SName, SPhone for "ABC" school using SQL from given relation as below.

        TEACHER(TID, TName, TAddress, TQualification)

        SCHOOL(SID, SName, SAddress, SPhone)

        SCHOOL\_TEACHER(SID, TID, No\_of Period).

CREATE DATABASE SCHOOL\_INFO;

USE SCHOOL\_INFO;

CREATE TABLE TEACHER (

TID INT PRIMARY KEY,

TName VARCHAR(30),

TAddress VARCHAR(10),

TQualification VARCHAR(10)

);

CREATE TABLE SCHOOL (

SID INT PRIMARY KEY,

SName VARCHAR(30),

SAddress VARCHAR(10),

SPhone NUMERIC(10)

);

CREATE TABLE SCHOOL\_TEACHER (

SID INT,

TID INT,

No\_of\_Period INT

);

INSERT INTO TEACHER VALUES

(1001, 'Heung-Min Son', 'South Korea', 'M.Sc.'),

(1002, 'Cristiano Ronaldo', 'Portugal', 'PhD'),

(1003, 'Marco Reus', 'Germany', 'M.Ed.'),

(1004, 'Mesut Ozil', 'Germany', 'B.Sc.'),

(1005, 'Jesse Lingard', 'England', 'M.A.');

INSERT INTO SCHOOL VALUES

(2001, 'ABC', '101 Maple St', 5551234567),

(2002, 'Swarnim', '203 Oak Ave', 5552345678),

(2003, 'Bella Pre-School', '304 Pine Rd', 5553456789),

(2004, 'Trinity College', '505 Birch Blvd', 5554567890),

(2005, 'NCCS', '607 Cedar Dr', 5555678901);

INSERT INTO SCHOOL\_TEACHER VALUES

(2001, 1001, 5),

(2001, 1002, 3),

(2002, 1003, 6),

(2003, 1004, 4),

(2001, 1005, 2),

(2001, 1002, 4),

(2004, 1003, 3),

(2005, 1005, 5),

(2005, 1001, 2);

SELECT T.TName,S.SName,S.SPhone FROM TEACHER AS T

JOIN SCHOOL\_TEACHER AS ST ON T.TID=ST.TID

JOIN SCHOOL AS S ON ST.SID=S.SID

WHERE S.Sname='ABC';

1. What are the advantages of using Database Management System over traditional filing system? Explain different data models with example.

A Database Management System (DBMS) is a software system designed to store, retrieve, manage, and manipulate data efficiently. It provides a structured way to organize data in databases, ensuring data integrity, security, and easy access.. A Traditional File System is a method of storing data in flat files (e.g., .txt, .csv, .dat) without a structured database. Each file contains records, but there is no relationship between files, leading to data redundancy and inconsistency.

The advantages of using DBMS over traditional filing system are given below:

1. Data Redundancy Control: DBMS minimizes duplication of data through normalization, while file systems often have redundant data across multiple files stored in different locations.
2. Data Consistency: Ensures all data conforms to rules and constraints, unlike file systems where consistency must be manually maintained.
3. Data Sharing: Allows concurrent access by multiple users/applications, while file systems typically allow only one user at a time.
4. Data Integrity: Provides mechanisms to maintain accuracy and validity of data through constraints and validation rules.
5. Security Features: Offers user authentication, authorization, and access controls at various levels.
6. Backup and Recovery: Built-in mechanisms for data backup and recovery from failures.
7. Data Independence: Separates physical storage from logical representation, allowing changes without affecting applications.
8. Efficient Query Processing: Provides powerful query languages (like SQL) for complex data retrieval.
9. Centralized Management: Single repository for data with centralized administration.
10. Concurrency Control: Manages simultaneous access by multiple users to prevent inconsistencies.
11. Data Model:

Data models are a collection of conceptual tools for describing data, relationships, data semantics and data constraints.

The types of data models are:

1. Object-based Logical Models:

The Object-Based Logical Model represents data as objects, similar to object-oriented programming concepts. It focuses on encapsulation, inheritance, and polymorphism, making it suitable for complex data structures.

1. E-R Model:

The entity-relationship model is a conceptual data modeling technique used to represent real-world data requirements in a structured way.. It helps in designing structure of database by defining entities having attributes and are connected via relationships, and constraints before actual implementation.

An entity is a distinguishable object that exists. Each entity is associated with a set of attributes describing it. A relationship is an association among several entities. The set of all entities or relationships of the same type is called the entity set or relationship set. Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set.

Example: For a university system,

1. Entities: Student, Course
2. Attributes: Student(sid,name,level,age), Course(cid,name,credits)
3. Relationsihp: Enrolled(sid,cid,grade)

The overall logical structure of a database can be expressed graphically by an E-R diagram:

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 2: ER database model

The symbols used in ER diagram are:

1. Rectangles: represent entity sets.
2. Ellipses: represent attributes.
3. Diamonds: represent relationships among entity sets.
4. Lines: link attributes to entity sets and entity sets to relationships.

The advantages of ER model are:

1. Easy to Understand: Visual representation helps non-technical users.
2. Database Design: Serves as a blueprint before SQL implementation.
3. Reduces Ambiguity: Clearly defines relationships and constraints.
4. Flexible: Can be extended (EER) for complex scenarios.

The limitations of ER model are:

1. No Standard Notation: Different tools use different symbols (Chen, Crow’s Foot, UML).
2. No Query Support: Only a design tool, not executable.
3. Limited to Conceptual Level: Does not optimize storage/physical design.
4. Object-Oriented Model:

The Object-Oriented Data Model (OODM) is a database model that integrates object-oriented programming (OOP) concepts with database systems. It represents data as objects, similar to how objects work in programming languages like Java, C++, or Python.

The object-oriented data model stores data as objects, each containing both data (like student and course) and functions (called methods) that work on that data. Objects are grouped into classes, which act as templates. To access or change an object’s data, we use its methods, keeping the internal details hidden. This makes it easy to update behavior, without rewriting the whole program. Objects can also include other objects, allowing complex, nested structures. Even if two objects have the same data, they are treated as separate because each has a unique identity. This model is useful for handling complex, real-world data and works well with object-oriented programming.

1. Record-based logical models:

The record-based logical model organizes data into fixed-format records, much like rows in a table. Each record is made up of fields (also called attributes), and each field holds a value of a specific type. This model is used to describe how data is structured at the logical level and is widely used in database systems.

1. Relational Model:

Data and relationships are represented by a collection of tables(called relations). Each table has rows (records/tuples) and columns (attributes).



Figure 3: Relational database model

1. Network Model:

Data is stored as records connected by links (like a graph). Relationships are represented by pointers. Example: A Customer linked to multiple Accounts using direct connections.



Figure 4: Network database model

1. Hierarchical Model:

Data is organized in a tree structure. Each parent can have many children, but each child has only one parent. Example: A Department has many Employees.The relational model does not use pointers or links, but relates records by the values they contain. This allows a formal mathematical foundation to be defined.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 5: Hieraarchical database model

1. Physical Data Models:

The physical data model shows how data is actually stored on a computer system—like on hard drives, memory, or storage blocks. It deals with low-level details such as: file formats, indexes, block sizes, access methods (e.g., how fast data is read/written). It is used by database administrators (DBAs) and system engineers to optimize performance. Example: A student record might be stored in a binary file using fixed-length blocks, and indexed by student ID to allow fast search.

1. Unifying model: Focuses on representing all types of data structures (like hierarchical, network, and relational) in a single unified way.
2. Frame memory: Represents data in frames, which are blocks or chunks of memory. Each frame stores both data and metadata about the structure of the data.

1. Self-describing Data Models:

A self-describing model is a asemi-strucxtued data model means that the data comes with its own description (metadata).This metadata describes: structure of the data, data types, relationships among data. Example: In a relational database, the system stores information about tables (like their column names and data types) in data dictionaries, so users don’t need to define them separately.

1. Explain the use of primary and foreign key in DBMS with example. What is the role of foreign key?

A primary key is a field (or combination of fields) in a table that uniquely identifies each row (tuple). It must not contain null values, and its value must be unique across all rows.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6: Primary key

Here, roll is the primary key because it uniquely identifies each student.

A foreign key is a field (or combination of fields) in one table that refers to the primary key in another table. It establishes a relationship between two tables. A foreign key can have duplicate values and can be NULL (unless restricted).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7: Foreign key

Roll in ENROLLMENT is a foreign key referencing Roll in the STUDENT table. It ensures that only valid students (existing in STUDENT) can enroll in courses.

SQL code:

CREATE TABLE STUDENT (

RollNo INT PRIMARY KEY,

Name VARCHAR(50),

Major VARCHAR(50)

);

-- Create ENROLLMENT table

CREATE TABLE ENROLLMENT (

RollNo INT,

Course VARCHAR(50),

Grade CHAR(1),

PRIMARY KEY (RollNo, Course),

FOREIGN KEY (RollNo) REFERENCES STUDENT(RollNo)

);

Role of foreign key:

Maintains referential integrity: ensures that the relationship between tables remains consistent.

Prevents orphan records: you can't insert an enrollment for a student who doesn't exist.

Controls cascading actions: on deleting or updating referenced rows, actions like CASCADE, SET NULL, or RESTRICT can be applied.

1. Define data independence. Explain three-schema architecture.

Data independence:

Data independence is the ability to modify the schema at one level of a database system without affecting the schema at the next higher level.

There are two types:

Physical Data Independence:

Changes in physical storage (e.g., indexing, file structure) do not affect the logical (conceptual) schema. Example: Changing from a heap file to a B+ tree index.

Logical Data Independence:

Changes in the conceptual schema (e.g., adding/removing attributes or tables) do not affect the external schemas or application programs. Example: Adding a new column to a table without affecting user views. Logical data independence is harder to achieve than physical.

Three-Schema Architecture:

The three-schema architecture is a framework to separate the database into three levels of abstraction. Its purpose is to seaparte user applications from physical database, to allow data indepenece and to simplify database management, maintenance and security.

****

Figure 8: 3-schema architecture

The layers of abstraction are:

1. Internal Level (Physical Schema):

It is the lowest level of abstraction. It describes how data is physically stored in the database. Includes file structures, indexes, compression, storage devices, etc. Example: Student data is stored as binary records in files on a hard disk, sorted by roll.

2. Conceptual Level (Logical Schema):

It is the midlle level of abstraction. It describes the structure of the entire database for a community of users. It is independent of physical storage through logical data independence .It includes entities, relationships, constraints(primary & foreign keys), data types, etc. Example: Student(Roll, Name, Major) is a logical table, with Roll as primary key and possibly foreign keys to other tables.

3. External Level (View Schema):

It is the highest level of abstraction. It describes how individual users or applications see the data. Each user can have a customized view. It allows access control, simplification, and security. Example:

A teacher’s view: Student(RollNo, Name)

A finance department’s view: Student(RollNo, FeeStatus)

1. Consider a banking database with three labels and primary key underlined as given below:

Customer (CustomerID , CustomerName, Address, Phone, Email)

Borrows (CustomerID, LoanNumber )

Loan ( LoanNumber , LoanType, Amount )

Write both relational algebra and SQL queries:

1. To display name of all customers who live in “Lalitpur” in ascending order of name.
2. To count total number of customers having loan at the bank.
3. To find name of those customers who have loan amount greater than or equal to 500000.
4. To find average loan amount of each account’s type.

CREATE DATABASE Bank;

USE Bank;

CREATE TABLE Customer (

CustomerID INT(5) PRIMARY KEY,

CustomerName VARCHAR(30),

Address VARCHAR(15),

Phone NUMERIC(10),

Email TEXT

);

CREATE TABLE Loan (

LoanNumber INT(5) PRIMARY KEY,

LoanType VARCHAR(10),

Amount FLOAT(10,2)

);

CREATE TABLE Borrows (

CustomerID INT(5),

LoanNumber INT(5),

PRIMARY KEY (CustomerID, LoanNumber),

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (LoanNumber) REFERENCES Loan(LoanNumber)

);

INSERT INTO Customer VALUES

(10001, 'John Doe', 'Lalitpur', 5551234567, 'johndoe@example.com'),

(10002, 'Jane Smith', 'Lalitpur', 5552345678, 'janesmith@example.com'),

(10003, 'Mark Johnson', '789 Pine Rd', 5553456789, 'markj@example.com'),

(10004, 'Emily Davis', '101 Maple Dr', 5554567890, 'emilydavis@example.com'),

(10005, 'David Lee', 'Lalitpur', 5555678901, 'davidlee@example.com');

INSERT INTO Loan VALUES

(20001, 'Personal', 500000.00),

(20002, 'Home', 1500000.00),

(20003, 'Car', 1200000.00),

(20004, 'Personal', 3050000.00),

(20005, 'Student', 80000.00),

(20006, 'Home', 2000000.00);

INSERT INTO Borrows VALUES

(10001, 20001),

(10002, 20002),

(10002, 20003),

(10003, 20004),

(10004, 20005),

(10005, 20006);

DELETE FROM Customer;

DELETE FROM Loan;

DELETE FROM Borrows;

SELECT CustomerName FROM Customer WHERE Address='Lalitpur' ORDER BY CustomerName ASC;

πCustomerName (σAddress = 'Lalitpur' (Customer))

SELECT COUNT(DISTINCT CustomerID) AS TotalCustomersWithLoans

FROM Borrows;

COUNT (πCustomerID (Borrows))

SELECT DISTINCT c.CustomerName

FROM Customer AS c

JOIN Borrows AS b ON c.CustomerID = b.CustomerID

JOIN Loan AS l ON b.LoanNumber = l.LoanNumber

WHERE l.Amount >= 500000;

πCustomerName ( σAmount ≥ 500000 ((Customer ⨝ Borrows) ⨝ Loan))

SELECT LoanType, AVG(Amount) AS AvgLoanAmount

FROM Loan

GROUP BY LoanType;

γLoanType, AVG(Amount) (Loan)

1. What do you mean by Schema and Instance in DBMS? Explain both with examples.

Schema:

A schema is the overall design or blueprint of the database. It defines the structure: tables, attributes, data types, relationships, constraints, etc. It is fixed (or changes rarely). It acts like a plan or template of how the data is organized. Example:

STUDENT(RollNo INT, Name VARCHAR(50), Major VARCHAR(30))

This is a schema of the STUDENT table. It defines that each student has a roll number, name, and major.

Instance:

An instance is the actual content of the database at a particular moment in time. It refers to the current set of rows (records) stored in the database. It changes frequently due to insert, delete, or update operations.

Example:

STUDENT Table Instance (Current Data)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 9: Instance of Student table

This is an instance of the STUDENT table at a certain time.