MAILAM
Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai, Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

AD3391 DATABASE DESIGN AND MANAGEMENT

II YEAR / III SEM

SYLLABUS

COURSE OBJECTIVES:

- To introduce database development life cycle and conceptual modeling
- To learn SQL for data definition, manipulation and querying a database
- To learn relational database design using conceptual mapping and normalization
- To learn transaction concepts and serializability of schedules
- To learn data model and querying in object-relational and No-SQL databases

UNIT I CONCEPTUAL DATA MODELING **8**

Database environment – Database system development lifecycle – Requirements collection – Database design – Entity-Relationship model – Enhanced-ER model – UML class diagrams.

UNIT II RELATIONAL MODEL AND SQL **10**

Relational model concepts – Integrity constraints – SQL Data manipulation – SQL Data definition – Views – SQL programming.

UNIT III RELATIONAL DATABASE DESIGN AND NORMALIZATION **10**

ER and EER-to-Relational mapping – Update anomalies – Functional dependencies – Inference rules – Minimal cover – Properties of relational decomposition – Normalization (upto BCNF).

UNIT IV TRANSACTION MANAGEMENT **8**

Transaction concepts – properties – Schedules – Serializability – Concurrency Control – Two phase locking techniques

UNIT V OBJECT RELATIONAL AND NO-SQL DATABASES **9**

Mapping EER to ODB schema – Object identifier – reference types – row types – UDTs – Subtypes and supertypes – user-defined routines – Collection types – Object Query Language; No – SQL: CAP theorem – Document-based: MongoDB data model and CRUD operations; Column – based: Hbase data model and CRUD operations.

TOTAL: 45 PERIODS

COURSE OUTCOMES

After the completion of this course, students will be able to:

CO1: Understand the database development life cycle and apply conceptual modeling

CO2: Apply SQL and programming in SQL to create, manipulate and query the database

CO3: Apply the conceptual-to-relational mapping and normalization to design relational database

CO4: Determine the serializability of any non-serial schedule using concurrency techniques

CO5: Apply the data model and querying in Object-relational and NO-SQL databases.

TEXT BOOKS

1. Thomas M. Connolly, Carolyn E. Begg, Database Systems – A Practical Approach to Design, Implementation, and Management, Sixth Edition, Global Edition, Pearson Education, 2015.
2. Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson, 2017.

REFERENCES

1. Toby Teorey, Sam Lightstone, Tom Nadeau, H. V. Jagadish, "DATABASE MODELING AND DESIGN – Logical Design", Fifth Edition, Morgan Kaufmann Publishers, 2011.
2. Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Ninth Edition, Cengage learning, 2012.
3. Abraham Silberschatz, Henry F Korth, S Sudharshan, "Database System Concepts", 6th Edition, Tata Mc Graw Hill, 2011.
4. Hector Garcia-Molina, Jeffrey D Ullman, Jennifer Widom, "Database Systems: The Complete Book", 2nd edition, Pearson.
5. 5. Raghu Ramakrishnan, "Database Management Systems", 4th Edition, Tata Mc Graw Hill,2010.

S. Jayabharathi
STAFF INCHARGE

BC

HOD

PRINCIPAL

NOV/DEC - 2023, APRIL/MAY 2024 UNIVERSITY QP UPDATION

Unit	Part – A				Part – B				Part – C		
	Q. No	Page No.	Q. No	Page No.	Q. No	Page No.	Q. No	Page No.	Q. No	Page No.	
01	28,29	7,7	8,30	2,7	21	53,54	8	20	4	29	NOV/DEC-2023, APR /MAY 2024
							23	56			
					3,8	12,20	5,3	14,10			
02	1,7&17	1,3&6	10	48	6	32	4	18	10	50	NOV/DEC-2023, APR /MAY 2024
					11	53	4	30			
03	14,6	5,3	20	7	14	28	4	15	14	28	NOV/DEC-2023, APR /MAY 2024
					3,4	13,15	9	24			
04	2,17	1,6	1,11	1,3	16	37	13	27	-	-	NOV/DEC-2023, APR /MAY 2024
					13	30	14	33	-	-	
05	25,26	7,7	28,27	7,7	1	7	1	7	-	-	NOV/DEC-2023, APR /MAY 2024
					10	29	12	35			

S. Jayabharathi

PREPARED BY

1. Mrs. S. Jayabharathi, AP/AI&DS

BATCH COORDINATOR

Mrs. G. Vasanthi, AP/AI&DS

VERIFIED BY

Dr. S. Artheeswari
HOD/AI&DS

PRINCIPAL

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai,

Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / III SEM

UNIT I – CONCEPTUAL DATA MODELING

SYLLABUS:

Database environment – Database system development lifecycle – Requirements collection – Database design – Entity-Relationship model – Enhanced-ER model – UML class diagrams.

PART A

1. What is Data?

- Data is the raw material that can be processed for any computing machine. **For example** – Employee name, Product name, Name of the student, Marks of the student, Mobile number, Image etc.

2.Define an information.

- Information is the data that has been converted into more useful or intelligent form.
- For example: Report card sheet.
- The information is needed for the following reasons –
  1. To gain knowledge about the surroundings.
  2. To keep the system up to date.
  3. To know about the rules and regulations of the society.

3.Define Knowledge.

- The human mind purposefully organizes the information and evaluates it to produce knowledge.

4.Give the Example of data, information and knowledge.

- A student secures 450 marks. Here 450 is data, marks of the student are the information and hard work required to get the marks is knowledge.

PREPARED BY: Mrs. S. JAYABHARATHI AP/AI&DS

5.What is Database?

- Database is a collection of related data and data is a collection of facts and Figures that can be processed to produce information.
- Mostly data represents recordable facts.
- Data aids in producing information, which is based on facts.
- **For example,** if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

6.What is DBMS?

- **Definition:** A database management system (DBMS) is collection of information data and various programs that are used to handle the data.
- The primary goal of DBMS is to provide a way to store and retrieve the required information from the database in convenient and efficient manner.
- For managing the data in the database two important tasks are conducted-
  1. Define the structure for storage of information.
  2. Provide mechanism for manipulation of information.

7.List the characteristics of Database Systems.

- Following are the characteristics of database system:
  1) Representation of some aspects of real-world applications
  2) Systematic management of information.
  3) Representing the data by multiple views.
  4) Efficient and easy implementation of various operations such as insertion, deletion and updation.
  5) It maintains data for some specific purpose.

8.List any two advantages of database system. (APR/MAY 2024)

- In addition, the database systems must ensure the safety of information stored.
- Following are the advantages of DBMS –
  1) DBMS removes the data redundancy that means there is no duplication of data in database.
  2) DBMS allows to retrieve the desired data in required format.
  3) Data can be isolated in separate tables for convenient and efficient use.
  4) Data can be accessed efficiently using a simple query language.

9.Is it possible for several attributes to have same domain? Illustrate your answer with suitable example.

- A domain is the set of legal values that can be assigned to an attribute.
- Each attribute in a database must have a well-defined domain; we can't mix values from different domains in the same attribute.
- Hence, it is not possible for several attributes to have same domain.
- **For example -** Student domain has attributes RollNo, Name, Address. Similarly, Employee domain has EmpID, Ename, Salary, Address.
- We cannot define the same domain for defining several attributes.

10. What is data model?

- Data model is a structure below the database.
- It is a collection of conceptual tools for describing data, relationships among data, semantics (meaning) of data and constraints.

11.What are different types of data models?

- Relational Data Model
- Entity relational Data model
- Object based Data Model
- Semi-structured Data model

12. Define data independence

- Data independence is an ability by which one can change the data at one level without affecting the data at another level.
- Here level can be physical, conceptual or external.

13.What is meant by instance and schema of the database?

- The collection of information at particular moment is called instances.
- The overall design of the database is called schema.

14.Explain entity relationship model.

- The ER Data model specifies enterprise schema that represents the overall logical structure of a database.
- The ER model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.

15.Give the limitations of E-R model? How do you overcome this?

- **Loss of information content:** Some information be lost or hidden in ER Model.
- **Limited relationship representation:** ER model Represents limited relationship as compared to another data models like relational model etc.
- **No representation of data manipulation:** It is difficult to show data manipulation in ER model.
- **Popular for high level design:** ER model is very popular for designing high level design

16.What is an entity?

- An entity is an object that exists and is distinguishable from other objects.
- **For example -** Student named "ARUN" is an entity and can be identified by her name.
- Entity is represented as a box, in ER model.

17.What do you mean by derived attributes?

- Derived attributes are the attributes that contain values that are calculated from other attributes.
- To represent derived attribute there is dotted ellipse inside the solid ellipse.
- **For example -**Age can be derived from attribute DateOfBirth. In this situation, DateOfBirth might be called Stored Attribute.

18.What is a weak entity?

- An Entity type has a key attribute that uniquely identifies each entity in the entity set.
- But some entity type exists for which key attributes can't be defined.
- These are called Weak Entity types.

19.Define the term relationship set.

- The relationship set is a collection of similar relationships. For example – Following Figure1.1. shows the relationship works_for for the two entities Employee and Departments.

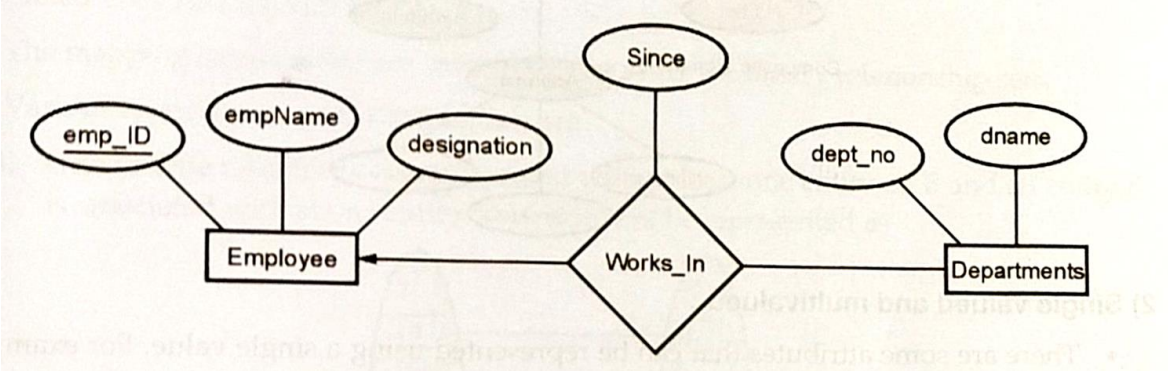


Figure 1.1 Relationship Set

20.Define the terms i) Key attribute ii) Value set.

Key attribute:

- An entity type usually has an attribute whose values are distinct from each individual entity in the collection. Such an attribute is called a key attribute.

Value set:

- Each simple attribute of an entity type is associated with a value set that specifies the set of values that may be assigned to that attribute for each individual entity.

21.Define weak and strong entity sets.

Weak entity set: An Entity set that do not have key attribute of their own are called weak entity sets.

Strong entity set: An Entity set that has a primary key is termed a strong entity set

22.Define the terms i) DDL ii) DML.

DDL: Data base schema is specified by a set of definitions expressed by special language called a data definition language.

DML: A data manipulation language is a language that enables users to access or manipulate data as organized by the appropriate data model.

23.Write short notes on relational model.

- The relational model uses a collection of tables to represent both data and the relationships among those data.
- The relational model is an example of a record-based model.

24. Define UML diagram

- The Unified Modeling Language (UML) is a standard diagramming notation for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

25. List the various element in UML diagram

- Various elements of the class diagram are –
    1. Classifier
    2. Attributes and association lines
    3. Methods or operations
    4. Generalization
    5. Dependency
    6. Composition and aggregation

26.What is classifier in UML?

- The classifier represents the behavioural and structural features of the system.
- It can be specialized.
- The various elements in classifier are classes, interfaces, use cases and actors. In class diagram the most commonly used classifiers are classes and interfaces.

27.Define superclass and subclass in ER Model

Superclass: An entity type that represents a general concept at a high level, called superclass.

Subclass: An entity type that represents a specific concept at lower levels, is called subclass

28.What are the Disadvantages of Traditional File Processing System? (NOV/DEC 2023)

- Data Redundancy
- Data Inconsistency
- Difficulty in accessing data
- Data Isolation
- Integrity problem
- Atomicity problem
- Security problem
- Concurrent access anamolies

29.List the applications of DBMS. (NOV/DEC 2023)

- Accounting
- Manufacturing
- Banking
- Universities
- Reservation systems.
- Telecommunication

30. State the different types of Integrity constraints used in designing a relational database. (APR/MAY 2024)

- Domain Integrity constraints
- Entity Integrity constraints
- Referential Integrity constraints
- Key Integrity constraints

PART B

1.Write about Database Systems Applications. Discuss about Traditional File Based Systems and their Disadvantages.

Database Systems Applications

- There are wide range of applications that make use of database systems. Some of the applications are –

  1) Accounting: Database systems are used in maintaining information employees, salaries, and payroll taxes.

  2) Manufacturing: For management of supply chain and tracking production of items in factories database systems are maintained.

  3) For maintaining customer, product and purchase information the databases are used.

  4) Banking: In banking sector, for customer information, accounts and loan and for performing banking applications the DBMS is used.

  5) For purchase on credit cards and generation of monthly statements database systems are useful.

  6) Universities: The database systems are used in universities for maintaining student salaries, and payroll taxes.

  7) Reservation systems: In airline/railway reservation systems, the database is used to maintain the reservation and schedule information.

  8) Telecommunication: In telecommunications for keeping records of the calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about communication networks the database systems are used.

Traditional File Based Systems

- Earlier database systems are created in response to manage the commercial data.
- These data are typically stored in files. To allow users to manipulate these files various programs are written for

  1) Addition of new data

  2) Updating the data

  3) Deleting the data.

- As per the addition of new need, separate application programs were required to write. Thus, as the time goes by, the system acquires more files and more application programs.
- This typical file processing system is supported by conventional operating system. Thus, the file processing system can be described as -
- The system that stores the permanent records in files and it needs different application programs to extract or add the records.

Disadvantages of Traditional File Processing System

- Before introducing database management system, this file processing system was in use.
- However, such a system has many drawbacks. Let us discuss them –The traditional file system has following disadvantages:

1.Data redundancy

- Data redundancy means duplication of data at several places. Different programmers create different files and these files might have different structures, there are chances that some information may appear repeatedly in some or more format at several places.

2) Data inconsistency:

- Data inconsistency occurs when various copies of same data may no longer get matched. For example, changed address of an employee may be reflected in one department and may not be available (or old address present) for another department.

3) Difficulty in accessing data

- The conventional file system does not allow to retrieve the desired data in efficient and convenient manner

4) Data isolation

- As the data is scattered over several files and files may be in different formats, it becomes to retrieve the desired data from the file for writing the new application

5) Integrity problems

- Data integrity means data values entered in the database all within a specified range and are of correct format. With the use of several files enforcing such constraints on the data becomes difficult.

6) Atomicity problems:

- An atomicity means particular operation must be carried out entirely or not at all with the database. It is difficult to ensure atomicity in conventional file processing system.

7) Concurrent access anomalies:

- For efficient execution, multiple users update data simultaneously, in such a case data need to be synchronized. As in traditional file Systems, data is distributed over multiple files, one cannot access these files concurrently.

8) Security problems:

- Every user is not allowed to access all the data of database system. Since application program in file system are added in an ad hoc manner enforcing such security constraints become difficult.

2.Discuss the Advantages and Disadvantages of Database systems. Also, differentiate between Database systems and Conventional file systems.

Advantages of Database Systems

1. DBMS removes the d duplication of data in data redundancy that means there is no duplication of data in database.
2. DBMS allows to retrieve the desired data in required format.
3. Data can be isolated in separate tables for convenient and efficient use.
4. Data can be accessed efficiently using a simple query language.
5. Data integrity can be maintained. That means the constraints can be applied on data and it should be in some specific range.
6. The atomicity of data can be maintained. That means, if some operation is performed on one particular table of the database, then the change must be reflected for the entire database.
7. The DBMS allows concurrent access to multiple users by using the synchronization technique.
8. The security policies can be applied to DBMS to allow the user to access only desired part of the database system.

Disadvantages of Database Systems

1) Complex design: Database design is complex, difficult and time consuming.

2) Hardware and software cost: Large amount of investment is needed to setup the required hardware or to repair software failure.

3) Damaged part: If one part of database is corrupted or damaged, then entire database may get affected.

4) Conversion cost: If the current system is in conventional file system and it, we need to convert it to database systems then large amount of cost is incurred in purchasing different tools, and adopting different techniques as per the requirement.

5) Training: For designing and maintaining the database systems the people need to be trained.

Database systems	**Conventional file systems**
Data redundancy is less	Data redundancy is more
Security is high	Security is very low
Database systems are used when security constraints are high	Conventional file systems are used where there is less demand for security constraints.
Database systems define the data in a structured manner. Also, there is well defined co-relation among the data.	File systems define the data in un-structured manner. Data is usually isolated form.
Data inconsistency is less in database systems.	Data inconsistency is more in database systems
User is unknown to the physical address of the data used in database systems. We can retrieve the data in any desired format using database system.	User locates the physical address of file to access the data in conventional file systems.
We can retrieve the data in any desired format using database system.	We cannot retrieve the data in any desired format using file systems.
There is ability to access the data concurrently using database systems.	There is no ability to access the data concurrently using database systems.

3.Describe the Three Schema Architecture or Illustrate in detail about the three levels (External, Conceptual, Internal) of database architecture. (APR/MAY 2023) (NOV/DEC 2023)

Definition:

- Database schema is a collection of database objects like tables, views, indexes and so on associated with one particular database username.
- This username is called the schema owner.
- For example, Student Schema can be owner of STUDENT and MARKS tables.
- The Course schema can be the owner of SUBJECT table.
- The goal of three-schema architecture is to separate the user application from the physical database.
- The architecture of database is divided into three levels based on three types of schemas - internal schema, conceptual schema or external schema as in Figure 1.2.

1. Internal level:

- It contains internal schema.
- This schema represents the physical storage structure of database.
- This schema is maintained by the software and user is not allowed to modify it.
- This level is closest to the physical storage.
- It typically describes the record layout of the files and types of files, access paths etc.

2. Conceptual level:

- It contains conceptual schema.
- This schema hides the details of internal level. This level is also called as logical level as it contains the constructs used for designing the database.
- It contains information like table name, their columns, indexes and constraints, database operations.
- A representational data model is used to describe conceptual schema when a database system is implemented.

3. External level:

- It contains the external schema or user views. At this level, the user will get to see only the data stored in the database.

- Either they will see whole data values or any specific records. They will not have any information about how they are stored in the database.

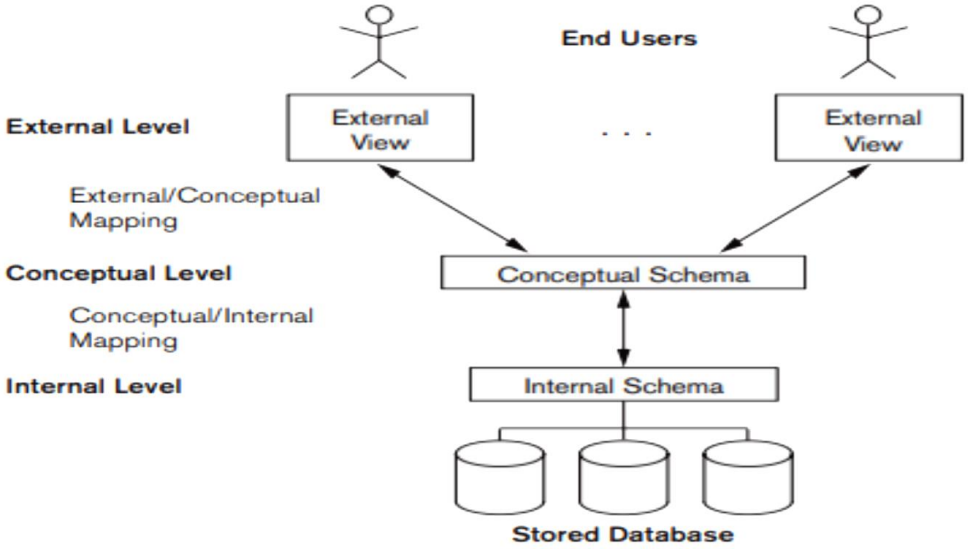


Figure 1.2 Three schema architecture

- The processes of transforming requests and results between levels are called mappings.
- In the three-schema architecture there are two mappings -
  1) External - Conceptual Mapping
  2) Conceptual - Internal Mapping

4.Give the concept about Instances and Schema

Schema: The overall design of the database is called schema. **For example -** In a program we do variable declaration and assignment of values to the variable. The variable declaration is called schema and the value assigned to the variable is called instance, The schema for the student record can be

RollNo	Name	Marks

Example of student schema

Instances: When information is inserted or deleted from the database then the database gets changed. The collection of information at particular moment is called instances. **For example -** following is an instance of student database.

RollNo	Name	Marks
10	ARUN	98
20	Balu	97

Example of instance

Types of Schemas:

- The database has several schemas based on the levels of abstraction.
  1. **Physical Schema:** The physical schema is a database design described at the physical level of abstraction.
  2. **Logical Schema:** The logical schema is a database design at the logical level of abstraction.
  3. **Subschema:** A database may have several views at the view level which are called subschemas.

5.Write about Database Languages.

- There are two types of languages supported by database systems. These are -

DDL

- **Data Definition Language (DDL)** is a specialized language used to specify database schema by a set of definitions.
- It is a language which is used for creating and modifying the structures of tables, views, indexes and so on.
- DDL is also used to specify additional properties of data.
- Some of the common commands used in DDL are –
  1. CREATE
  2. ALTER
  3. DROP.
- The main use of CREATE command is to build a new table, Using ALTER command, the users can add up some additional column and drop existing columns.
- Using DROP command, the user can delete table or view.

DML

- DML stands for Data Manipulation Language.
- This language enables users to access or manipulate data as organized by appropriate data model.
- The types of access are -.
  1. **Retrieval** of information stored in the database
  2. **Insertion** of new information into the database.
  3. **Deletion** of information from the database.
  4. **Modification** of information stored in database.
- There are two types of DML-
  1. **Procedural DML -** Require a user to specify what data are needed and how to get those data.
  2. **Declarative DML -** Require a user to specify what data are needed without specifying how to get those data.
- Query is a statement used for requesting the retrieval of information. This retrieval of information using some specific language is called query language.

6.Write about Data Models and Conceptual Modeling in detail

- **Definition-** It is a collection of conceptual tools for describing data, relationships among data, semantics (meaning) of data and constraints.
- Data model is a structure below the database.
- Data model provides a way to describe the design of database at physical, logical and view level.
- There are various data models used in database systems and these are as follows -

(1) Relational model

- Relation model consists of collection of tables which stores data and also represents the relationship among the data.
- Table is also known as relation.
- The table contains one or more columns and each column has unique name.
- Each table contains record of particular type, and each record type defines a fixed number of fields or attributes.
- **For example -** Following Figure shows the relational model by showing the relationship between Student and Result database. For example -

Student Ram lives in city Chennai and his marks are 78. Thus, the relationship between these two databases is maintained by the SeatNo. Column

SeatNo	Name	City
101	Ram	Chennai
102	Shyam	Pune

Relational DBMS Example

SeatNo	Marks
101	78
102	95

Relational DBMS Example

Advantages

(i) Structural Independence

- Structural independence is an ability that allows us to make changes in one database structure without affecting other.
- The relational model has structural independence.
- Hence, making required changes in the database is convenient in relational database model.

(ii) Conceptual Simplicity

- The relational model allows the designer to simply focus on logical design and not on physical design.
- Hence, relational models are Conceptually simple to understand.

(iii) Query Capability

- Using simple query language (such as SQL) user can get information from the database or designer can manipulate the database structure.

(iv) Easy design, maintenance and usage

- The relational models can be designed logically hence they are easy to maintain and use.

Disadvantages

i. Relational model requires powerful hardware and large data storage devices.

ii. May led to slower processing time.

iii. Poorly designed systems lead to poor implementation of database systems.

(2) Entity relationship model:

- As the name suggests the entity relationship model uses collection of basic objects called entities and relationships.
- The entity is a thing or object in the real world.
- The entity relationship model is widely used in database design.
- **For example -** Following is a representation of Entity Relationship model in which the relationship works_for is between entities Employee and Department.

Advantages

1. **Simple:** It is simple to draw ER diagram when we know entities and relationships.
2. **Easy to understand:** The design of ER diagram is very logical and hence they are easy to design and understand.
3. **Effective:** It is effective communication tool.
4. **Integrated:** The ER model can be easily integrated with Relational model.
5. **Easy conversion:** ER model can be converted easily into other type of models.

Disadvantages

1. **Loss of information:** While drawing ER model some information can be hidden or lost.
2. **Limited relationships:** The ER model can represent limited relationships as compared to other models.
3. **No Representation for data manipulation:** It is not possible to represent data manipulation in ER model
4. **No industry standard:** There is no industry standard for notations of ER diagram.

(3) Object Based Data Model:

- The object-oriented languages like C++, Java, C# are becoming the domain in software development.
- This led to object-based data model.
- The object-based data model combines object-oriented features with relational data model.

Advantages:

1. **Enriched modelling:** The object-based data model has capability of modelling the real-world objects.
2. **Reusability:** There are certain features of object-oriented design such as inheritance, polymorphism which help in reusability.
3. **Support for schema evolution:** There is a tight coupling between data and applications, hence there is strong support for schema evolution.
4. **Improved performance:** Using object-based data model there can be significant improvement in performance using object-based data model.

Disadvantages:

1. **Lack of universal data model:** There is no universally agreed data model for an object-based data model, and most models lack a theoretical foundation.
2. **Lack of experience:** In comparison with relational database management the use of object-based data model is limited. This model is more dependent on the skilled programmer.
3. **Complex:** More functionalities present in object-based data model make the design complex.

(4) Semi-structured data model

- The semi-structured data model permits the specification of data where individual data items of same type may have different sets of attributes.
- The Extensible Markup Language (XML) is widely used to represent semi-structured data model.

Advantages

1. Data is not constrained by fixed schema.
2. It is flexible.
3. It is portable.

Disadvantage

1. Queries are less efficient than other types of data model.

7. Write about Data Independence

Definition

- Data independence is an ability by which one can change the data at one level without affecting the data at another level. Here level can be physical, conceptual or external as shown in figure 1.3.
- Data independence is one of the important characteristics of database management system. By this property, the structure of the database or the values stored in the database can be easily modified by without changing the application programs.

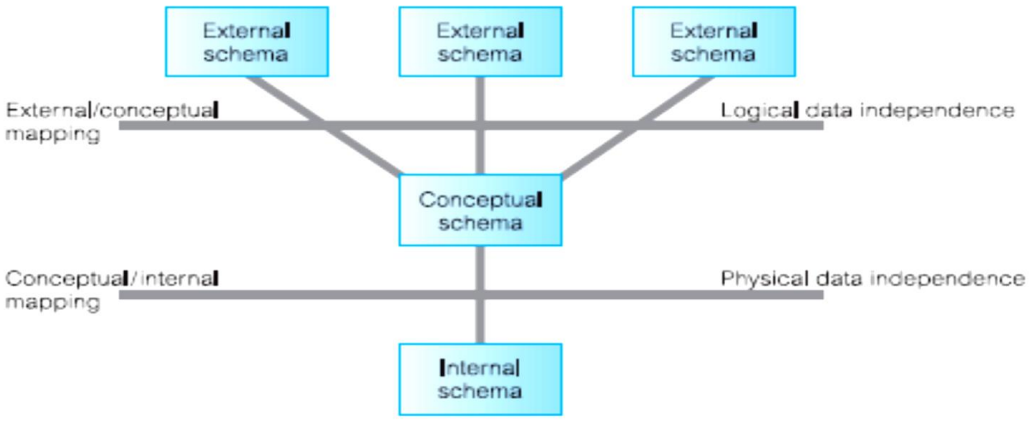


Figure1.3 Data Independence

- There are two types of data independence

1.Physical Independence

- This is a kind of data independence which allows the modification of physical schema without requiring any change to the conceptual schema.
- **For example -** if there is any change in memory size of database server then it will not affect the logical structure of any data object.

2. Logical Independence

- This is a kind of data independence which allows the modification of conceptual schema without requiring any change to the external schema.
- **For example -** Any change in the table structure such as addition or deletion of some column does not affect user views.
- By these data independence the time and cost acquired by changes in any one level can be reduced and abstract view of data can be provided to the user.

8.Sketch the typical Component modules of DBMS. Also, explain the interactions with architecture. OR Explain the architecture of DBMS. (NOV/DEC 2023) OR Classify database system architecture and explain. (APR/MAY 2023)

Typical Component modules of DBMS.

- The typical structure of typical DBMS is based on relational Data Model as shown in figure 1.4:

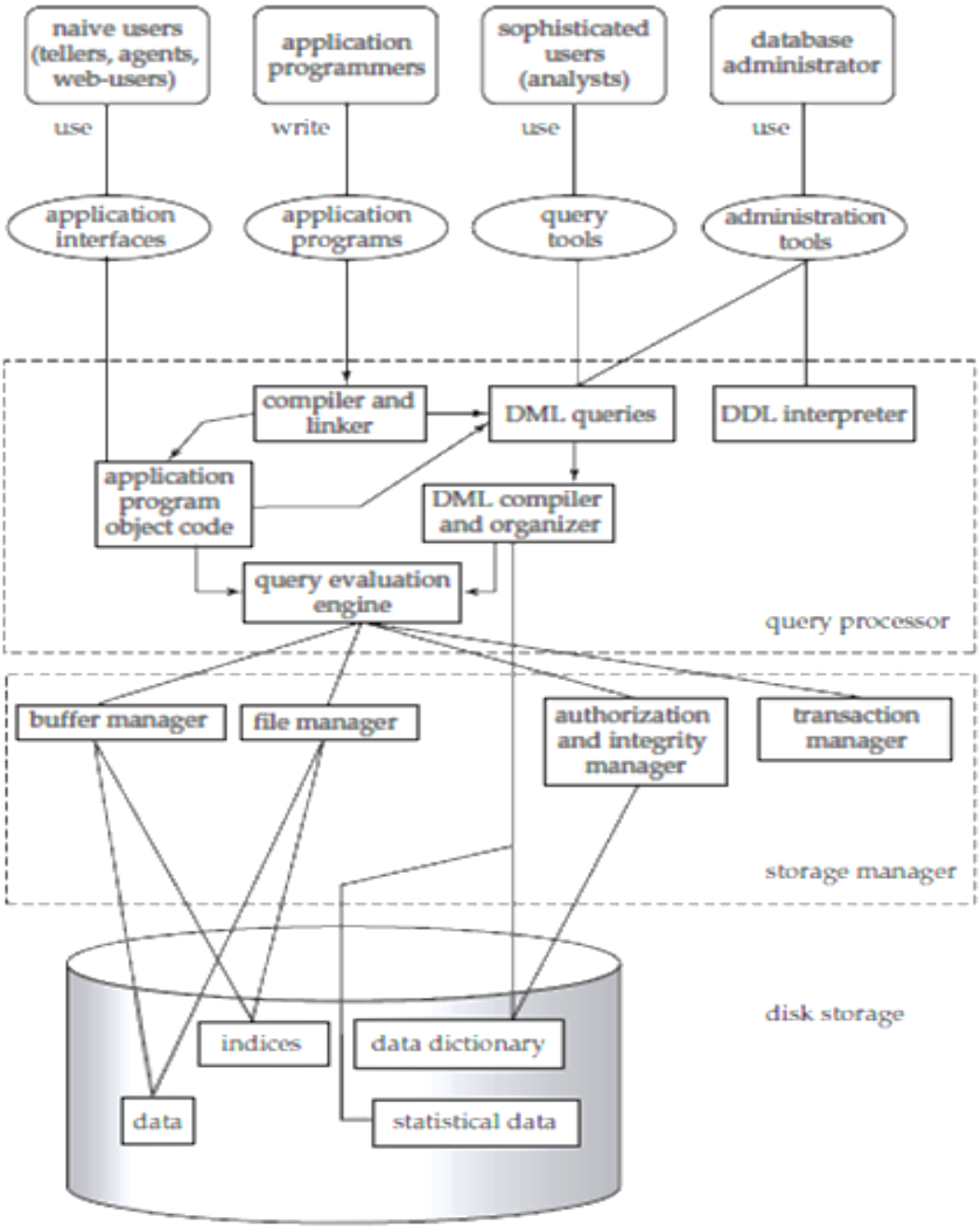


Figure1.4 Components of DBMS

- Consider the top part of Figure. It shows application interfaces used by naive users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator.
- The lowest part of the architecture is for disk storage.
- The two important components of database architecture are - Query processor and storage manager.

Query processor:

- The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL interpreter, DML compiler and query evaluation engine.
- With the following components of query processor, various functionalities are performed -
    1. **DDL interpreter:** This is basically a translator which interprets the DDL statements in data dictionaries.
    2. **DML compiler**: It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.
    3. **Query evaluation engine:** It executes the low-level instructions by the DML compiler generated.
    - When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query.
    - An execution plan is a blueprint for evaluating a query.
    - It is evaluated by query evaluation engine.

Storage Manager

- Storage manager is the component off database system that provides interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include –
    1. **Authorization and integrity manager:** Validates the users who want to access the data and tests for integrity constraints.

2. **Transaction manager:** Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
3. **File manager:** Manages allocation of space on disk storage and representation of the information on disk.
4. **Buffer manager:** Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.
    - Storage manager implements several data structures such as
        1. **Data Files:** Used for storing itself.
        2. **Data dictionary:** Used for storing metadata, particularly schema of database.
        3. **Indices:** Indices are used to provide fast access to data items present in the database.

9.Write about Database System Development Lifecycle in detail or Outline the flow of database system development lifecycle and explain. (APR/MAY 2023) (NOV/DEC 2022)

Database System Development Lifecycle:

- The stages in database system development life cycle are as shown in following figure 1.5.

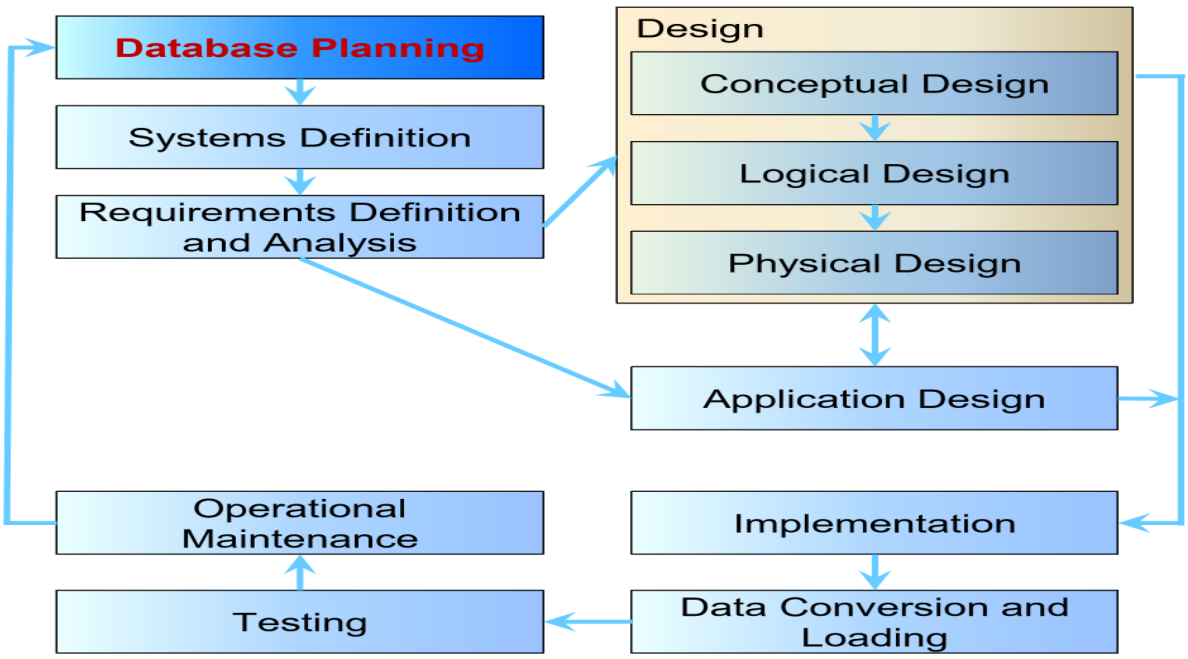


Figure1.5 Lifecycle of DBMS

1. Database planning

- This is the first stage in database system development life cycle.
- In this phase, the management activities that help in making development efficient and effective are planned.
- Planning should include development of standards.

2.System definition

- This stage describes the scope and boundaries of the database major user views.
- User view defines what is required of a database system from the perspective of various roles such as manager, supervisor, stock control and so on.

3.Requirement collection and analysis

- This stage involves collection and analysis of information about the part of the enterprise to be served by the database

4. Database Design

- It is the process of creating a design that help in achieving the objectives of h database system under development.
- Database design is carried out in three main phases - Conceptual, logical and physical.

5. DBMS Selection

- It is a phase in which the appropriate DBMS is selected to support database.

6. Application Design

- It is a phase of Database System development lifecycle in which the design of use interface and application programs that use the database systems is build.

7.Prototyping

- The working model of database system is built in this phase.
- The prototype may not provide all the required features or functionalities of final system.
- The main purpose of developing the prototype is to help the user to identify if basic required functionalities are working or not.

8.Implementation

- After completion of design stages, the database and application programs implemented.

- It is a physical realization of the database and application design.
- The database implementation is achieved using Data Definition Language (DDL) and Data Manipulation Language (DML).

- The application programs can be implemented using the programs like C, C+, Java, Fortran's and so on.

9. Data Conversion and Loading

- It is a process of transferring any existing data into new database and converting any existing application to run on the new database.
- This stage is required only when a new database system is replacing the old system.

10. Testing

- Testing is a technique of running a database system with the intent of finding errors.

11. Operational Maintenance

- Operational maintenance is a phase that comes in picture after installation of new database system. Maintenance is a process in which the database system is monitored and maintained.

10.Explain about requirements collection for database system in detail.

- This stage involves collection and analysis of information about the part of the enterprise to be served by the database.
- Following type of information is collected-
    1. A description of the data used or generated.
    2. The details of how data is to be used.
    3. Any additional requirements for the new database system.
- This information is then analyzed to identify the requirements of new database system.
- A document containing the specification of these requirements is then prepared. It is called requirements specification.
- The collected information can be converted to requirement statements using various techniques such as-
    1. Structured Analysis and Design (SAD)
    2. Data Flow Diagram (DFD)
    3. Hierarchical Input Process Output (HIPO)
    4. Documentations.
- When there are more multiple user views then requirements are managed using following approaches –
    1. Centralized approach

2. View integration approach
3. Combined approach

1.Centralized Approach

- There is a single set of requirements for each user-view as shown in figure 1.6
- The data model representing all user views is created during database design stage. Generally, this approach is preferred when there is a significant overlap in requirements for each user view.

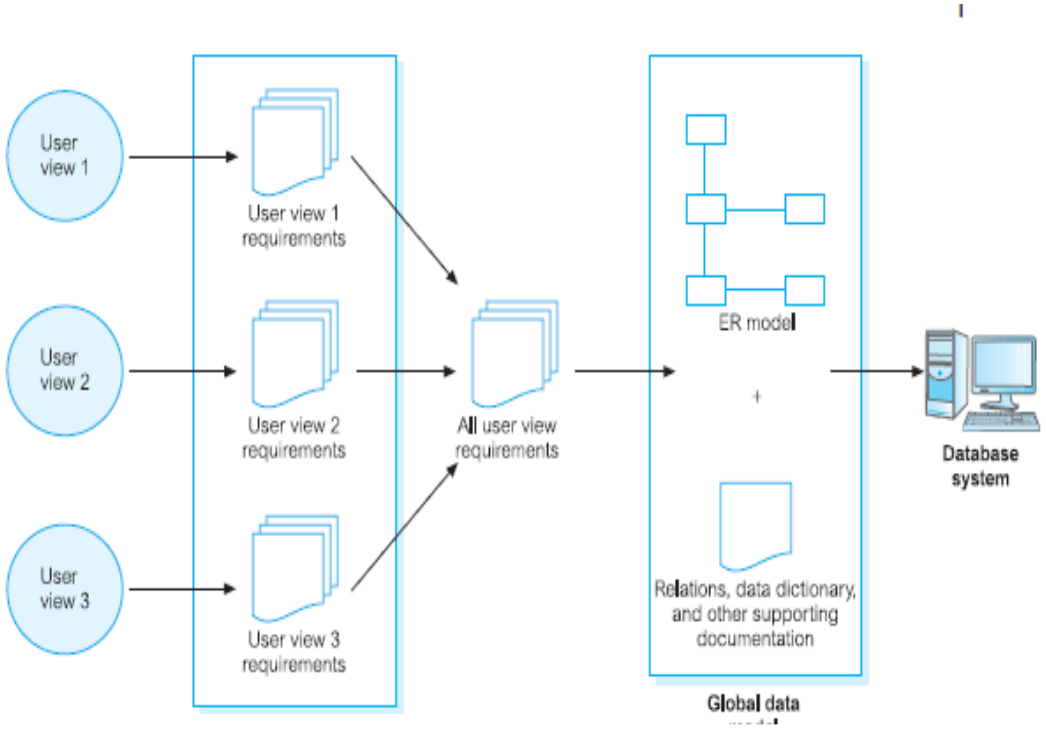


Figure 1.6 Centralized Approach

1. View integration Approach

- Requirements for each user view remain as separate lists as shown in figure 1.7.
- LData models representing each user view are created and then merged during database design stage.
- A data model that represents a single user view is called local data model.
- The local data models are then merged at a later stage of database design to produce global data model.

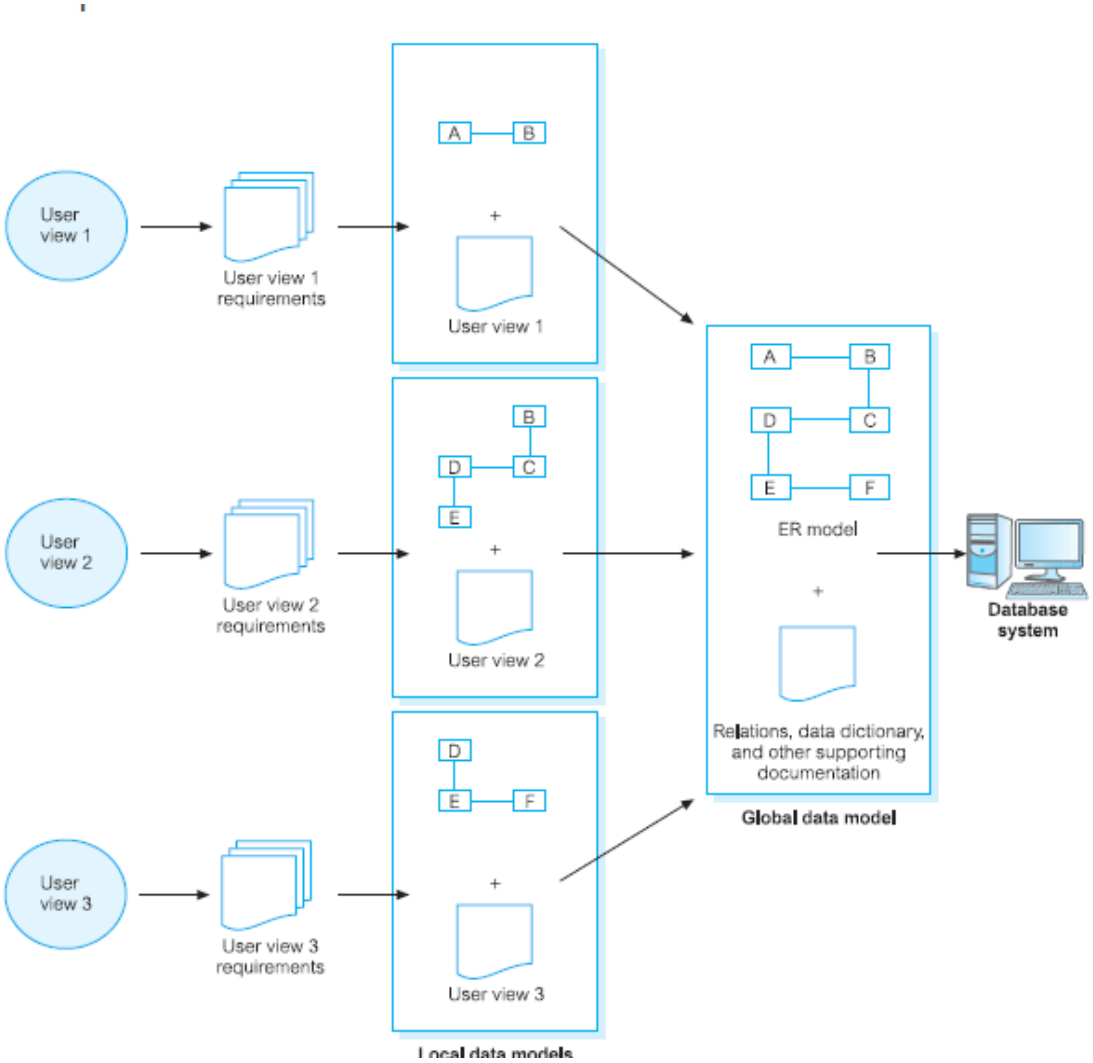


Figure 1.7 View integration Approach

3.Combined Approach

- In this approach there is appropriate combination of centralized and view approach.

11.How to design the Database?

Designing the Database

- It is the process of creating A design that help in achieving the objectives database system under development.
- Database design is carried out in three main phases as shown in figure 1.8:
  1. Conceptual database design
  2. Logical database design
  3. Physical database design

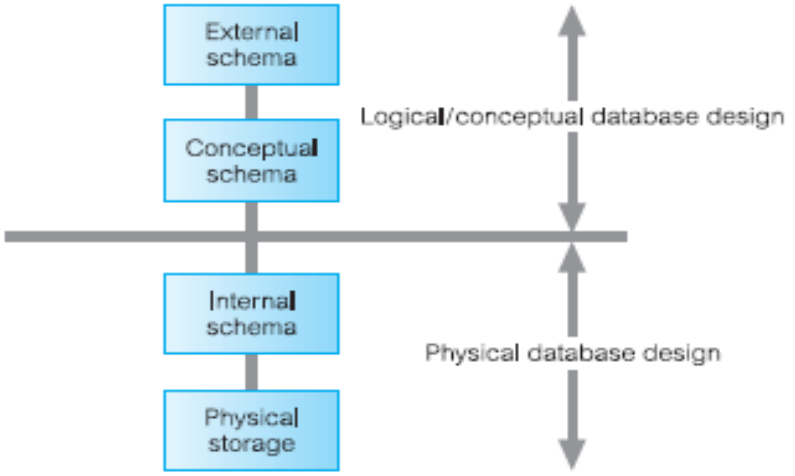


Figure 1.8 Database design

1.Conceptual Database design

- The conceptual Database design is the first phase in database design.
- In this phase, the conceptual data model is created, the data model is created using the information present in the requirements specification.
- Conceptual database design is entirely independent of implementation details, target DMB, Application program, programming languages, hardware platform and so on.
- The information in conceptual database design acts as an input for the next phase i.e. logical database design.

2.Logical Database Design

- The second phase of database design is logical database design.
- In this phase a logical data model is created. It is based on the relational data model of Database system.
- Throughout the process of development of logical database design, the logical model is tested and validated against the user requirements. The technique of normalization is used for correctness of the logical model.
- The information in the logical database design acts as an input for the next phase i.e. physical database design.

3.Physical Database Design

- This is the final stage of database design.
- The main aim of physical database design is physical implementation of logical database design.
- The conceptual and logical database design for larger system is separated from the physical database design.
- The database design is **an iterative** process and in each iteration the data model are refined.

12. Explain the concept about Entity-Relationship Model in Detail. OR Explain the following terms briefly in ER Model: attribute, domain, entity, relationship, entity set, relationship set, one-many, many-many, participating constraint, overlapping constraint, weak entity set, aggregation, role indicator. (NOV/DEC 2022)

Introduction to Entity Relationship Model

- Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.
- Let us first understand the design process of database design as shown in figure 1.9

Design Phases

- Following are the six steps of database design process. The ER model is most relevant to first three steps

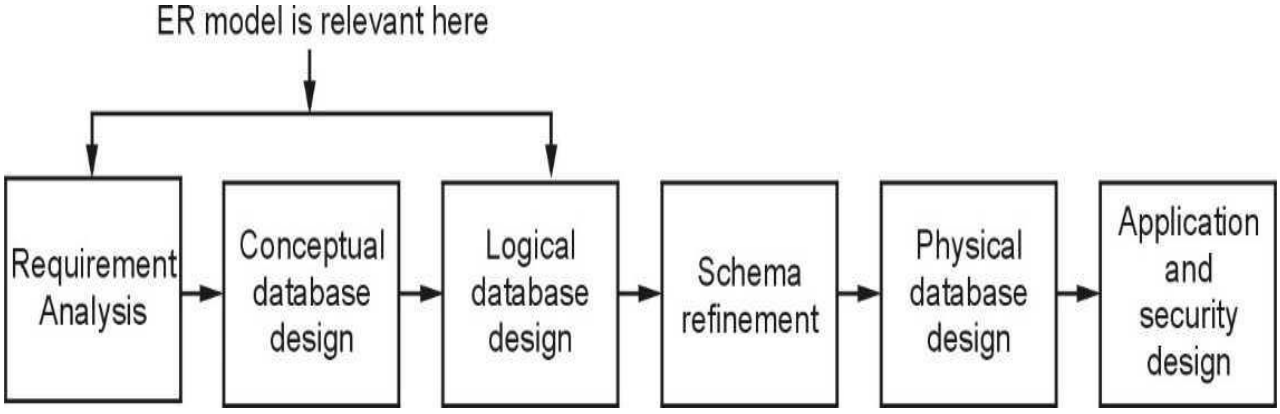


Figure. 1.9 Database design process

Step 1: Requirement analysis:

- In this step, it is necessary to understand what data need to be stored in the database, what applications must be built, what are all those operations that are frequently used by the system.

- The requirement analysis is an **informal** process and it requires proper **communication** with user groups.
- There are several methods for **organizing and presenting information** gathered in this step.
- Some **automated tools** can also be used for this purpose.

Step 2: Conceptual database design:

- This is a step in which **E-R Model** i.e. Entity Relationship model is built.
- E-R model is a **high-level data model** used in database design.
- The **goal** of this design is to create a simple description of data that matches with the requirements of users.

Step 3: Logical database design:

- This is a step in which ER model in **converted to relational database schema**, sometimes called as the logical schema in the relational data model.

Step 4: Schema refinement:

- In this step, **relational database schema is analysed** to identify the potential problems and to refine it.
- The schema refinement can be done with the help of **normalizing and restructuring the relations.**

Step 5: Physical database design:

- In this step, the design of database is **refined further.**
- The tasks that are performed in this step are - building **indexes** on tables and **clustering** tables, redesigning some parts of schema obtained from earlier design steps.

Step 6: Application and security design:

- Using design methodologies like UML (Unified Modeling Language) the design of the database can be accomplished.
- The **role of each entity** in every process must be reflected in the application task.
- For each role, there must be the provision for **accessing** some part of database and **prohibition of access** to some other part of database.
- Thus, some **access rules** must be enforced on the application (which is accessing the database) to protect the **security features.**

ER Model

- The ER data model specifies enterprise schema that represents the overall logical structure of a database.

- The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.
- The ER model consists of three basic concepts –

1) Entity Sets

- **Entity:** An entity is an object that exists and is distinguishable from other objects.
- **For example -** Student named "Poonam" is an entity and can be identified by her name. The entity can be concrete or abstract. The concrete entity can be - Person, Book, Bank. The abstract entity can be like - holiday, concept entity is represented as a box.

STUDENT

- **Entity set:** The entity set is a set of entities of the same types. **For example –** All students studying in class X of the School. The entity set need not be disjoint. Each entity in entity set have the same set of attributes and the set of attributes will distinguish it from other entity sets. No other entity set will have exactly the same set of attributes.

2) Relationship Sets

- Relationship is an association among two or more entities.
- The **relationship set** is a collection of similar relationships. **For example –** Following Figure 1.10 shows the relationship **works for** the two entities **Employee** and **Departments.**

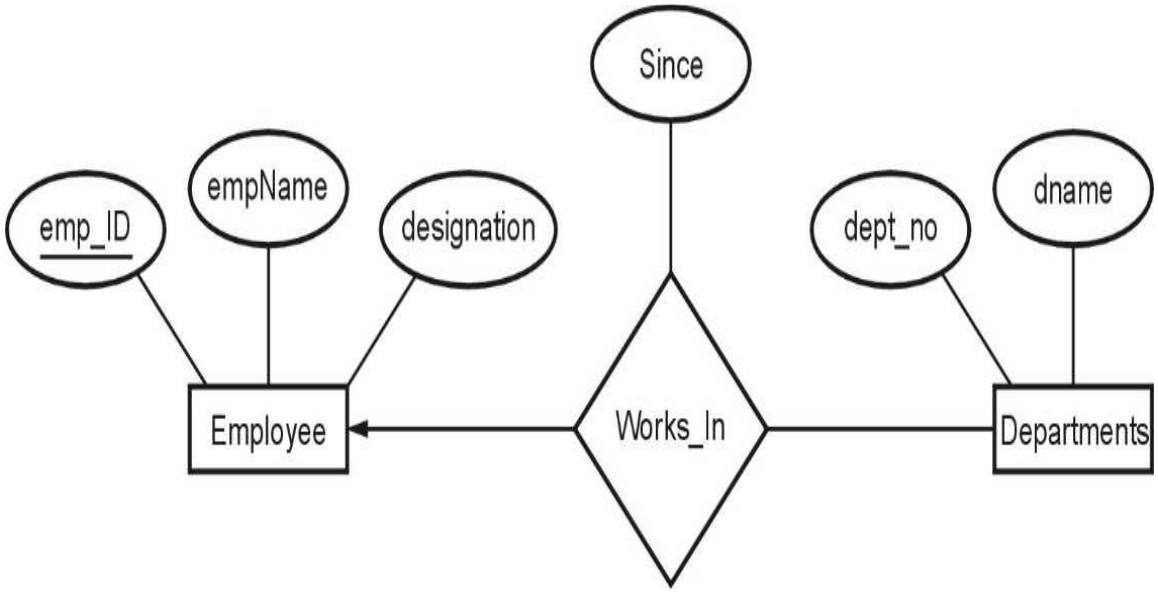


Figure 1.10 Relationship between Employee and Departments.

- The association between entity sets is called as **participation**. that is, the entity sets E1, E2, . . ., En participate in relationship set R.
- The function that an entity plays in a relationship is called that entity's **role**.

3) Attributes

- Attributes define the properties of a data object of entity.
- **For example,** if student is an entity, his ID, name, address, date of birth, class are its attributes as shown in figure 1.11.
- The attributes help in determining the unique entity.
- Student entity set with attributes- ID, name, address. Note that entity is shown by rectangular box and attributes are shown in oval. The primary key is underlined.

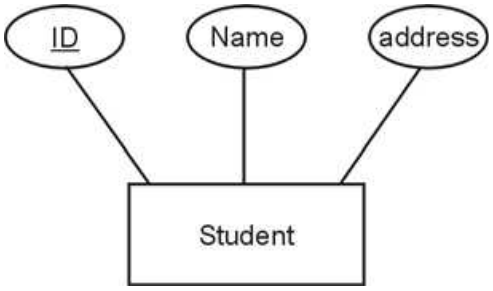


Figure 1.11 Attributes for Student

Types of Attributes

1) Simple and Composite Attributes:

- **Simple attributes** are attributes that are drawn from the atomic value domains.
- **For example -** Name = {Parth}; Age = {23}
- **Composite attributes:** Attributes that consist of a hierarchy of attributes
- For example - Address may consists of "Number", "Street" and "Suburb"

→ Address = {59 + 'JM Road' + 'ShivajiNagar'}

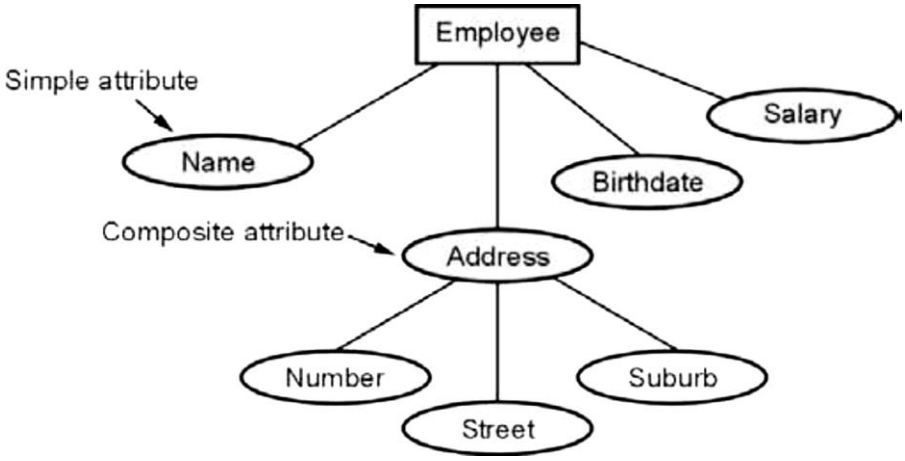


Figure 1.12 Simple and Composite attributes.

2) Single valued and multivalued:

- There are some attributes that can be represented using a single value.
- **For example -** StudentID attribute for a student is specific only one studentID.
- **Multivalued attributes:** Attributes that have a set of values for each entity. It is represented by concentric ovals as shown in figure 1.13.
- **For example -** Degrees of a person: 'BSc', 'MTech', 'PhD'

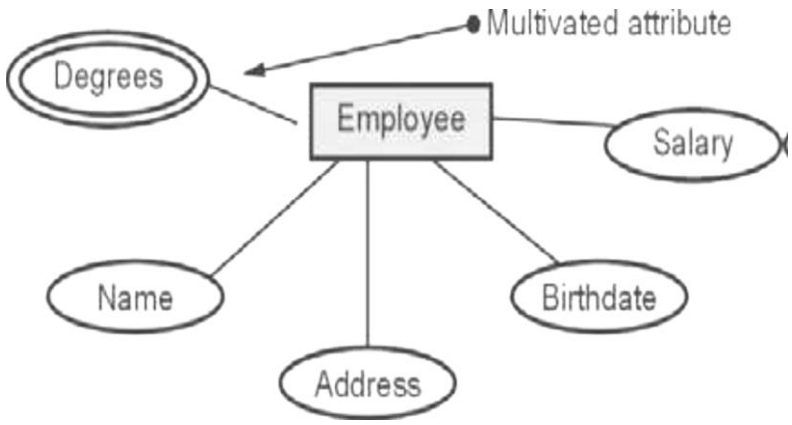


Figure 1.13 Single and Multivalued Attributes

3) Derived attribute:

- Derived attributes are the attributes that contain values that are calculated from other attributes.
- To represent derived attribute there is dotted ellipse inside the solid ellipse. **For example –**Age can be derived from attribute DateOfBirth as shown in figure 1.14. In this situation, DateOfBirth might be called Stored Attribute.

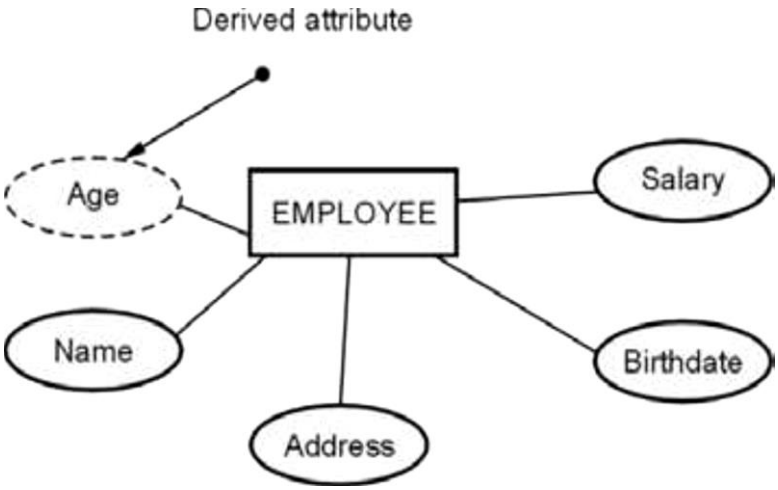


Figure 1.14 Derived Attributes

13. Define Mapping Cardinality. Also, explain the various types of Mapping Cardinality.

Mapping Cardinality

- Mapping Cardinality represents the number of entities to which another entity can be associated via a relationship set.
- The mapping cardinalities are used in representing the binary relationship sets.
- Various types of mapping cardinalities are -

1) One to One: An entity A is associated with at least one entity on B and an entity B is associated with at one entity on A. This can be represented as as shown in figure 1.15.

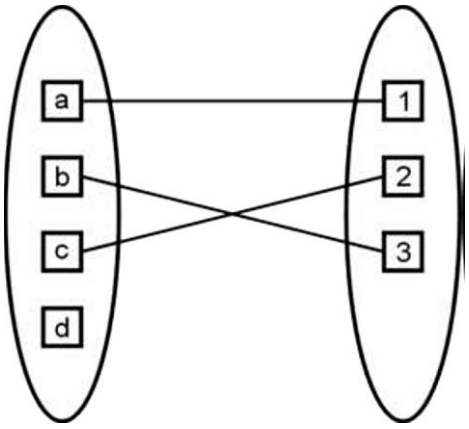


Figure 1.15 One to One Mapping

2) One to Many: An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A as shown in figure 1.16.

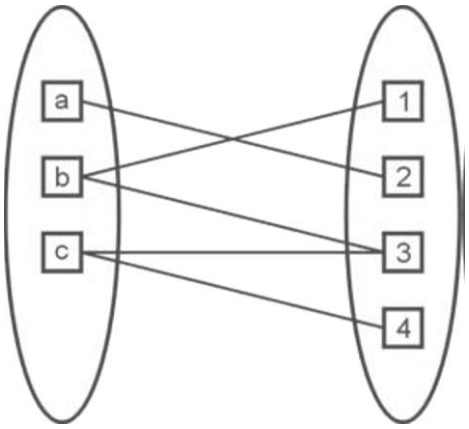


Figure 1.16 One to Many Mapping

3) 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 of entities in A as shown in figure 1.17.

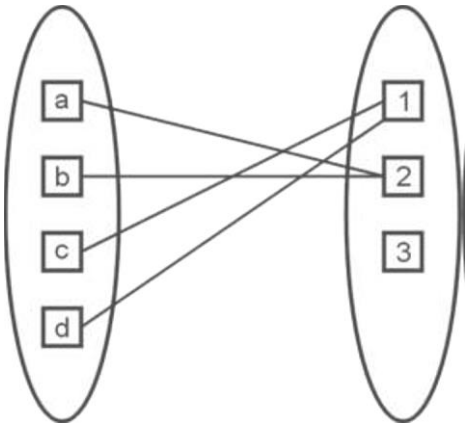


Figure 1.17 Many to One Mapping

4) 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 as shown in figure 1.18

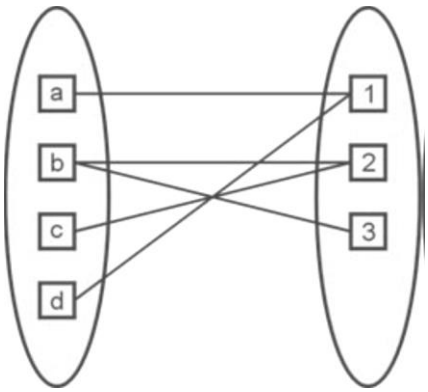


Figure 1.18 Many to Many Mapping

14. Explain ER diagram and its components in detail.

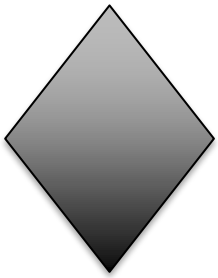
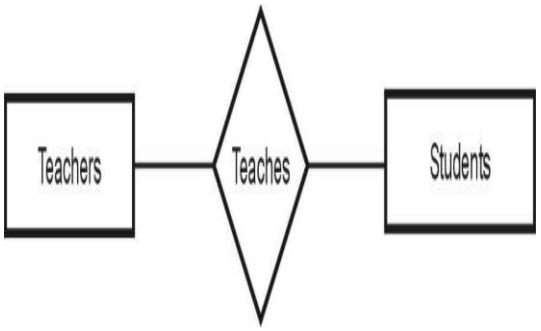
ER Diagrams

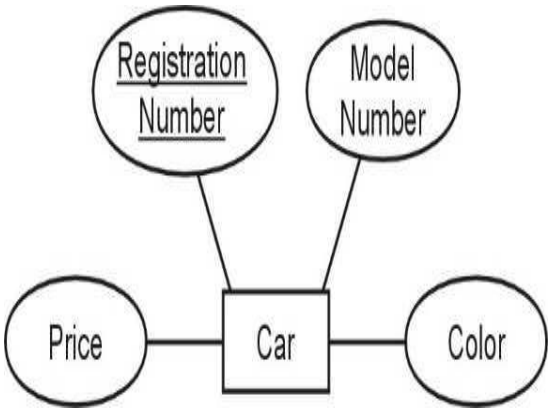
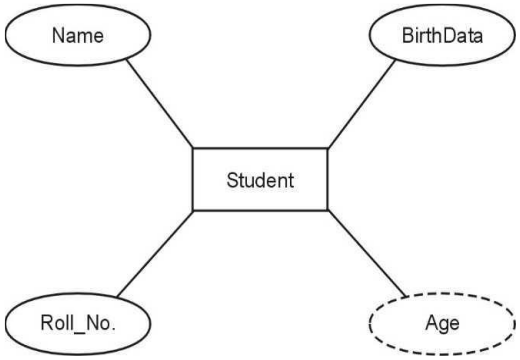
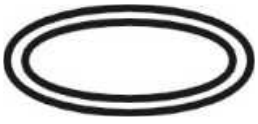
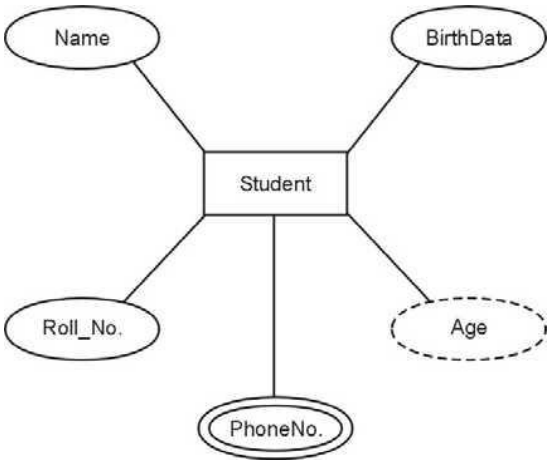
- An E-R diagram can express the overall logical structure of a database graphically. E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

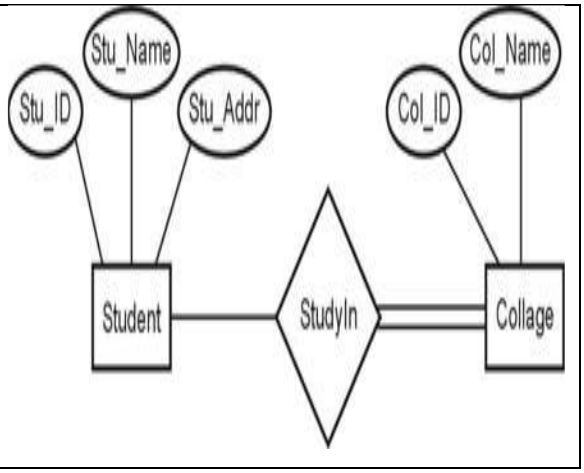
Features of ER model

1. E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
2. E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
3. E-R diagrams require no technical knowledge and no hardware support.
4. These diagrams are very easy to understand and easy to create even by a naive user.
5. It gives a standard solution of visualizing the data logically.

Various Components used in ER Model are –

Component	Symbol	Example
Entity: Any real-world object can be represented as an entity about which data can be stored in a database. All the real world objects like a book, an organization, a product, a car, a person are the examples of an entity.		
Relationship: Rhombus is used to setup relationships between two or more entities.		
Attribute: Each entity has		

a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc	 	
Derived attribute: Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth. To represent a derived attribute, another dotted ellipse is created inside the main ellipse		
Multivalued attribute: An attribute that can hold multiple values is known as multivalued attribute. We represent it with double ellipses in an E-R Diagram. E.g. A person can have more than one phone numbers so the phone number attribute is multivalued.		

Total participation: Each entity is involved in the relationship. Total participation is represented by double lines.		

Mapping Cardinality Representation using ER Diagram

- There are four types of relationships that are considered for key constraints.

1. **One to one relation:** When entity A is associated with at the most one entity B then it shares one to one relation. For example - There is one project manager who manages only one project as shown in figure 1.19.

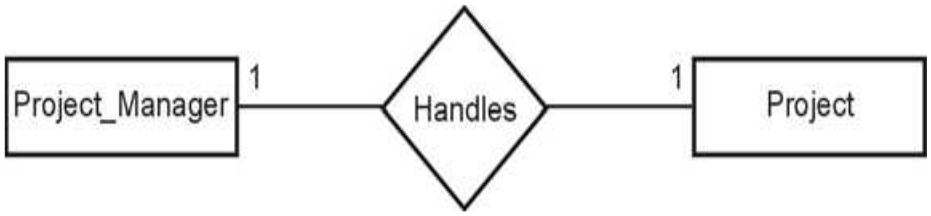


Figure 1.19 One to One Relation

2. **One to many:** When entity A is associated with more than one entity at a time then there is one to many relations. For example - One customer places order at a time as shown in figure 1.20.

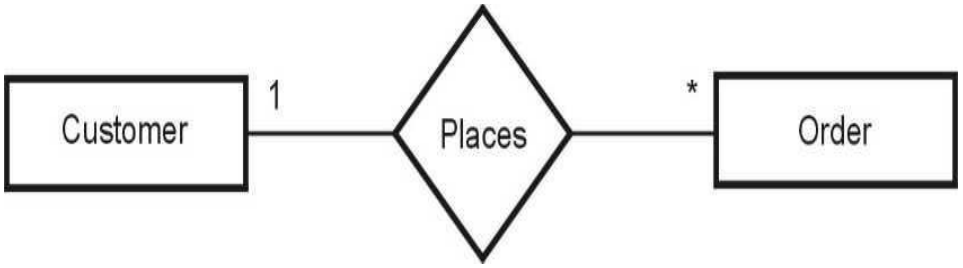


Figure 1.20 One to Many Relation

3. **Many to one:** When more than one entity are associated with only one entity then there is many to one relation. For example - Many students take a ComputerSciCourse as shown in figure 1.21.

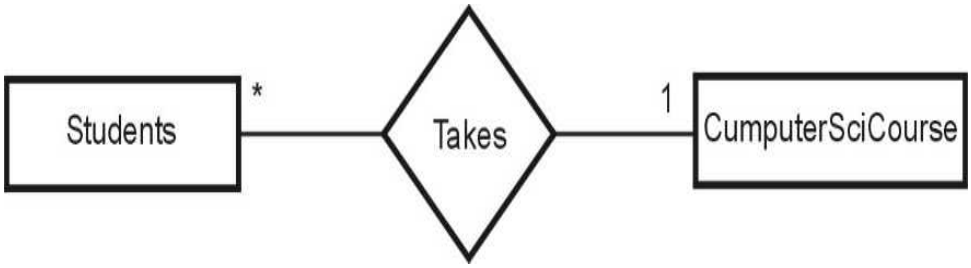


Figure 1.21 Many to One Relation

Alternate representation can be

Figure 1.22 Many to One Relation

4. **Many to many:** When more than one entity is associated with more than one entity. **For example -**Many teachers can teach many students as shown in figure 1.23.

Figure 1.23 Many to Many Relation

Alternate representation can be

Figure 1.24 Many to Many Relation

Ternary Relationship

- The relationship in which three entities are involved is called ternary relationship as shown in 1.25.

- **For example –**

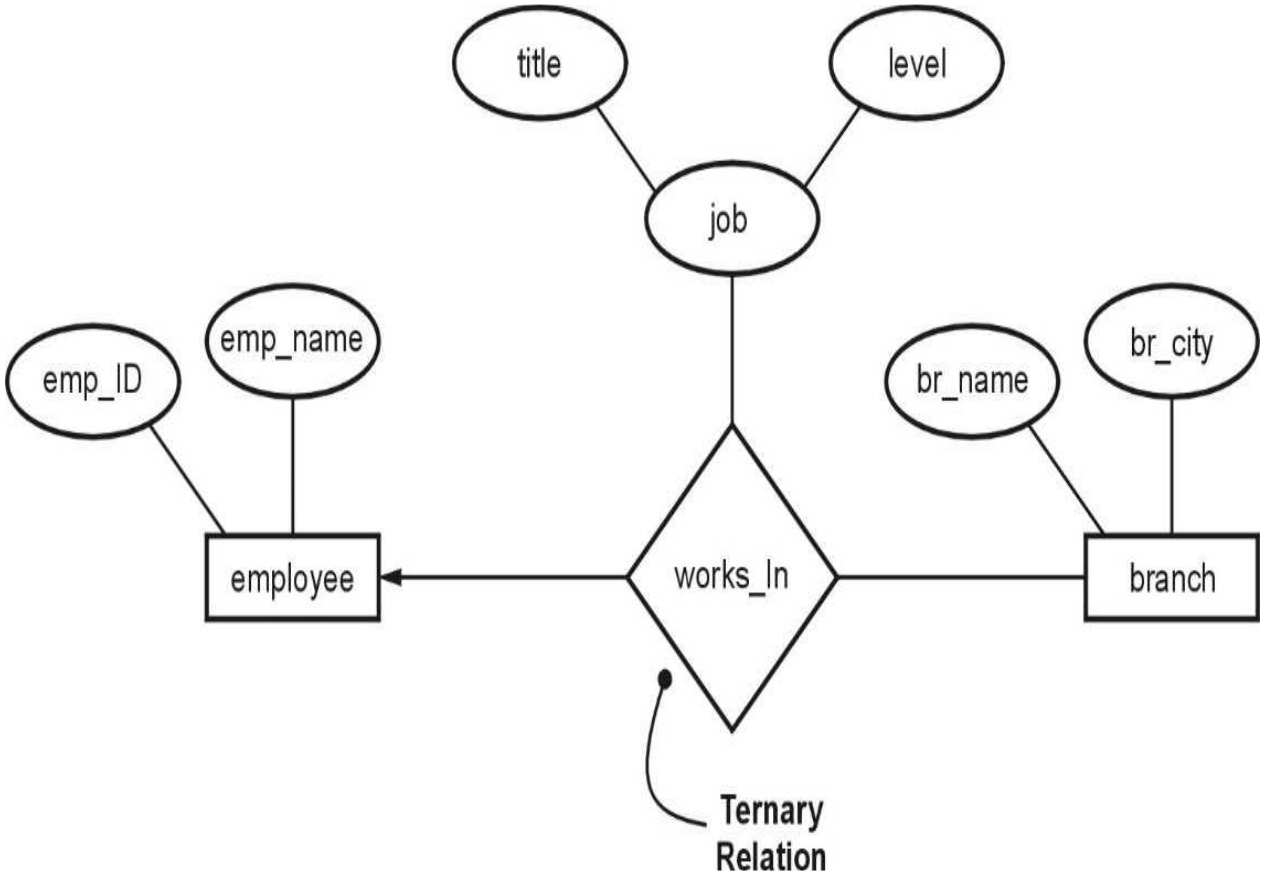


Figure 1.25 Ternary Relation

Binary and Ternary Relationships

- Although binary relationships seem natural to most of us, in reality it is sometimes necessary to connect three or more entities.
- If a relationship connects three entities, it is called ternary or "3-ary."
- Ternary relationships are required when binary relationships are not sufficient to accurately describe the semantics of an association among three entities.
- **For example -** Suppose, you have a database for a company that contains the entities, PRODUCT, SUPPLIER, and CUSTOMER as shown in figure 1.26. The usual relationships might be PRODUCT/ SUPPLIER where the company buys products from a supplier – a normal binary relationship. The intersection attribute for PRODUCT/SUPPLIER is wholesale_price.

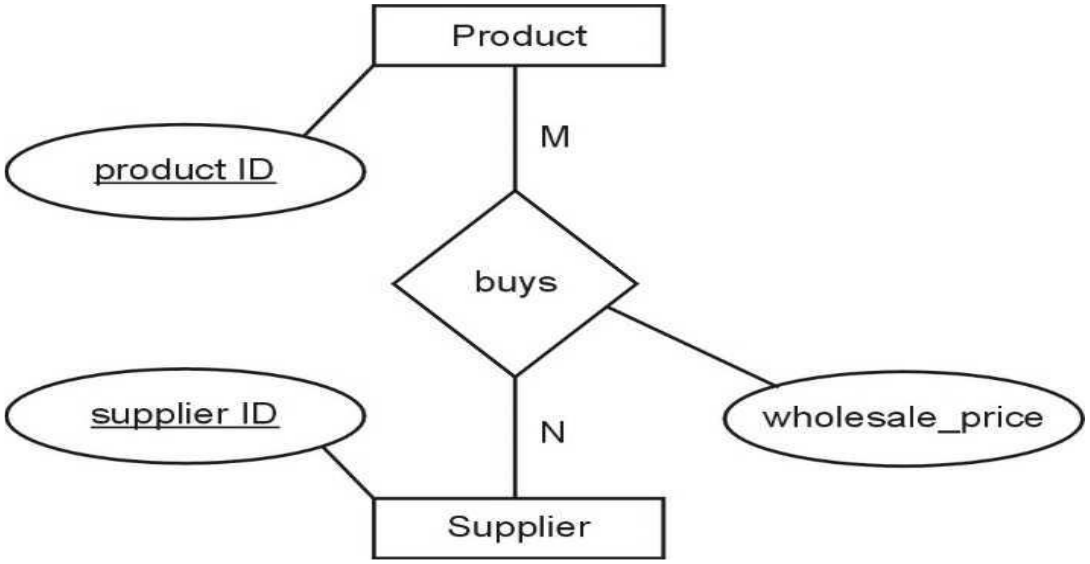


Figure 1.26 Normal Binary Relationship

- Now consider the CUSTOMER entity, and that the customer buys products.
- If all customers pay the same price for a product, regardless of supplier, then you have a simple binary relationship between CUSTOMER and PRODUCT.
- For the CUSTOMER/ PRODUCT relationship, the intersection attribute is retail_price as shown in figure 1.27.

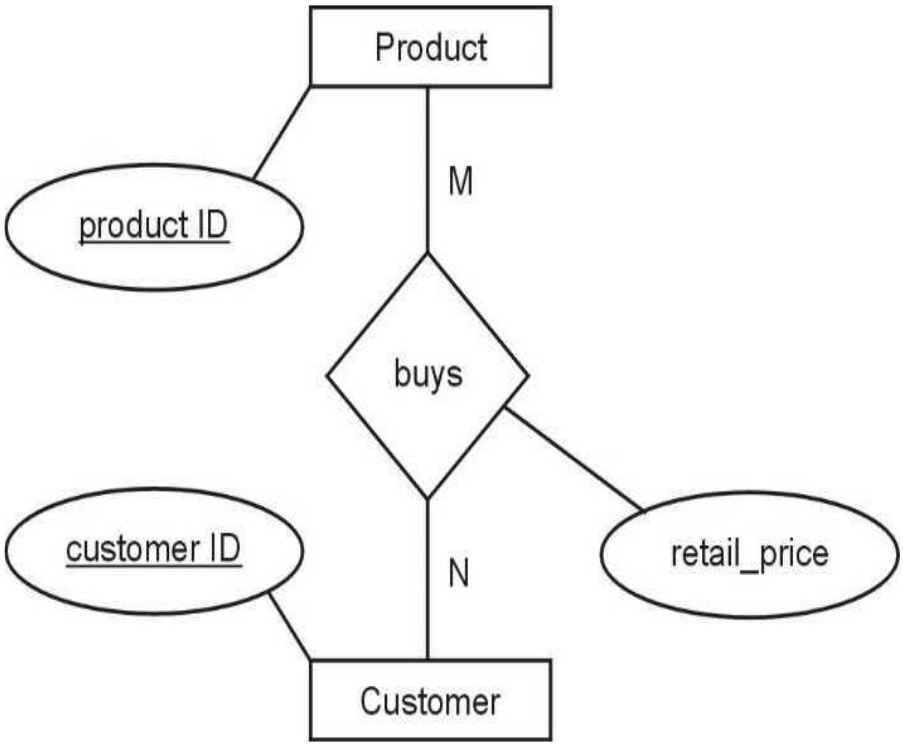


Figure 1.27 Simple Binary Relation

Single ternary relation:

- Now consider a different scenario.
- Suppose the customer buys products but the price depends not only on the product, but also on the supplier.
- Suppose you needed a customerID, a productID, and a supplierID to identify a price. Now you have an attribute that depends on three things and hence you have a relationship between three entities (a ternary relationship) that will have the intersection attribute, price as shown in figure 1.28.

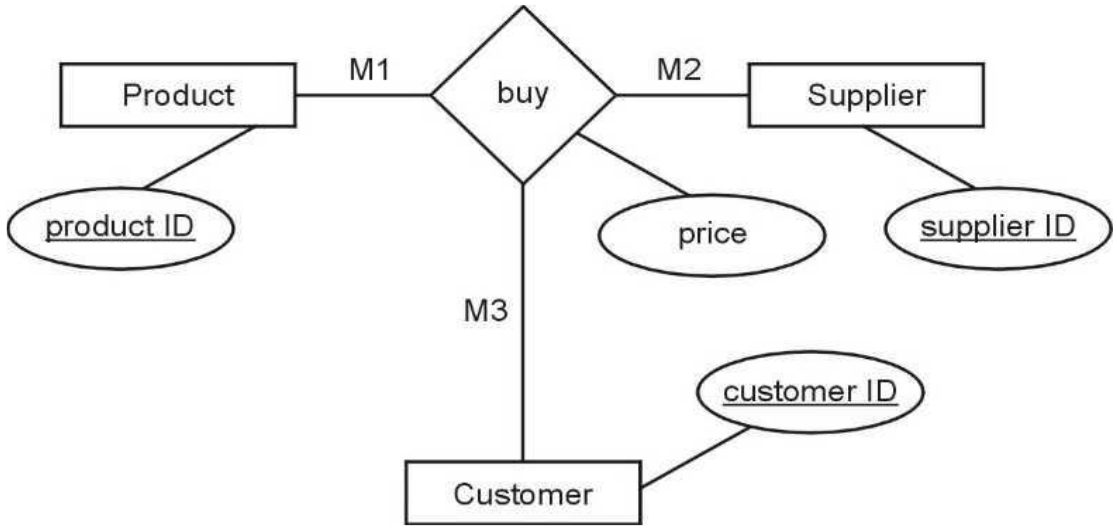


Figure 1.28 Single ternary relation

Weak Entity Set

- A weak entity is an entity that cannot be uniquely identified by its attributes Alone as shown in figure 1.29.
- The entity set which does not have sufficient attributes to form a primary key is called as weak entity set.

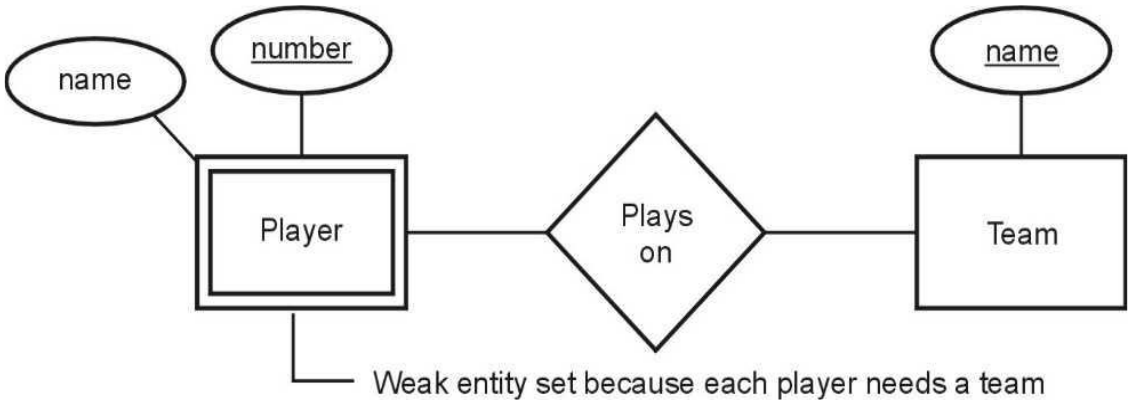


Figure 1.29 Weak Entity Set

Strong Entity Set

- The entity set that has primary key is called as strong entity set

Weak entity rules

- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
- The key for a weak entity is its own underlined attributes and the keys for the supporting entity sets. For example - player-number and team-name is a key for Players.

Difference between Strong and Weak Entity Set

Strong entity set	**Weak entity set**
It has its own primary key.	It does not have sufficient attribute to form a primary key on its own.
It is represented by rectangle	It is represented by double rectangle.
It represents the primary key which is underlined.	It represents the partial key or discriminator which is represented by dashed underline.
The member of strong entity set is called as dominant entity set	The member of weak entity set is called subordinate entity set.
The relationship between two strong entity sets is represented by diamond symbol.	The relationship between strong entity set and weak entity set is represented by double diamond symbol.
The primary key is one of the attributes which uniquely identifies its member.	The primary key of weak entity set is a combination of partial key and primary key of the strong entity set.

15. Explain with suitable example, the constraints of Specialization and Generalization of Enhanced-ER Model or Compare and contrast between Specialization and Generalization process in Enhanced-ER Model. (APR/MAY 2023)

Enhanced ER Model

Specialization and Generalization

- Some entities have relationships that form hierarchies. For instance, Employee can be an hourly employee or contracted employee.
- In this relationship hierarchies, some entities can act as superclass and some other entities can act as subclass.

Superclass: An entity type that represents a general concept at a high level, is called superclass.

Subclass: An entity type that represents a specific concept at lower levels, is called subclass.

- The subclass is said to inherit from superclass. When a subclass inherits from one or more super classes, it inherits all their attributes. In addition to the inherited attributes, a subclass can also define its own specific attributes.
- The process of making subclasses from a general concept is called **specialization.**
- This is top-down process.
- In this process, the sub-groups are identified within an entity set which have attributes that are not shared by all entities.
- The process of making superclass from subclasses is called **generalization.**
- This is a **bottom-up** process. In this process multiple sets are synthesized into high level entities.
- The symbol used for specialization/ Generalization is

- **For example –** There can be two subclass entities namely **Hourly_Emps** and **Contract_Emps** which are subclasses of **Employee** class. We might have attributes **hours_worked** and **hourly_wage** defined for **Hourly_Emps** and an attribute **contractid** defined for **Contract_Emps**.
- Therefore, the attributes defined for an **Hourly_Emps** entity are the attributes for **Employees** plus **Hourly_Emps**.
- We say that the attributes for the entity set **Employees** are inherited by the entity set **Hourly_Emps** and that **Hourly-Emps ISA** (read is a) Employees. It can be represented by following Figure 1.30.

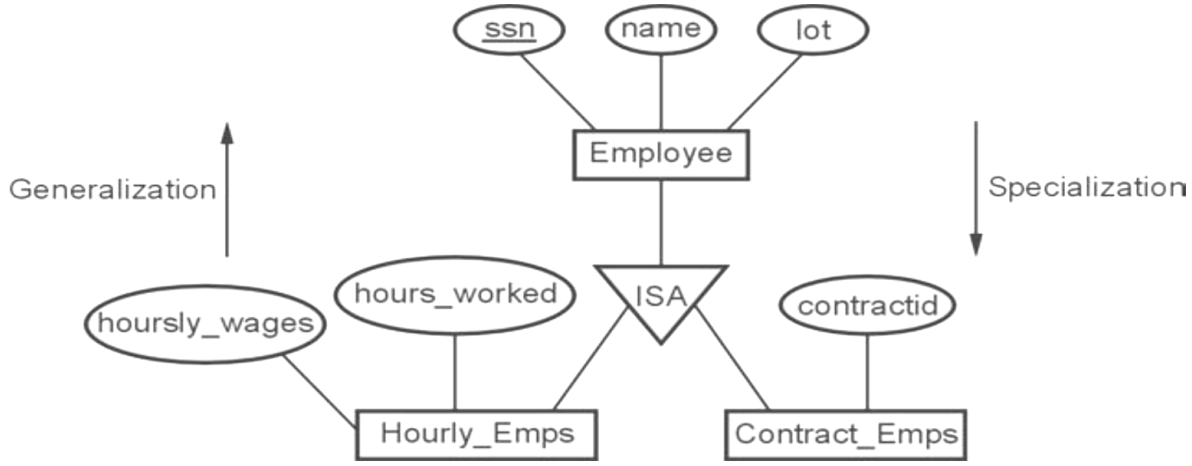


Figure 1.30 Specialization/Generalization

Constraints on Specialization/Generalization

- There are four types of constraints on specialization/generalization relationship. These are -

  1) Membership constraints: This is a kind of constraints that involves determining which entities can be members of a given lower-level entity. There are two types of membership constraints -

- **Condition defined:** In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate.
- **For example -** Consider the high-level entity Set
- **Employee** that has attribute **Employee_type**. All Employee entities are evaluated on defining **Employee_type** attribute. All entities that satisfy the condition student type = "ContractEmployee" are included in Contracted Employee. Since all the lower-level entities are evaluated on the basis of the same attribute this type of generalization is said to be **attribute-defined**.
- **User defined:** This is kind of entity set that in which the membership is manually defined.

  2) Disjoint constraints: The disjoint constraint only applies when a superclass has more than one subclass. If the subclasses are disjoint, then an entity occurrence can be a member of only one of the subclasses. For entity Student has either **Postgraduate_Student** entity or **Undergraduate_Student** as shown in figure 1.31.

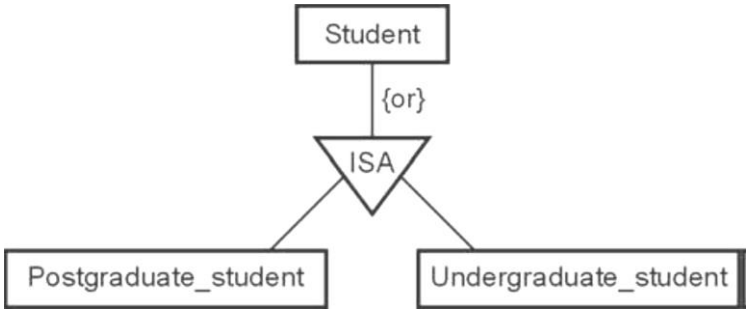


Figure 1.31 Disjoint constraints

3) Overlapping: When some entity can be a member of more than one subclasses. **For example -** Person can be both a Student or a Staff. The **And** can be used to represent this constraint as shown in figure 1.32.

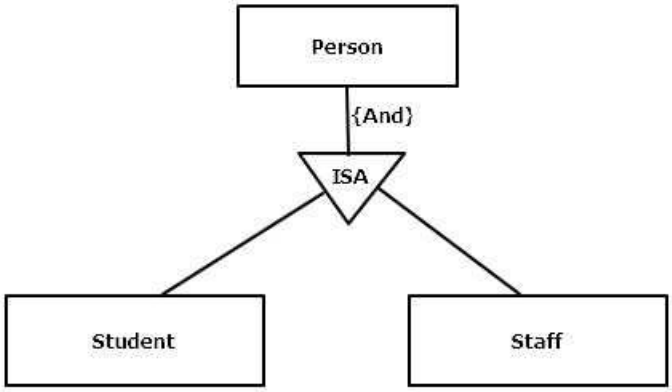


Figure 1.32 Overlapping

4) Completeness: It specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following -

- **Total generalization or specialization:** Each higher-level entity must belong to a lower-level entity set. **For example -** Account in the bank must either Savings account or Current Account. The **mandatory** can be used to represent this constraint as shown in figure 1.33.

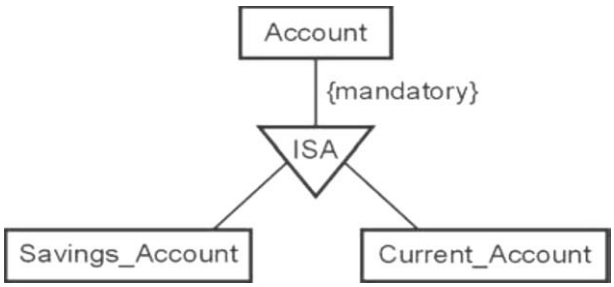


Figure 1.33 Total generalization or specialization

- **Partial generalization or specialization:** Some higher-level entities may not belong to any lower-level entity set as shown in figure 1.34.

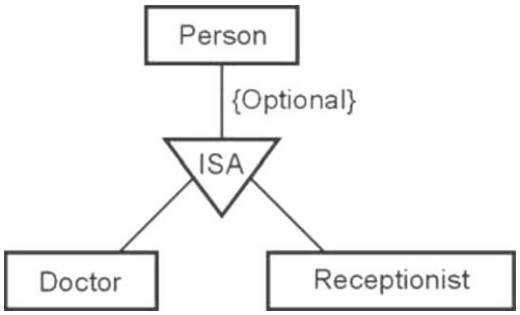


Figure 1.34 Partial generalization or specialization

Aggregation

- A feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation as shown in figure 1.35.
- **For example -** We treat the relationship set work and the entity sets employee and project as a higher-level entity set called work.

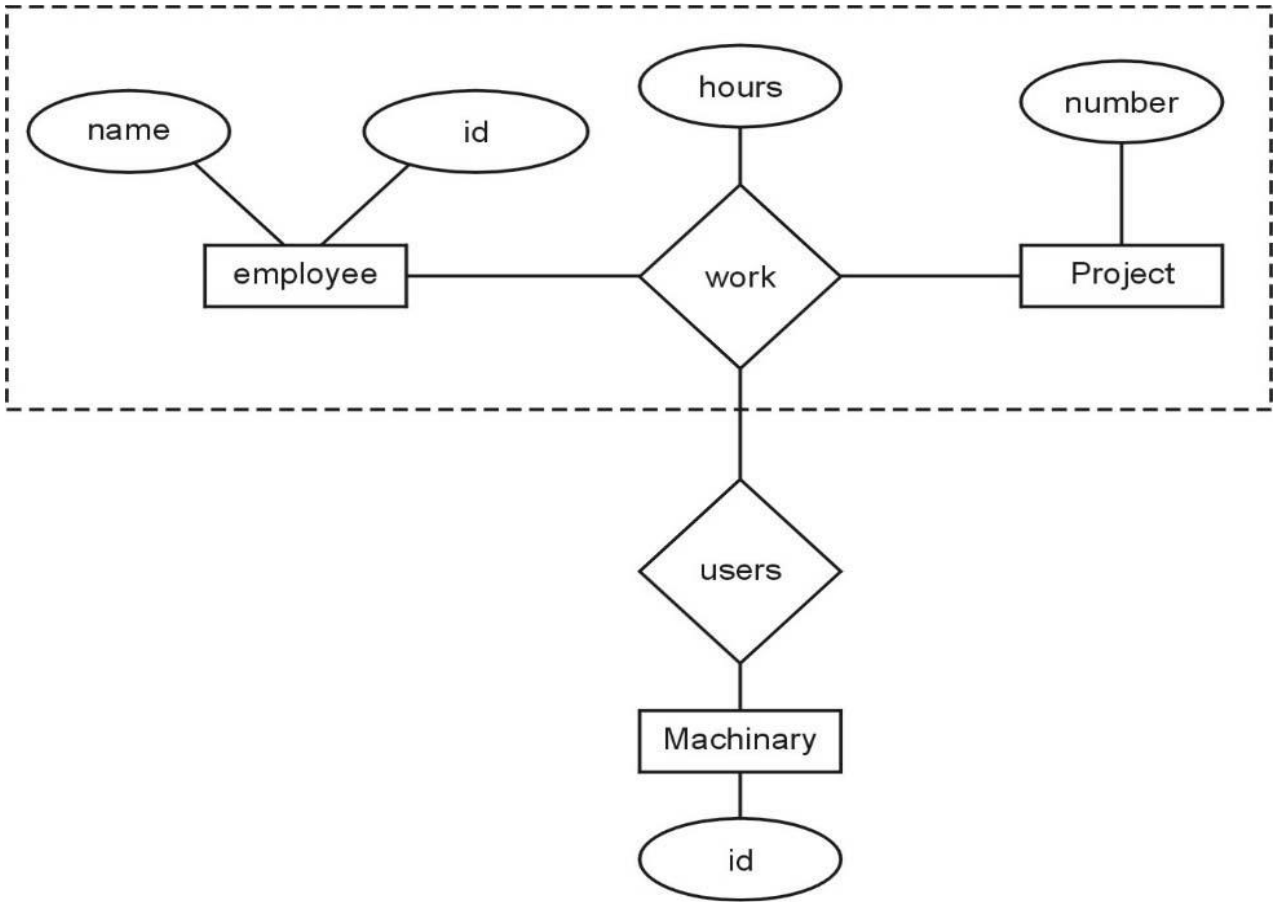


Figure 1.35 Aggregation

16. Draw the ER diagram for banking systems (home loan applications). (OR) Draw an ER diagram corresponding to customers and loans. (OR) Write short notes on: E-R diagram for banking system.

SOLUTION

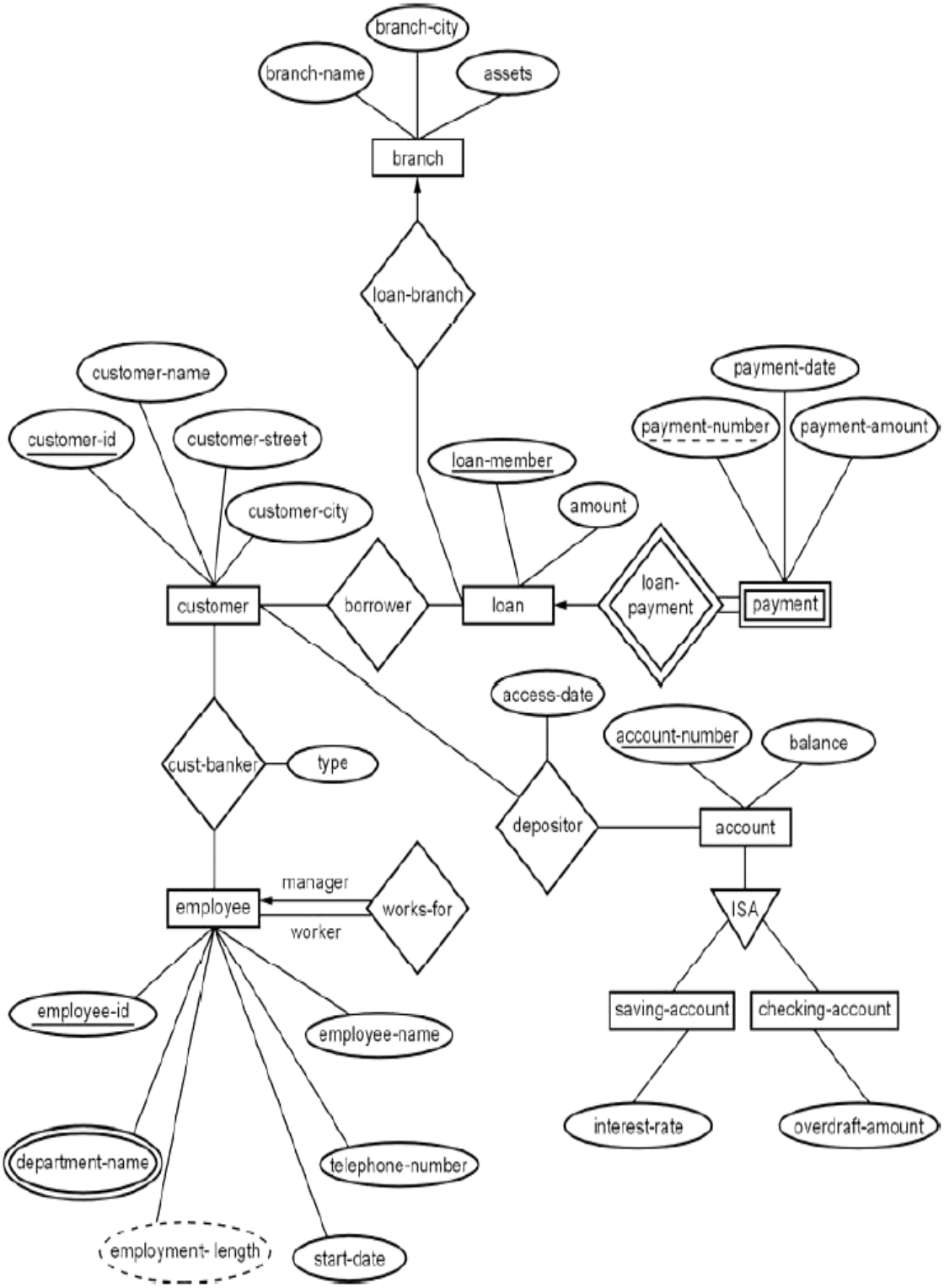


Figure 1.36 ER diagram for banking systems

17. Consider the relation schema given in Figure. Design and draw an ER diagram that captures the information of this schema.

Employee(empno,name,office,age)

Books(isbn,title,authors,publisher)

Loan(empno,isbn,date)

SOLUTION

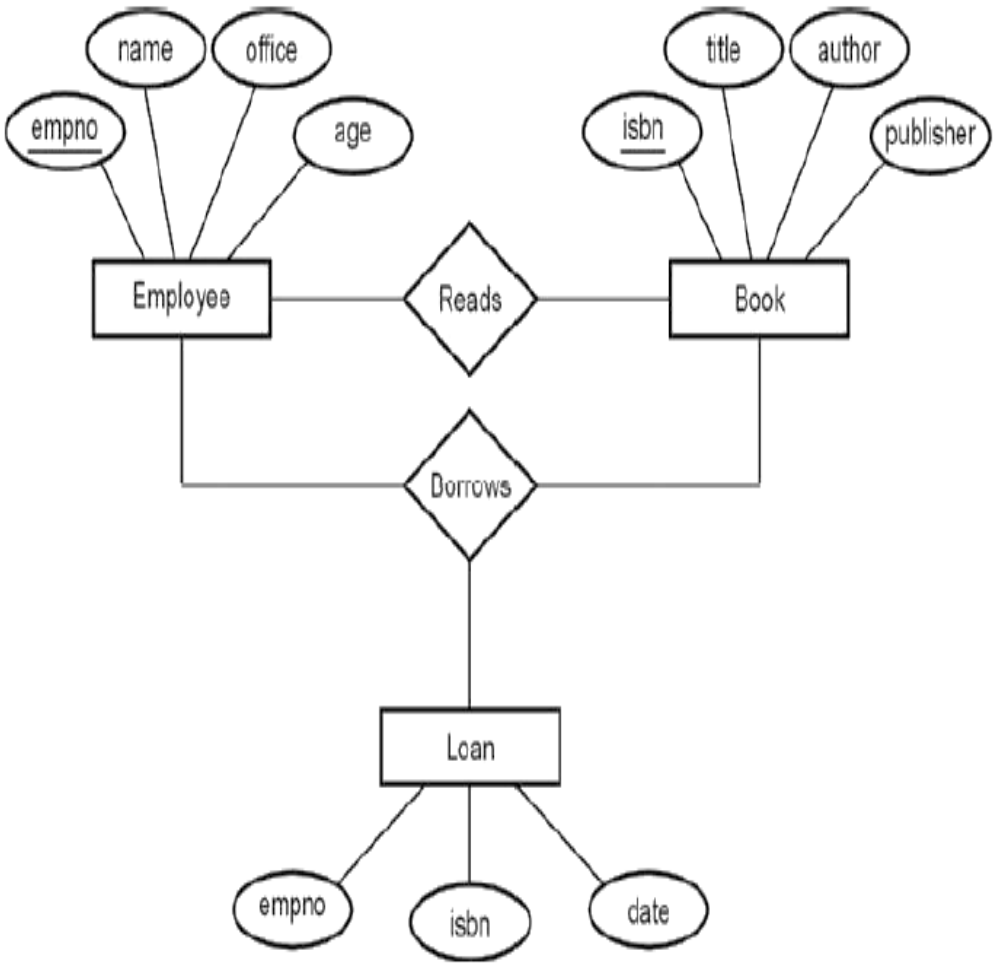


Figure 1.37 ER Diagram for Employee

18. Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for particular period of time and has an associated due date and date when the payment was received.

SOLUTION

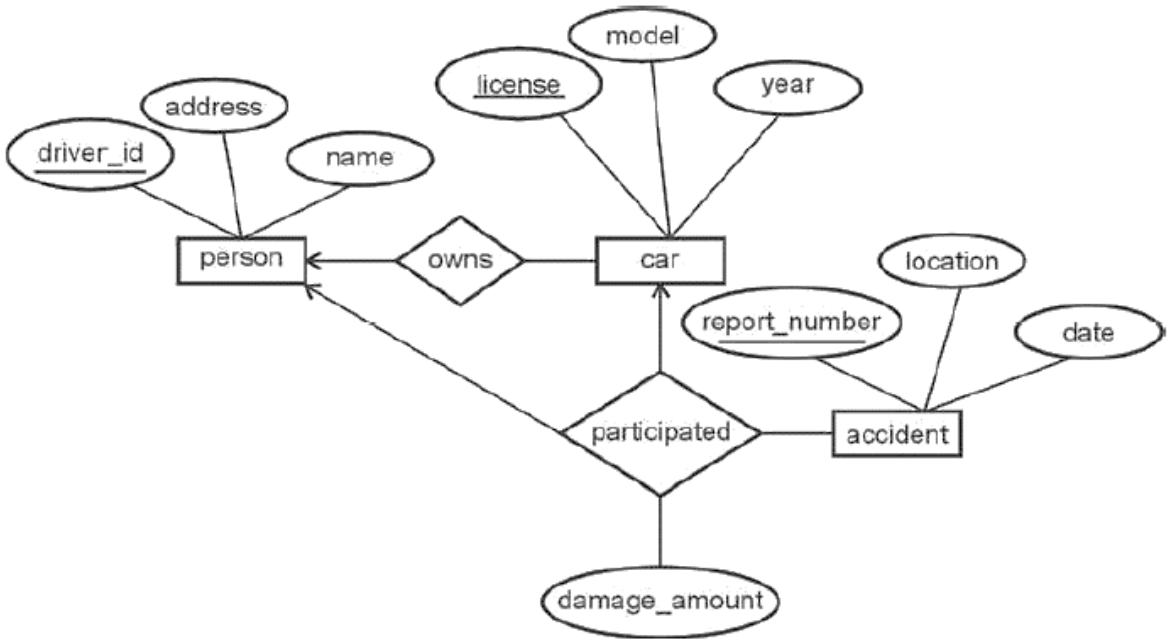


Figure 1.38 E-R diagram for a car insurance company

19. A car rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number license number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles.

Trucks: Cargo capacity

Sports cars: horsepower, renter age requirement

Vans: number of passengers

Off-road vehicles: ground clearance, drivetrain (four-or two-wheel drive)

Construct an ER model for the car rental company database.

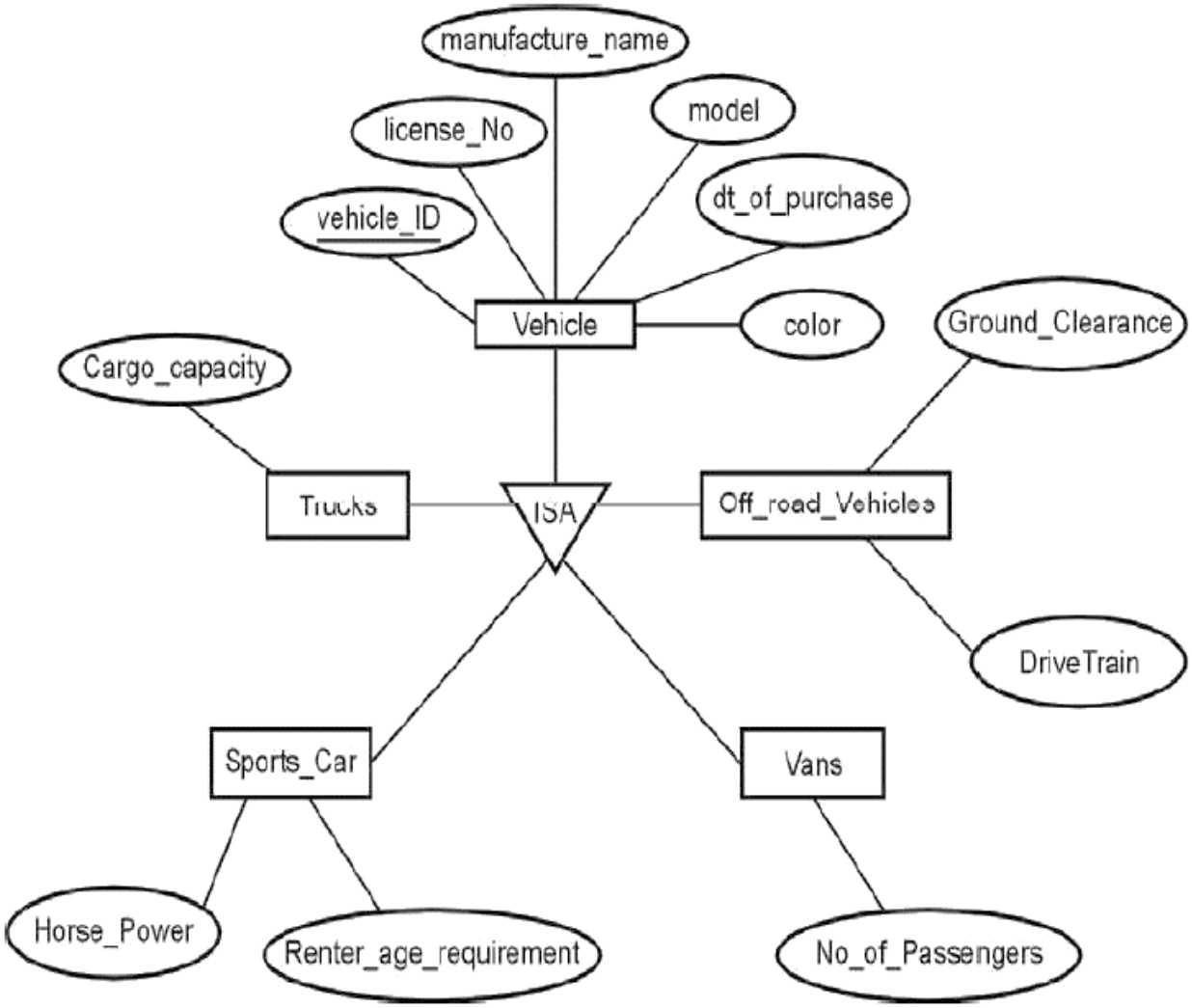


Figure 1.39 A car rental company

20. Draw E-R diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The Waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The Food preparation staffs (chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. During preparation they are able to let the waiter know the status of each item, and can send notifications when items are completed. The

system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on.

SOLUTION

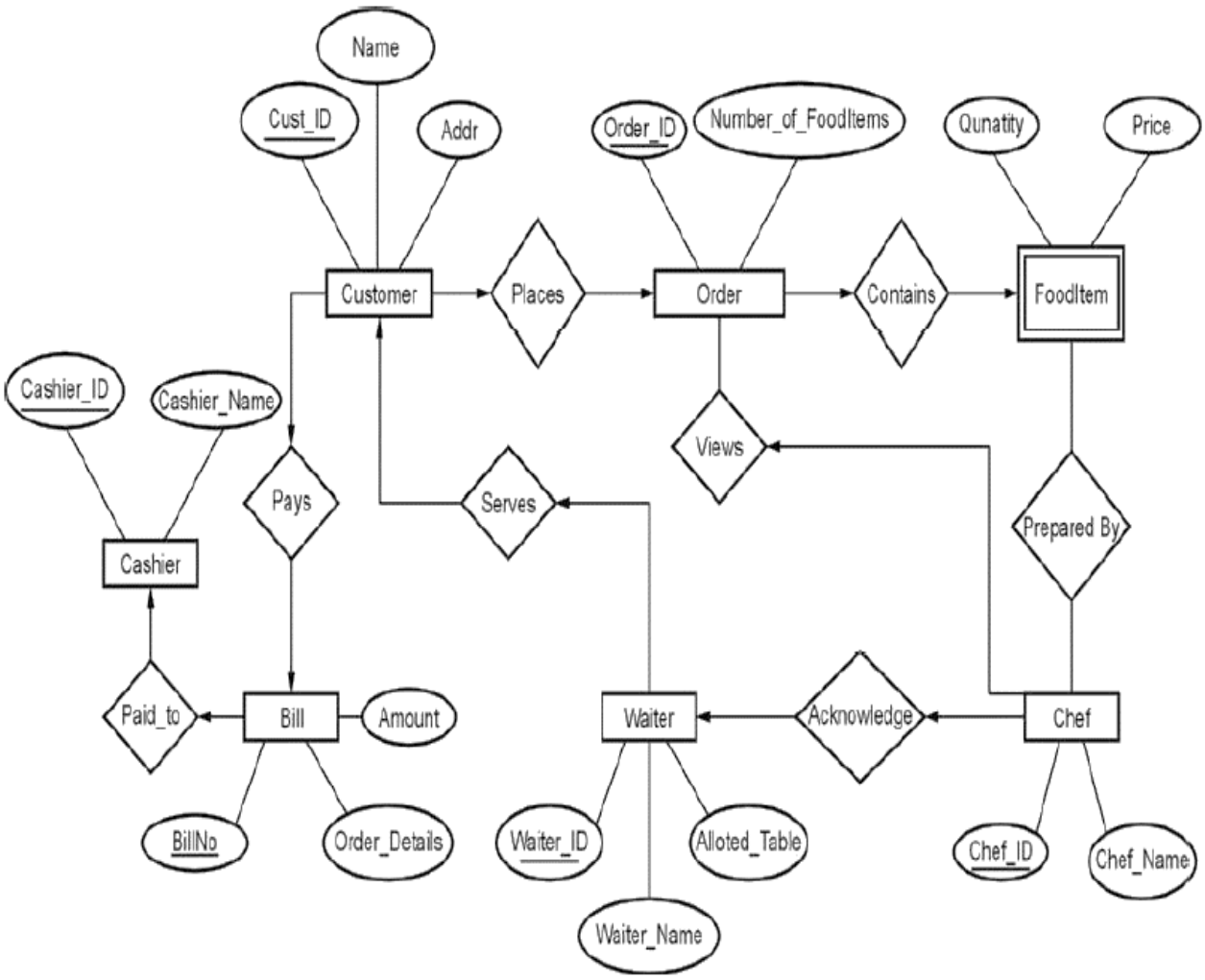


Figure 1.40 E-R diagram for the "Restaurant Menu Ordering System"

21. A university registrar's office maintains data about the following entities:
(1) courses, including number, title, credits, syllabus, and prerequisites;
(2) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
(3) students, including student-id, name, and program; and
(4) instructors, including identification number, name, department, and title. Further, the enrolment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modelled. Construct

an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints. (NOV/DEC 2023)

SOLUTION

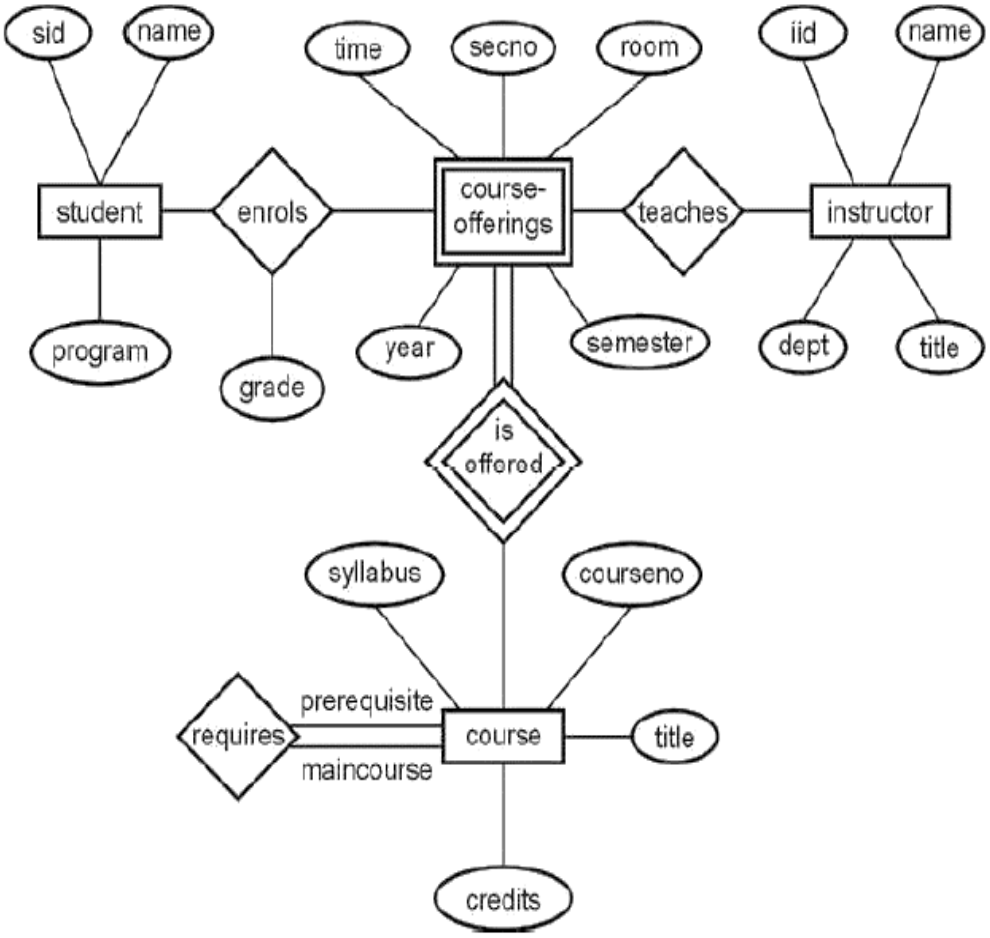


Figure 1.41 ER-Diagram for university registrar's office

22. What is aggregation in ER model? Develop an ER diagram using aggregation that captures following information: Employees work for projects. An employee working for particular project uses various machinery. Assume necessary attributes. State any assumptions you make. Also discuss about the ER diagram you have designed.

SOLUTION

ER Diagram: The ER diagram for above-described scenario can be drawn as follows -

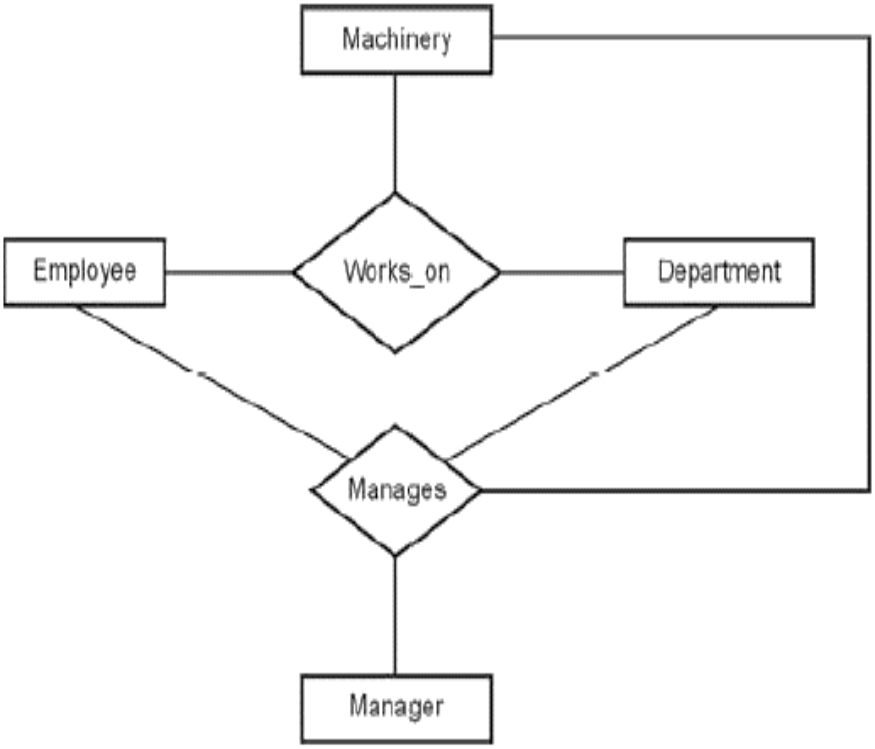


Figure 1.42 ER diagram using aggregation

- The above ER model contains the redundant information, because every Employee, Project, Machinery combination in **works_on** relationship is also considered in **manages** relationship.
- To avoid this redundancy problem, we can make use of aggregation relationship in ER diagram as follows –

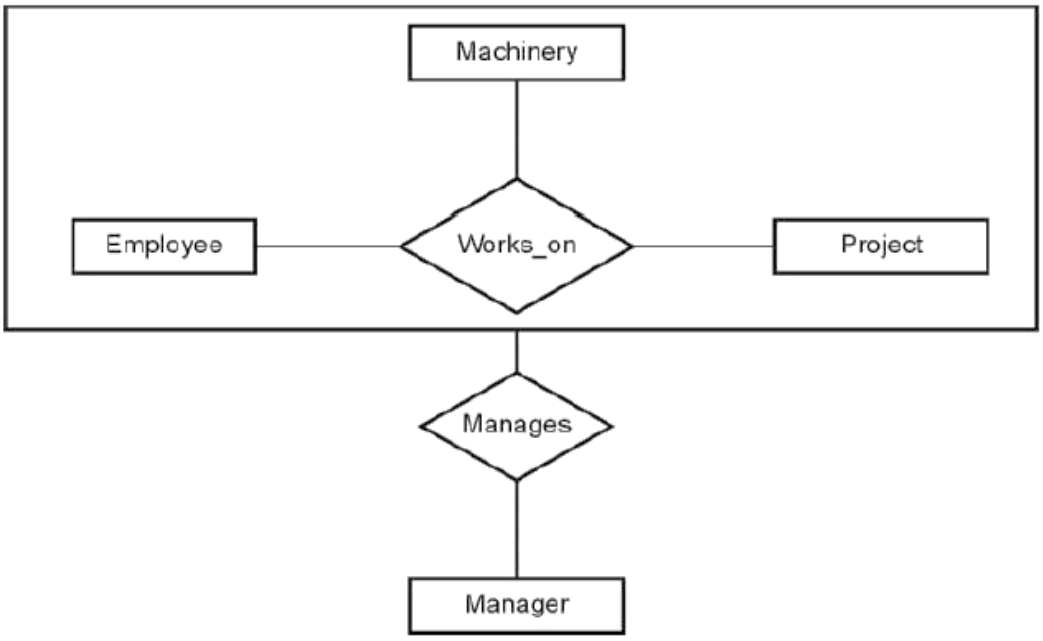


Figure 1.43 ER diagram using aggregation

- We can then create a binary relationship **manages** for between **Manager** and **(Employee, Project, Machinery).**

23. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. (APR/MAY 2024)

SOLUTION

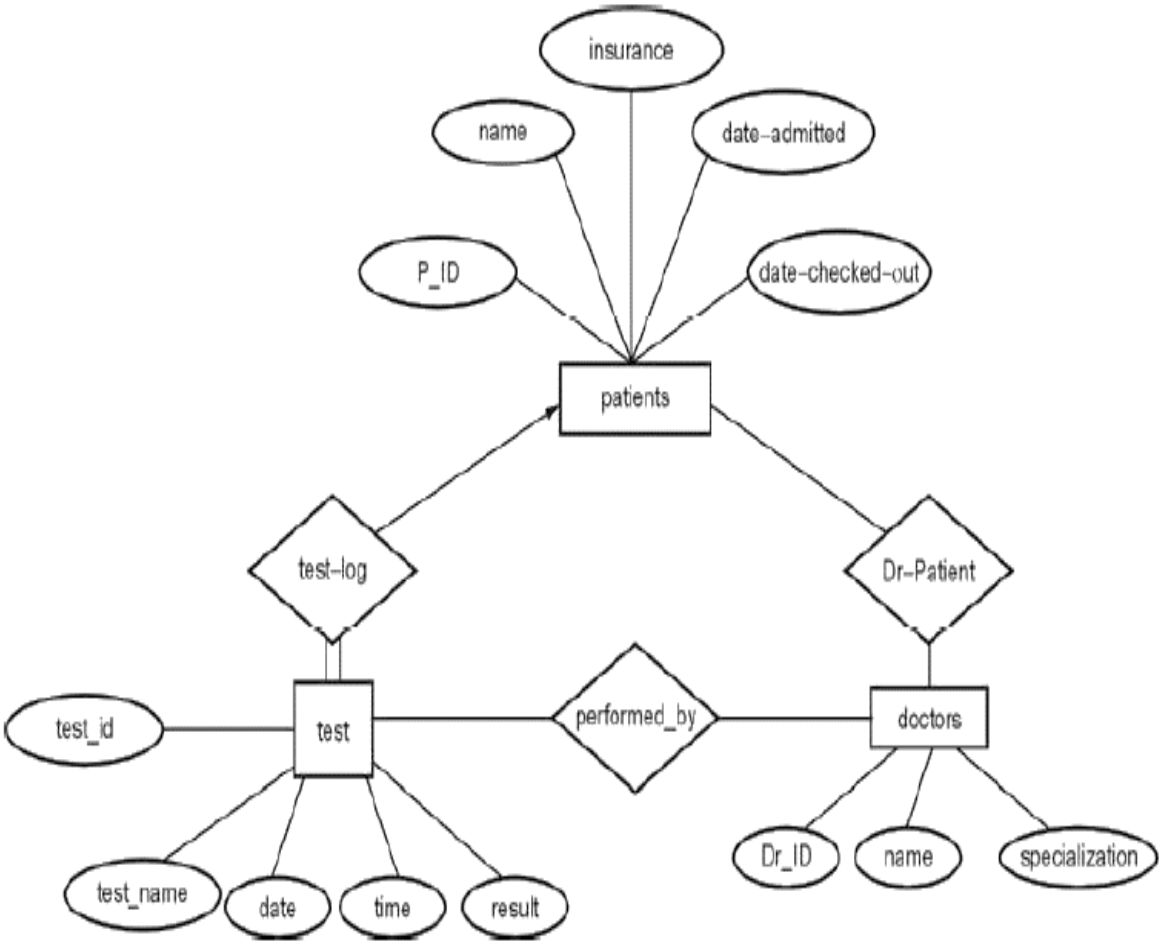


Figure 1.44 E-R diagram for a hospital

24. Describe the UML Class Diagrams with an example or Exemplify in detail about the diagrammatic representation of entity and relationship types in UML. (APR/MAY 2023)

Definition

- The Unified Modeling Language (UML) is a standard diagramming notation for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.
- The UML class diagram is used to illustrate **classes, interfaces** and their **associations.**
- The class diagram represents **static object modeling**.
- Various elements of the class diagram are –
    1, Classifier
    2. Attributes and association lines
    3. Methods or operations
    4. Generalization
    5. Dependency
    6. Composition and aggregation
- Let is discuss these concepts with the help of examples.

  1.Classifier The classifier represents the behavioural and structural features of the system. It can be specialized. The various elements in classifier are classes, interfaces, use cases and actors. In class diagram the most commonly used classifiers are classes and interfaces.

  2.Attributes and Association Lines The attributes of the classifier represent the structural properties as shown in figure 1.45. The attributes can be represented in various ways such as
    - Attribute text
    - Association Line
    - Both together

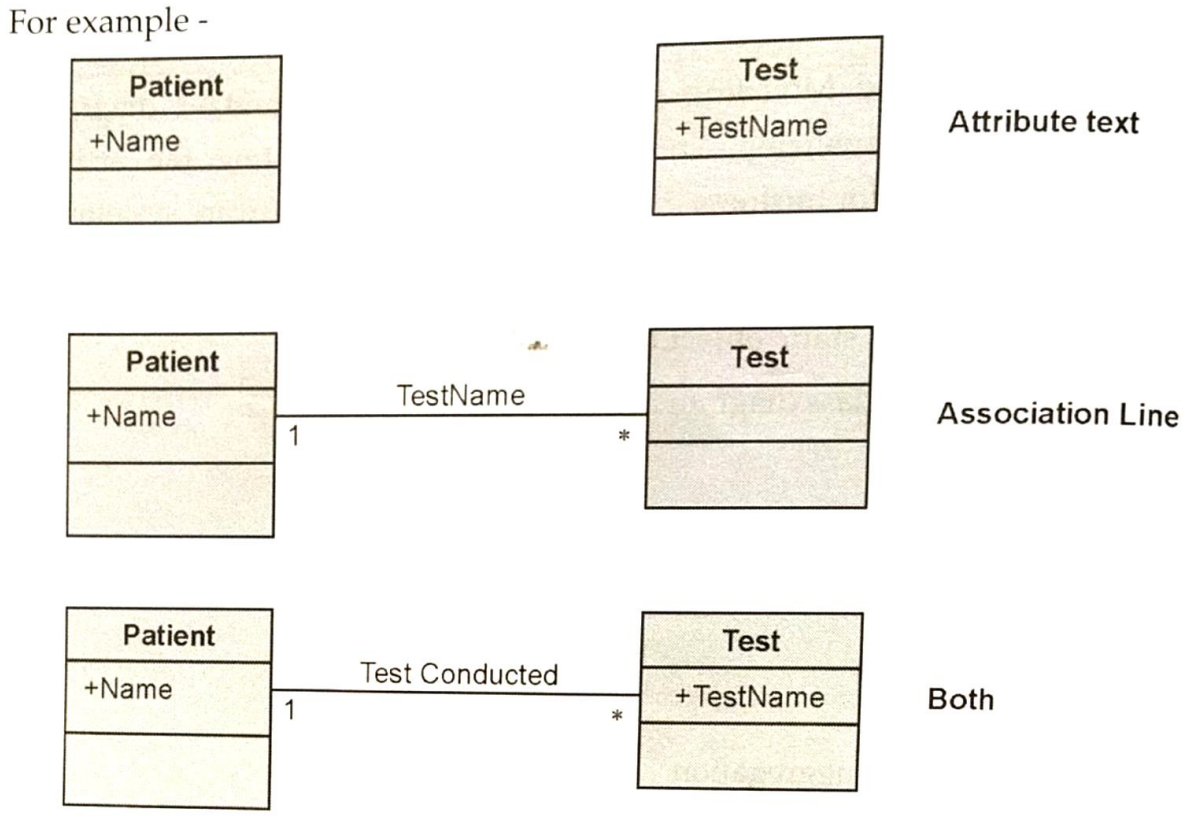


Figure1.45 Attributes and association line

- The full format of attribute text is Visibility Name: type multiplicity=default{property-string}.
- The visibility means public (+), private (-) or protected (#). It denotes the modes of accessibility of the attribute. Usually, the attributes have the private visibility mode.
- The multiplicity can be 1.1, 1. * and so on. It is denoted at the end of
- association line.
- While using the association relationship the role name is mentioned which denotes the association(relationship) between the two objects.
- The navigation arrow is used to represent the association between two classes.
- The relationship between two classes can be represented by either attributes or by associations. But when an object can be described by certain properties which are associated with primitive data types then such properties are denoted by the attributes of that object. When there is a relationship between two objects that has visual emphasis then those objects are connected by the **association line** as shown in figure 1.46.

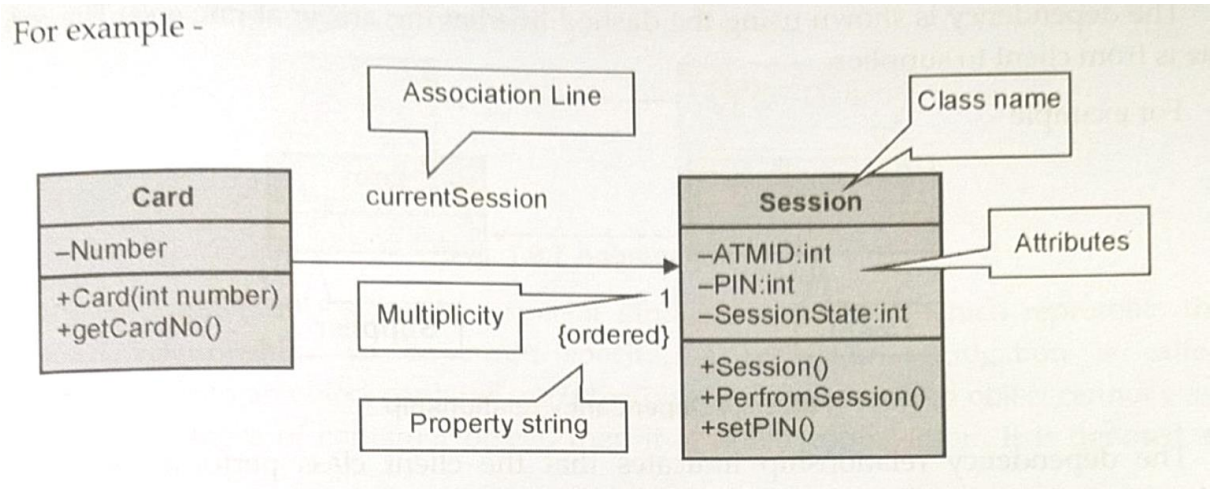


Figure1.46 Class

3.Methods and Operations

- For denoting the implementation, the methods are used in the class.
- These methods can be accessed by the object of a class. Refer Figure. 1.40 in which the entities in the third compartment of the class are all methods.

4.Generalization

- Generalization is a relationship between the superclass and subclass. This is "is a" relationship.
- A generalization is shown as a line with a hollow triangle as an arrowhead between the symbols representing the involved classifiers.
- The arrowhead points to the symbol representing the general classifier.
- **For example -** In banking system the generalization relationship is represented as shown in figure 1.47.

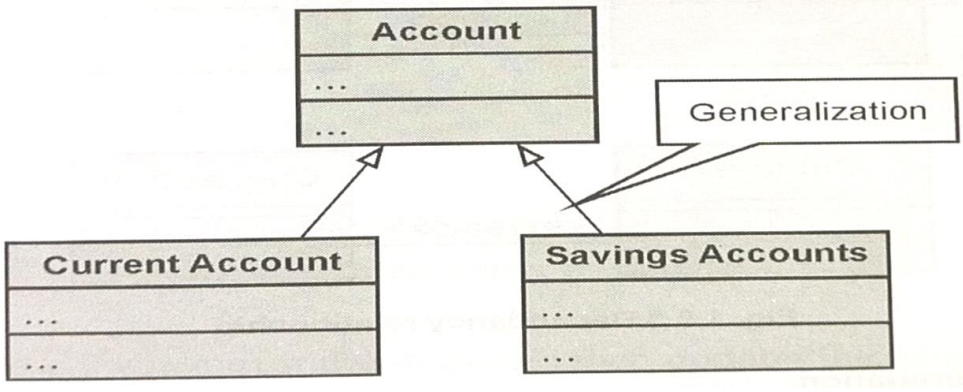


Figure1.47 Generalization relationship

- By the software perspective this relationship implies the inheritance concept of object-oriented programming.

5.Dependency

- The dependency link is most commonly used in class and package diagram. In this relationship the client element must have the knowledge of another supplier element.
- The dependency is shown using the dashed line having arrow at one end. The arrow line is from client to supplier as shown in figure 1.48.
- **For example,**

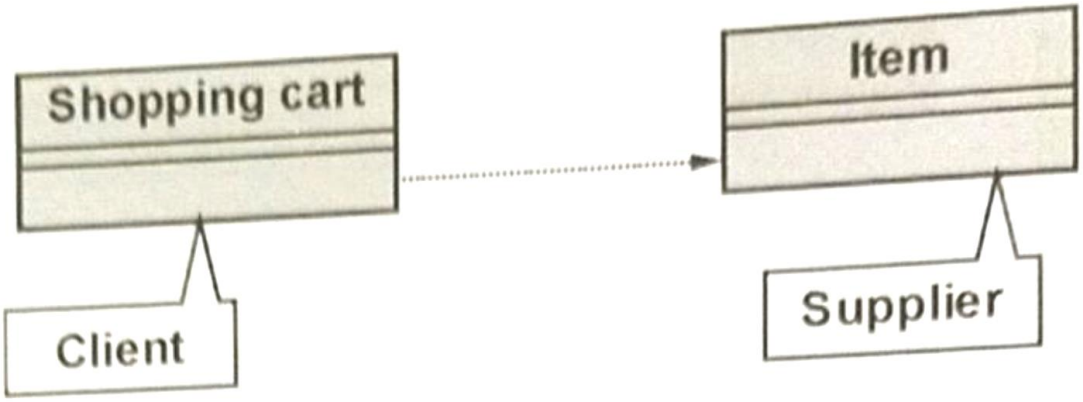


Figure.1.48 Dependency relationship

- The dependency relationship indicates that the client class performs one of the following functions - Temporarily uses a supplier class that has global scope.
- Temporarily uses a supplier class as a parameter for one of its operations Temporarily uses a supplier class as a local variable for one of its operations The dependency line can be labeled using the keywords or stereotypes. Various
- keywords or stereotypes that can be used in dependency relationship are -
- <<bind>>, <<realize> >, <<substitute> >, <trance>>, <<derive>>, <<refine>>, <<use>>, <<call> >, <<create>>, <<send>>

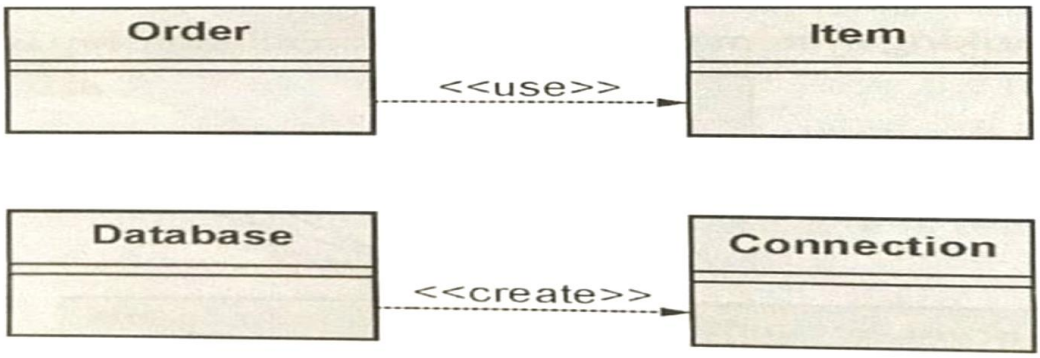


Figure.1.49 Dependency relationship

6.Composition and Aggregation

- Aggregation is used to represent the whole-part relationship. It normally possesses the Has-a relationship.
- It is denoted as follows-

For example: The aggregate relationship is

Figure.1.50 Aggregation

Composition:

- Composition is a special kind of aggregation which represents the whole-part relationship.
- To be more specific, a restricted aggregation is called composition.
- When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.
- It is denoted as follows –

- **For example: The composite relationship for ATM system is Solution: Problem description**

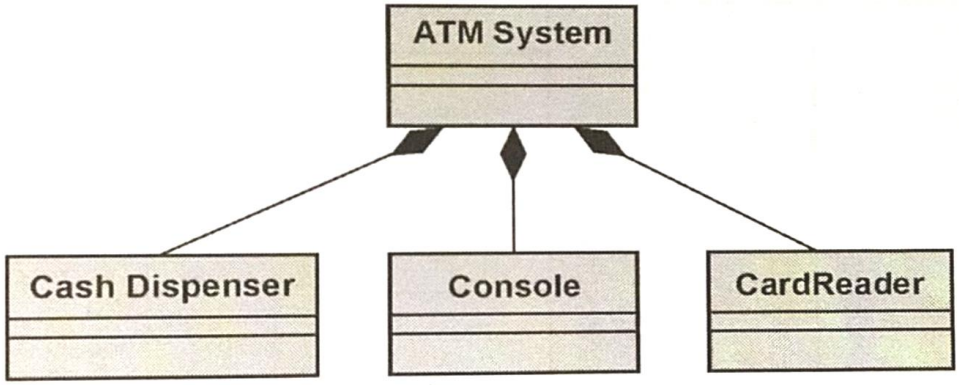


Figure1.51 Composition

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai,

Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / III SEM

UNIT II – RELATIONAL MODEL AND SQL

SYLLABUS:

Relational model concepts — Integrity constraints — SQL Data manipulation- SQL Data definition – Views — SQL programming.

PART A

1. What are the categories of SQL command? (NOV/DEC 2023)

- SQL commands are divided into the following categories:
  1. Data Definition Language (DDL)
  2. Data Manipulation Language (DML)
  3. Data Query Language (DQL)
  4. Data Control Language (DCL)
  5.Transaction Control Language (TCL)

2. What are the three classes of SQL expression?

- SQL expression consists of three clauses:
  a) Select
  b) From
  c) Where

3. List the string operations supported by SQL.

a) Pattern matching Operation
b) Concatenation
c) Extracting character strings
d) Converting between uppercase and lowercase letters.

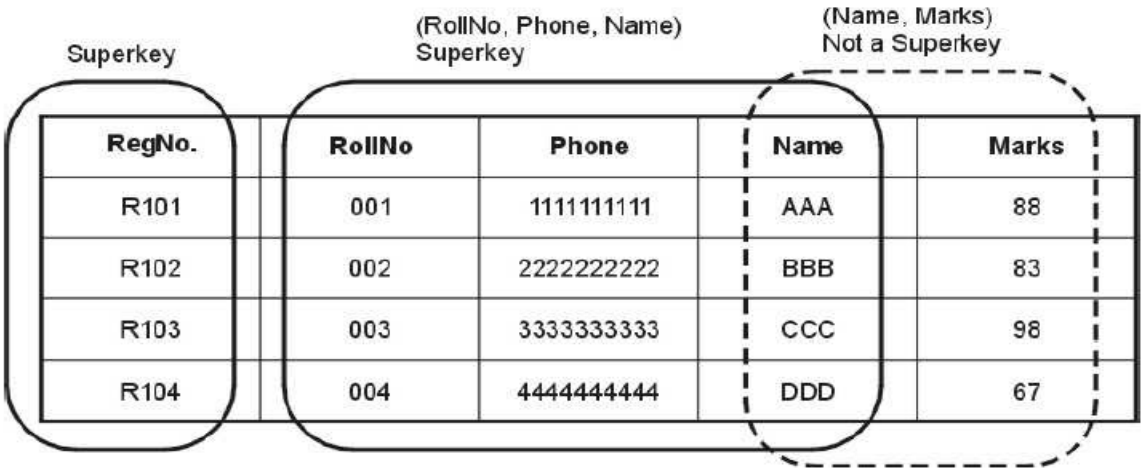
PREPARED BY: Mrs. S. JAYABHARATHI AP/AI&DS

4. What are aggregate functions? And list the aggregate functions supported by SQL.

- Aggregate functions are functions that take a collection of values as input and return a single value.
- Aggregate functions supported by SQL are
  a) Average: avg
  b) Minimum: min
  c) Maximum: max
  d) Total: sum
  e) Count: count

5. Define Super Key with example.

- It is a set of one or more attributes within a table that can uniquely identify each record within a table.
- The super key can be represented as follows:



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Super Key

6. Define Degree. Give an example.

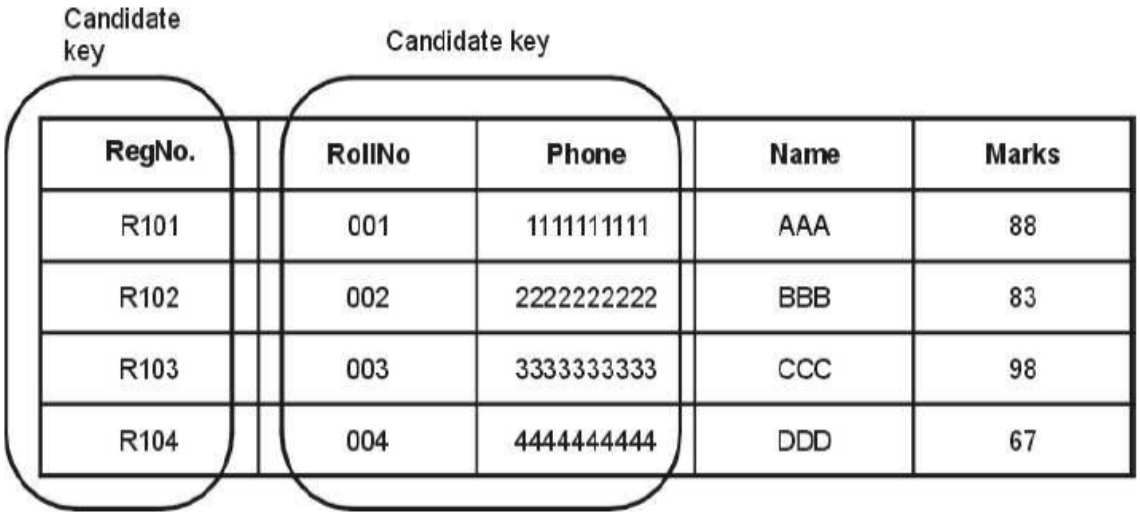
- It is nothing but total number of columns present in the relational database. In given Student table –

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

- The degree is 4.

7. Define Candidate Key with example. (NOV/DEC 2023)

- The candidate key is a subset of superset.
- In other words, candidate key is a single attribute or least or minimal combination of attributes that uniquely identify each record in the table.
- **Example:**



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Candidate Key

8. Define Tuple with an example.

- The single entry in the table is called tuple. The tuple represents a set of related data.
- **Example:** One of the tuples can be represented as

001	AAA	88	1111111111

9. What is NULL Attribute. Give an example.

- A null is a special symbol, independent of data type, which means either unknown or inapplicable.
- It does not mean zero or blank.
- **For example -** Consider a salary table that contains NULL.

Emp#	Job Name	Salary	Commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

10. List out the statements associated with a database transaction.

- COMMIT
- ROLLBACK
- SAVEPOINT

11. Define Cardinality with example.

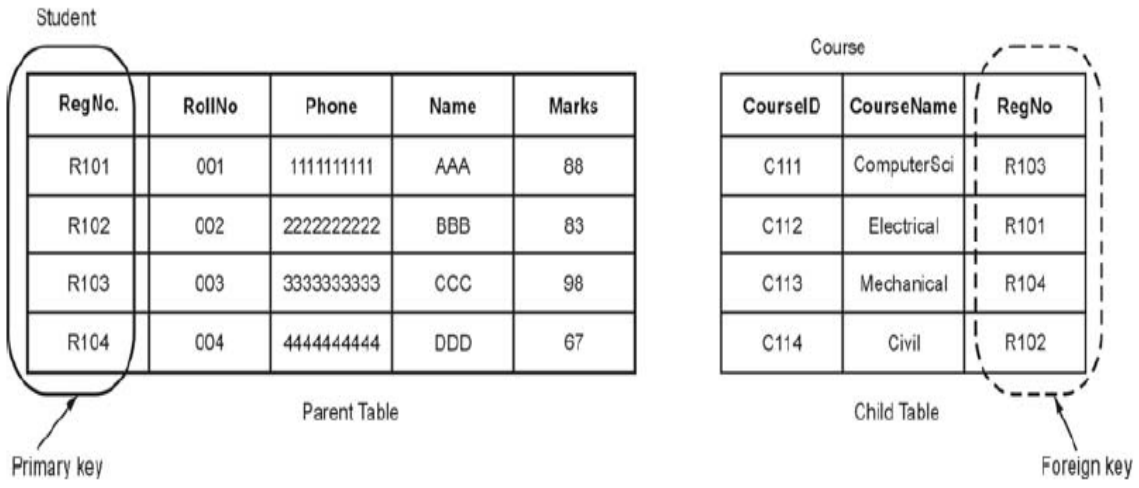
- It is total number of tuples present in the relational database.
- **Example:** Consider the table,

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

- The cardinality for the above table is 3.

12. Define Foreign Key. Give an example.

- Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of another table.
- Thus, foreign keys refer to primary key.
- The table containing the primary key is called **parent table** and the table containing foreign key is called **child table**.
- **Example:**



Example-Foreign Key

13. What is Join. Also, List the types of Joins.

- The join operation is used to **combine information from two or more relations**.
- Formally join can be defined as a cross-product followed by selections and projections, joins arise much more frequently in practice than plain cross-products. The join operator is used as

- **Types of Joins:**
  - Conditional Join
  - Equi-Join
  - Natural Join

14. Define Entity Integrity Rule.

- **This rule states that** "In the relations, the value of attribute of primary key cannot be null".
- The NULL represents a value for an attribute that is currently unknown or is not applicable for this tuple.
- The Nulls are always to deal with incomplete or exceptional data.
- The primary key value helps in uniquely identifying every row in the table.
- Thus, if the users of the database want to retrieve any row from the table or perform any action on that table, they must know the value of the key for that row.
- Hence it is necessary that the primary key should not have the NULL value.

15. Define Table or Relation. Give an example.

- In relational model, table is a collection of data items arranged in rows and columns.
- The table cannot have duplicate data or rows. Below is an example of student table.

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333
004	DDD	67	4444444444

16. Define Keys. Also, List the types of Keys.

- Keys are used to specify the tuples distinctly in the given relation.
- Various types of keys used in relational model are –
  1. Super Key
  2. Candidate Key
  3. Primary Key
  4. Alternate Key
  5. Foreign Key

17. Define Primary Key. Give an example. (NOV/DEC 2023)

- The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely.
- **For example –** Consider the following representation of primary key in the student table

Primary key

RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Primary Key

18. Define Alternate Key with an example.

- The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tuples. **For example –**

Primary key

Alternate key

RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Alternate Key

19. Write down the Rules for Primary Key.

- The primary key may have one or more attributes.
- There is **only one primary key** in the relation.
- The value of primary key attribute cannot be NULL.
- The value of primary key attribute does not get changed.

20. Define Selection.

- This operation is used to fetch the **rows or tuples** from the table(relation).
- **Syntax:** The syntax is

$$\sigma_{predicate}(relation)$$

- where σ represents the select operation. The predicate denotes some logic using which the data from the relation(table) is selected.

21. Define Projection.

- Project operation is used to project only a certain set of attributes of a relation.
- That means if you want to see only the names all of the students in the student table, then you can use Project operation.
- Thus, to **display particular column** from the relation, the projection operator is used.
- It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.
- **Syntax:**

$$\Pi C1, C2\ldots (r)$$

- where C1, C2 etc. are attribute names (column names).

22. Define Cartesian Product.

- This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.

$$\text{Syntax} : A \times B$$

23. What is Referential Integrity? (APR/MAY 2023)

- Referential integrity refers to the **accuracy and consistency** of data within a relationship.
- In relationships, data is linked between two or more tables. This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary - or parent - table). Because of this, we need to ensure that data on both sides of the relationship remains intact.
- **The referential integrity rule states that** "whenever **a foreign key** value is used it must **reference a valid, existing primary** key in the parent table".

24. What is DCL Command?

- **DCL commands:** It stands for Data Control Language. It includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
- **Examples of DCL commands:**
    1. GRANT-gives user's access privileges to database.
    2. REVOKE-withdraw user's access privileges given by using the GRANT command.

25. Define Nested Queries.

- In nested queries, a **query is written inside a query**. The result of inner query is used in execution of outer query.
- There are two types of nested queries:
    1. **Independent Query**
    2. **Co-related Query**

PART B

1. Explain in detail about Relational Databases with suitable examples.

Introduction to Relational Databases

- Relation database is a **collection of tables** having unique names.
- **For example –** Consider the example of Student table in which the information about the student is stored.

RollNo	Name	Phone
001	AAA	1111111111
002	BBB	2222222222
003	CCC	3333333333

- The above table consists of three column headers RollNo, Name and Phone.
- Each row of the table indicates the information of each student by means of his Roll Number, Name and Phone number.
- Similarly consider another table named Course as follows –

CourseID	CourseName	Credits
101	Mechanical	4
102	Computer Science	6
103	Electrical	5
104	Civil	3

- Clearly, in above table the columns are **CourseID, CourseName** and **Credits.**
- The CourseID **101** is associated with the course named **Mechanical** and associated with the course of mechanical there are **4 credit points.**
- Thus, the relation is represented by the table in the relation model. Similarly, we can establish the relationship among the two tables by defining the third table. **For example –**
- Consider the table Admission as

RollNo	CourseID
001	102
002	104
003	101

- From this third table we can easily find out that the course to which the RollNo 001 is admitted is computer Science.

Relational Model

- There are some commonly used terms in Relational Model and those are -

Table or relation:

- In relational model, table is a collection of data items arranged in rows and columns. The table cannot have duplicate data or rows. Below is an example of student table

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333
004	DDD	67	4444444444

Tuple or record or row:

- The single entry in the table is called tuple.
- The tuple represents a set of related data.
- In above **Student** table there are four tuples.
- One of the tuples can be represented as

001	AAA	88	1111111111

Attribute or columns:

- It is a part of table that contains several records.
- Each record can be broken down into several small parts of data known as attributes.
- **For example,** the above table consists of four attributes such as **RollNo, Name, Marks** and **Phone**.

Relation schema:

- A relation schema describes the structure of the relation, with the name of the relation (i.e. name of table), its attributes and their names and type.

Relation Instance:

- It refers to specific instance of relation i.e. containing a specific set of rows. **For example –** the following is a relation instance – which contains the records with marks above 80.

RollNo	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

Domain:

- For each attribute of relation, there is a set of permitted values called domain.
- **For example –** in above table, the domain of attribute **Marks** is set of all possible permitted marks of the students.
- Similarly, the domain of **Name** attribute is all possible names of students.
- That means Domain of Marks attribute is (88,83,98)

Atomic:

- The domain is atomic if elements of the domain are considered to be indivisible units.
- For example, in above **Student** table, the attribute **Phone** is non-atomic.

NULL attribute:

- A null is a special symbol, independent of data type, which means either unknown or inapplicable.
- It does not mean zero or blank. For example - Consider a salary table that contains NULL

Emp#	Job Name	Salary	Commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

Degree:

- It is nothing but total number of columns present in the relational database.

  In given Student table –

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

- The degree is 4.

Cardinality:

- It is total number of tuples present in the relational database.
- In above given table the cardinality is 3.

Example: Find out following for given Staff table

i) No of Columns

ii) No of tuples

iii) Different attributes

iv) Degree

v) Cardinality

StaffID	Name	Sex	Designation	Salary	DOJ
S001	John	M	Manager	50000	1 Oct. 2012
S002	Ram	M	Executive	20000	20 Jan. 2015
S003	Meena	F	Supervisor	40000	12 Aug. 2011

Solution:

i) No of Columns = 6

ii) No of Tuples= 3

iii) Different attributes are StaffID, Name, Sex, Designation, Salary, DOJ

iv) Degree= Total number of columns=6

v) Cardinality =Total number of rows = 3

2. Define Keys. Also explain distinction among the terms primary key, candidate key, foreign key and super key with suitable example.

Keys

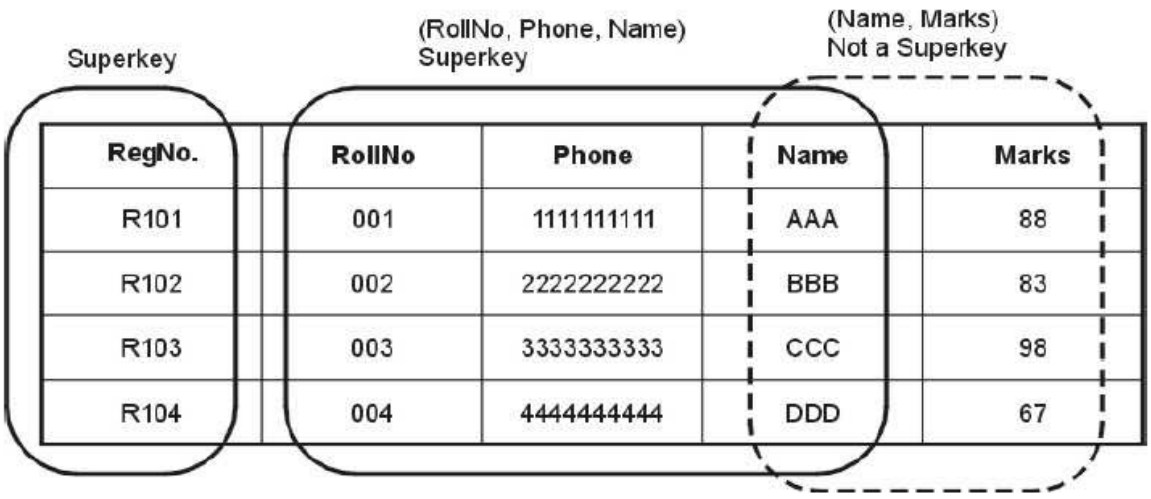
- Keys are used to specify the tuples distinctly in the given relation.
- Various types of keys used in relational model are – Super key, Candidate Keys, primary keys, foreign keys. Let us discuss them with suitable example

1) Super Key (SK):

- It is a set of one or more attributes within a table that can uniquely identify each record within a table.
- **For example –** Consider the Student table as follows –

Reg No.	Roll No	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

- The super key can be represented as follows

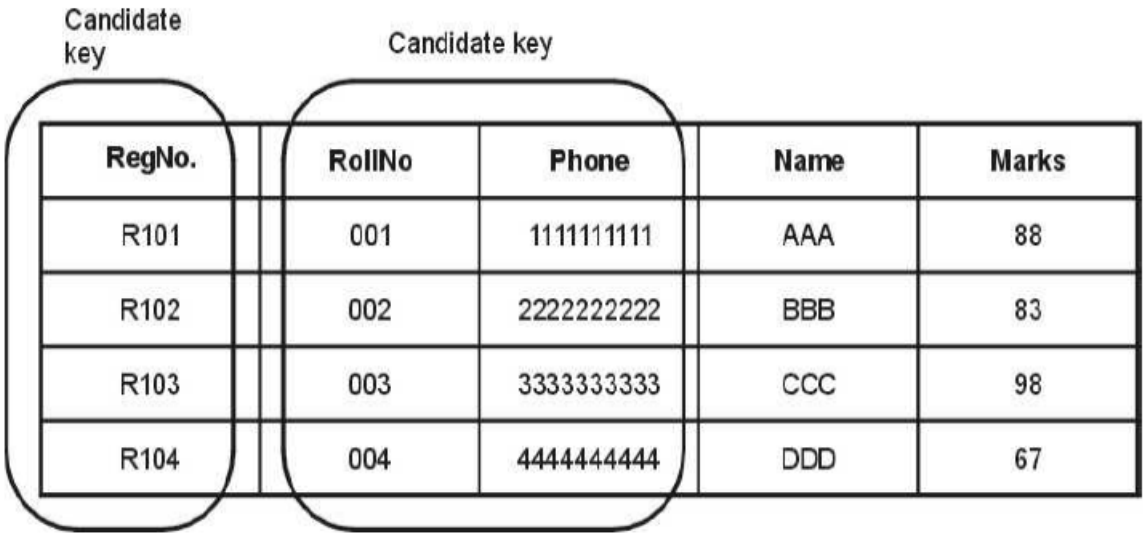


Example- Super Key

- Clearly using the (RegNo) and (RollNo, Phone, Name) we can identify the records uniquely but (Name, Marks) of two students can be same, hence this combination not necessarily help in identifying the record uniquely.

2) Candidate Key (CK):

- The candidate key is a subset of superset. In other words candidate key is a single attribute or least or minimal combination of attributes that uniquely identify each record in the table.
- **For example -** in above given **Student** table, the candidate key is RegNo, (RollNo, Phone). The candidate key can be key can be



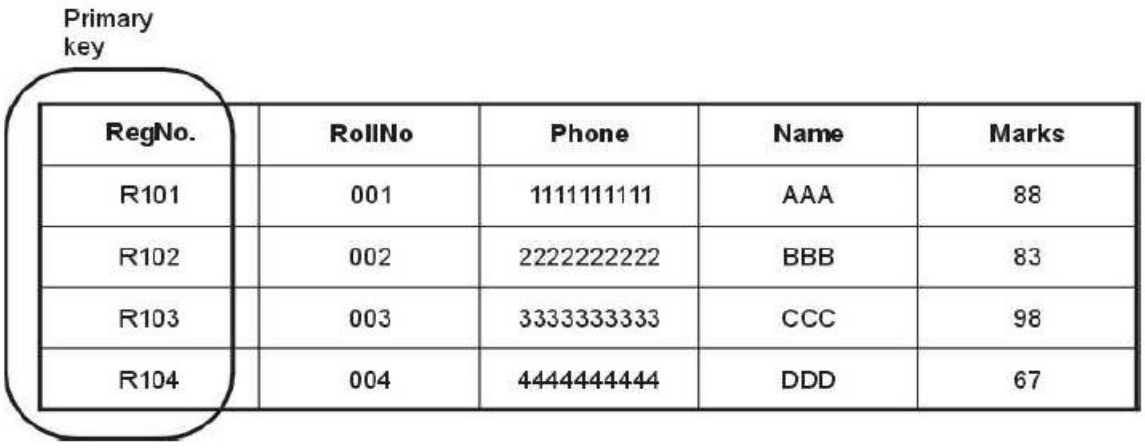
RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Candidate Key

- Thus, every candidate key is a super key but every super key is not a candidate key.

3) Primary Key (PK):

- The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely.
- **For example –** Consider the following representation of primary key in the student table



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Example- Primary Key

- Other than the above-mentioned primary key, various possible primary keys can be **(RollNo), (RollNo, Name), (RollNo, Phone)**
- The relation among super key, candidate key and primary can be denoted by **Candidate Key=Super Key – Primary Key**

Rules for Primary Key

(i) The primary key may have one or more attributes.

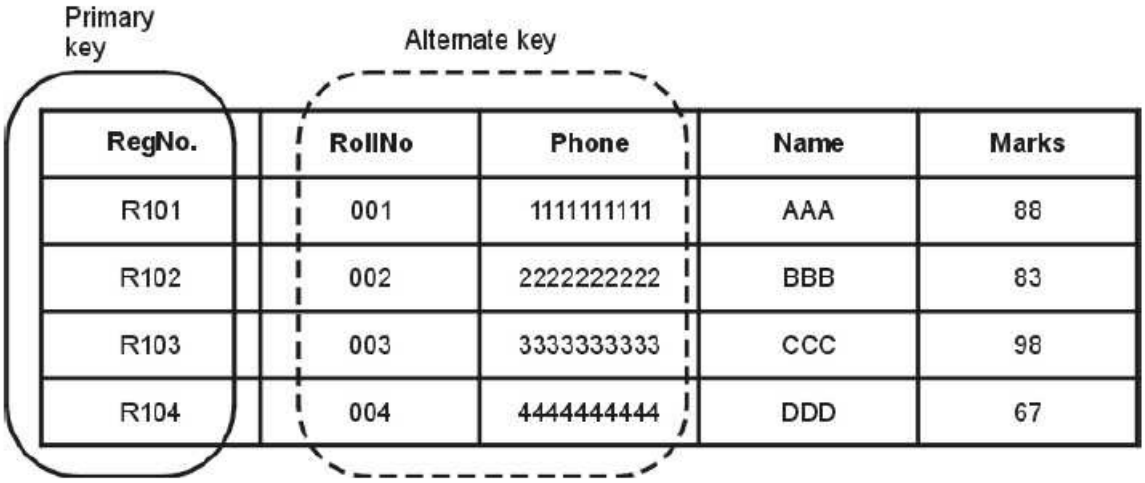
(ii) There is **only one primary key** in the relation.

(iii) The value of primary key attribute cannot be NULL.

(iv) The value of primary key attribute does not get changed.

4) Alternate key:

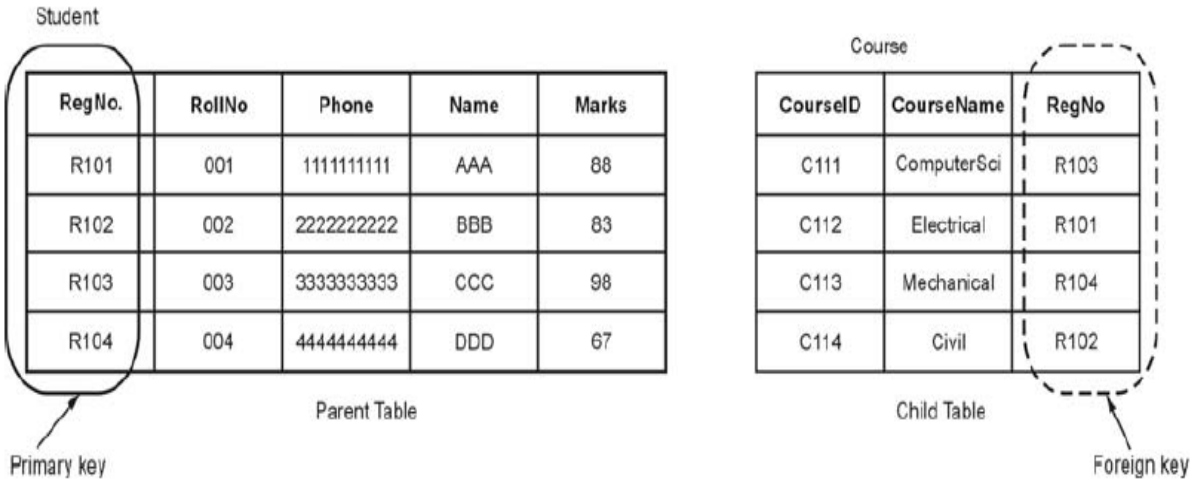
- The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tuples. **For example –**



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

5) Foreign key:

- Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of another table.
- Thus, foreign keys refer to primary key.
- The table containing the primary key is called **parent table** and the table containing foreign key is called **child table**. **Example –**



- From above example, we can see that two tables are linked.
- For instance, we could easily find out that the 'Student CCC has opted for ComputerSci course.

3. Discuss the entity Integrity and referential integrity constraints. Why are they important? Explain them with suitable examples.

Integrity Constraints

- Database integrity means correctness or accuracy of data in the database. A database may have number of integrity constraints. For example –
  (i) The Employee ID and Department ID must consist of two digits.
  (ii) Every Employee ID must start with letter.
- The integrity constraints are classified based on the concept of primary key and foreign key. Let us discuss the classification of constraints based on primary key and foreign key as follows –

Entity Integrity Rule

- **This rule states that** "In the relations, the value of attribute of primary key cannot be null".
- The NULL represents a value for an attribute that is currently unknown or is not applicable for this tuple.
- The Nulls are always to deal with incomplete or exceptional data.
- The primary key value helps in uniquely identifying every row in the table.
- Thus, if the users of the database want to retrieve any row from the table or perform any action on that table, they must know the value of the key for that row.
- Hence it is necessary that the primary key should not have the NULL value.

Referential Integrity Rule

- Referential integrity refers to the **accuracy and consistency** of data within a relationship.
- In relationships, data is linked between two or more tables. This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary - or parent - table). Because of this, we need to ensure that data on both sides of the relationship remain intact.
- **The referential integrity rule states that** "whenever **a foreign key** value is used it must **reference a valid, existing primary** key in the parent table".
- **Example:** Consider the situation where you have two tables: **Employees** and **Managers**. The **Employees** table has a foreign key attribute entitled **ManagedBy**, which points to the record for each employee's manager in the **Managers** table.
- Referential integrity enforces the **following three rules:**

  i) You cannot add a record to the **Employees** table unless the **ManagedBy** attribute points to a valid record in the **Managers** table. Referential integrity prevents the insertion of incorrect details into a table. Any operation that doesn't satisfy referential integrity rule fails.

  ii) If the primary key for a record in the **Managers** table changes, all corresponding records in the **Employees** table are modified.

  iii) If a record in the **Managers** table is deleted, all corresponding records in the **Employees** table are deleted.

Advantages of Referential Integrity

Referential integrity offers following advantages:

i) Prevents the entry of duplicate data.

ii) Prevents one table from pointing to a non-existent field in another table.

iii) Guaranteed consistency between "partnered" tables.

iv) Prevents the deletion of a record that contains a value referred to by a foreign key in another table.

v) Prevents the addition of a record to a table that contains a foreign key unless there is a primary key in the linked table.

Database integrity

- The foreign key is a key in one table that refers to the primary key of another table.
- The foreign key is basically used to link two tables. **For example –**
- Consider Customer table as follows –

Customer

CustID	Name	City
C101	AAA	Chennai
C102	BBB	Mumbai
C103	CCC	Pune

Order

OrderID	Description	CustID
111	Bolts	C103
222	Nuts	C103
333	Beams	C101
444	Screws	C102
555	Disks	C101

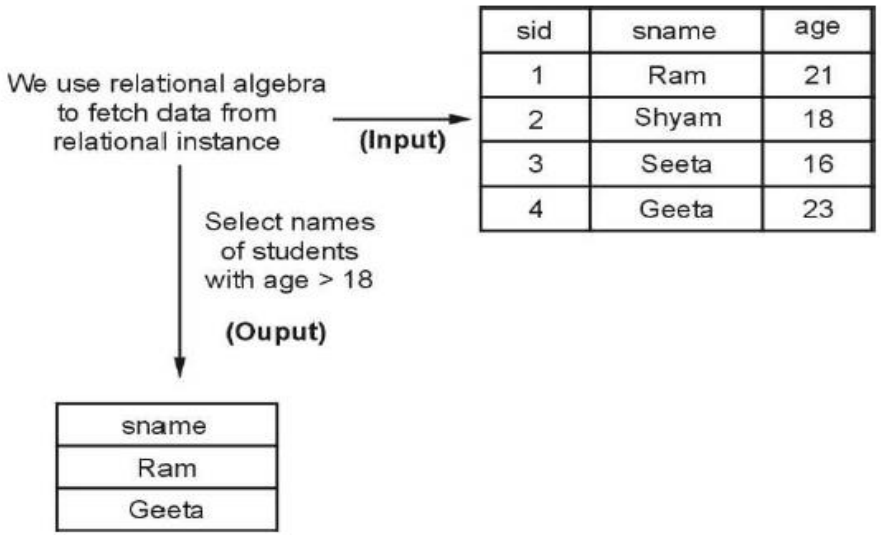
Note that the "CustID" column in the "Order" table points to the "CustID" column in the "Customer" table.

- The "CustID" column in the "Customer" table is the PRIMARY KEY in the "Customer" table.
- The "CustID" column in the "Order" table is a FOREIGN KEY in the "Order" table.
- The table containing the foreign key is called the **child table**, and the table containing the primary key is called the **referenced** or **parent table**.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to. `

4. Explain select, project, cartesian product and join operations in relational algebra with an example. or Describe in detail about the relationship between mathematical relations and relations in relational data model. (APR/MAY 2023) OR Is relational algebra procedural or non-procedural. Explain the operations with example. APR/MAY 2024)

Relational Algebra

- There are two formal query languages associated with relational model and those are relational algebra and relational calculus.
- **Definition:** Relational algebra is a procedural query language which is used to access database tables to read data in different ways.
- The queries present in the relational algebra are denoted using operators.
- Every operator in relational algebra accepts relational instances (tables) as input and returns relational instance as output. **For example:**



sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

sname
Ram
Geeta

- Each **relational algebra is procedural**. That means Each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query.
- A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query.

Relational Operations

- Various types of relational operations are as follows –

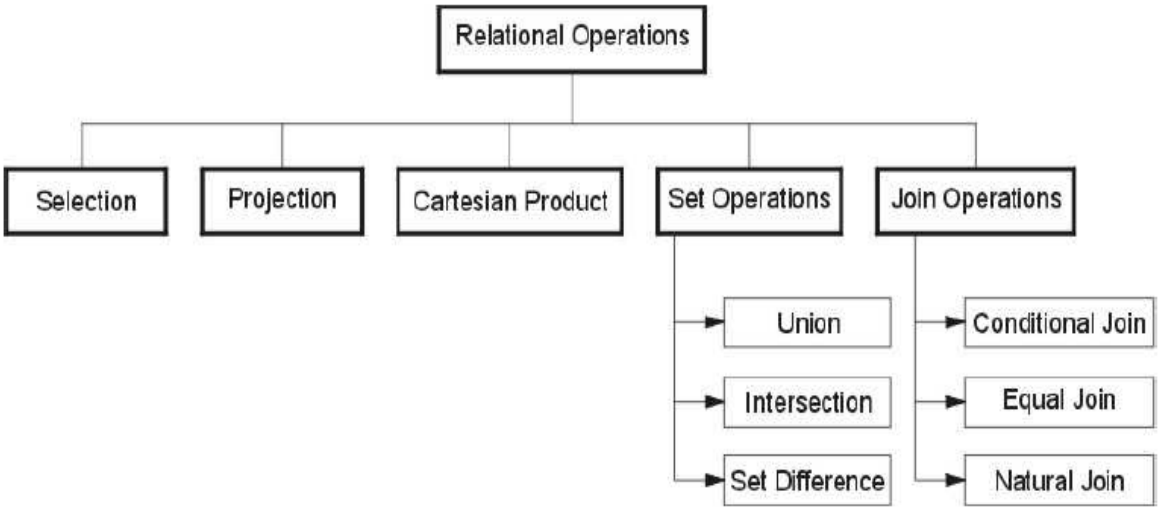


Figure 2.1 Various types of relational operations

1) Selection:

- This operation is used to fetch the **rows or tuples** from the table(relation).
- **Syntax:** The syntax is

$$\sigma_{predicate}(relation)$$

- where σ represents the select operation. The predicate denotes some logic using which the data from the relation(table) is selected.
- **For example -** Consider the relation student as follows

sid	sname	age	gender
1	Ram	21	Male
2	Shyam	18	Male
3	Seeta	16	Female
4	Geeta	23	Female

- **Query:** Fetch students with age more than 18
- We can write it in relational algebra as

$$\sigma_{age>18}(Student)$$

- The output will be –

sname
Ram
Geeta

- We can also specify conditions using and, or operators.

$$\sigma_{age>18 \text{ and } gender='Male'}(Student)$$

sname
Ram

(2) Projection:

- Project operation is used to project only a certain set of attributes of a relation.
- That means if you want to see only the names all of the students in the student table, then you can use Project operation.
- Thus, to **display particular column** from the relation, the projection operator is used.
- It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.
- **Syntax:**

$$\Pi C1, C2\ldots (r)$$

- where C1, C2 etc. are attribute names (column names).
- **For example -** Consider the Student table given below
- **Query:** Display the name and age all the students
- This can be written in relational algebra as

$$\Pi_{sname,\ age}(Student)$$

- Above statement will show us only the Name and Age columns for all the rows of data in Student table.

sname	age
Ram	21
Shyam	18
Seeta	16
Geeta	23

(3) Cartesian product:

- This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.

Syntax : $A \times B$

- **For example:** Suppose there are two tables named Student and Reserve as follows

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

Query: Find the names of all the students who have reserved isbn = 005.

- To satisfy this query we need to extract data from two table. Hence the cartesian product operator is used as

$$(\sigma_{Student.sid = Reserve.sid \wedge Reserve.Isbn = 005}(Student \times Reserve)$$

- As an output we will get

sid	sname	age	sid	isbn	day
1	Ram	21	1	005	07-07-18
2	Shyam	18	2	005	03-03-18

- Note: that although the sid columns is same, it is repeated.

(4) Set operations:

- Various set operations are - union, intersection and set-difference.
- Let us understand each of these operations with the help of examples.

(i) Union:

- This operation is used to fetch data from two relations(tables) or temporary relation (result of another operation).
- For this operation to work, the relations(tables) specified should have **same number of attributes(columns) and same attribute domain.**
- Also, the **duplicate tuples are automatically eliminated** from the result.

Syntax : $A \cup B$

- where A and B are relations.
- **For example:** If there are two tables student and book as follows –

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

- **Query:** We want to display both the student name and book names from both the tables then

$$\Pi_{Sname}(Student) \cup \Pi_{bname}(Book)$$

(ii) Intersection:

- This operation is used to fetch data from both tables which is common in both the tables.

Syntax : $A \cap B$

- where A and B are relations.
- **Example –** Consider two tables – Student and Worker

Student

Name	Branch
AAA	ComputerSci
BBB	Mechanical
CCC	Civil
DDD	Electrical

Worker

Name	Salary
XXX	3000
AAA	2000
YYY	1500
DDD	2500

- **Query:** If we want to find out the names of the students who are working in a company then

$\Pi_{name}(Student) \cap \Pi_{name}(Worker)$

Name
AAA
DDD

(iii) Set-Difference:

- The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Syntax : **A – B**

- **For Example:** Consider two relations Full_Time_Employee and Part_Time_Employee, if we want to find out all the employee working for Fulltime, then the set difference operator is used –

$\Pi_{EmpName}(Full\_Time\_Employee) - \Pi_{EmpName}(Part\_Time\_Employee)$

(5) Join:

- The join operation is used to **combine information from two or more relations**.
- Formally join can be defined as a cross-product followed by selections and projections, joins arise much more frequently in practice than plain cross-products. The join operator is used as

- There are three types of joins used in relational algebra

i) Conditional join: This is an operation in which information from two tables is combined using **some condition** and this condition is specified along with the join operator.

$A \bowtie_c B = \sigma_c(A \times B)$

Thus ⋈ is defined to be a cross-product followed by a selection. Note that the condition c can refer to attributes of both A and B. The condition C can be specified using <,<=,>,<= or = operators.

- For example, consider two table student and reserve as follows –

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

- If we want the names of students with sid(Student) = sid(Reserve) and isbn = 005, then we can write it using Cartesian product as –

$$(\sigma_{((Student.sid = Reserve.sid) \wedge (Reserve.(Isbn) = 005))}(Student \times Reserve))$$

Here there are two conditions as

i) (Student.sid = Reserve.sid) and ii) (Reserve.isbn = 005) which are joined by ∧ operator.

Now we can use $\bowtie_c$ instead of above statement and write it as –

$$(Student \bowtie_{(Student.sid = Reserve.sid) \wedge (Reserve.(Isbn) = 005)} Reserve))$$

- The result will be –

sid	sname	age	isbn	day
1	Ram	21	005	07-07-18
2	Shyam	18	005	03-03-18

ii) Equijoin:

- This is a kind of join in which there is **equality condition** between **two attributes(columns) of relations(tables)**. For example - If there are two table Book and Reserve table and we want to find the book which is reserved by the student having isbn 005 and name of the book is 'DBMS', then:

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

$(\sigma_{bname} = \text{'DBMS'} (\text{Book} \bowtie_{(Book.isbn = Reserve.isbn)} \text{Reserve})$

Then we get

isbn	bname	Author	sid	day
005	DBMS	XYZ	1	07-07-18
005	DBMS	XYZ	2	03-03-18

iii) **Natural Join** : When there are **common columns** and we have to **equate** these **common columns** then we use natural join. The symbol for natural join is simply ⋈ without any condition. For example, consider two tables -

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

- Now if we want to list the books that are reserved, then that means we want to match **Books.isbn** with **Reserve.isbn**. Hence it will be simply

  Books ⋈ Reserve

(6) **Rename operation** : This operation is used to rename the output relation for any query operation which returns result like Select, Project etc. Or to simply rename a relation(table). The operator ρ(rho) is used for renaming.

Syntax : ρ(RelationNew, RelationOld)

- **For example:** If you want to create a relation Student_names with **sid** and **sname** from **Student**, it can be done using rename operator as:
  $\rho(\text{Student\_names}, (\Pi_{sid.sname}(\text{Student}))$

(7) Divide operation

- The division operator is used when we have to evaluate queries which contain the keyword **ALL**.
- It is denoted by A/B where A and B are instances of relation.
- **For example -** Find all the customers having accounts in all the branches. For that consider two tables - Customer and Account as

Customer

Name	Branch
A	Pune
B	Mumbai
A	Mumbai
C	Pune

Account

Branch
Pune
Mumbai

Now A/B will give us

Name
A

- Here, we check all the branches from Account table against all the names from Customer table.
- We can then find that only customer A has all the accounts in all the branches.

Formal Definition of Division Operation : The operation A/B is define as the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there is a tuple <x,y> in A.

Example 1.12.1

Consider following databases reserves (sid, bid, day) sailors (sid, sname, rating, age) boats (bid,bname,color)

(i) Find the names of sailors who have reserved boat number 103

(ii) Find the names of sailors who have reserved a red boat

(iii) Find the id of sailors with age over 20 who have not reserved red boat

(iv) Find the names of sailors who have reserved at least one boat

Solution:

(i) $(\Pi_{sname}((\sigma_{bid=103}$ Reserves) ⋈ Sailors)

(ii) $(\Pi_{sname}((\sigma_{color='red'}$ Boats) ⋈ Reserves ⋈ Sailors)

(iii) $(\Pi_{sid}((\sigma_{age>20}$ Sailors) $-\Pi_{sid}((\sigma_{color='red'}$ Boats) ⋈ Reserves)

(iv) $(\Pi_{sname}$(Sailors ⋈ Reserves)

Example 1.12.2 *Consider the following expressions, which use the result of a redational algebra operation as the input to another operation. For each expression explain in words what the expression does : a)* $\sigma_{year \geq 2009(takes)}$ ⋈ *Student b)* $\sigma_{year \geq 2009(takes}$ ⋈ *Student) c)* $\Pi_{ID, name, course\text{-}id}$ *(student ⋈ takes)*

Solution:

a) Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took.

b) Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took but the selection must be before join operation.

c) Display the ID, Name and Course_id of all the students who took any course in the university.

Example 1.12.3

Consider following relational database

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

loan (loan_number, branch_name, amount)

borrower (customer_name, loan_number)

account (account_number, branch_name, balance)

depositor (customer_name, account_number)

i) Find the names of all branches located in "Chennai".

ii) Find the names of all borrowers who have a loan in branch "ABC".

Solution:

i) $\Pi_{branch\_name}(\sigma_{branch\_city = 'Chennai'}(branch))$

ii) $\Pi_{customer\_name}(\sigma_{branch\_name = 'ABC'}(borrower \bowtie loan))$

Example 1.12.4

Consider the table

author (author_id, first_name, last_name)

author_pub(author_id, pub_id, author_position)

book (book_id, book_title, month, year, editor)

pub (pub_id, title, book_id)

(i) Give the relational algebra expression that returns names of all the authors that are book editors

(ii) Give the relational algebra expression that returns names of all the authors that are not book editors

(iii) Write a relational algebra expression that returns the names of all authors who have at least one publication in the database.

Solution:

i) $(\Pi_{first\_name,last\_name}(author_{\ author\_id\ =\ editor} \bowtie book)$

ii) $(\Pi_{first\_name,last\_name}((\Pi_{author\_id}(author) - \Pi_{editor}(book)) \times author)$

iii) $(\Pi_{first\_name,last\_name}(author \times author\_pub)$

Example 1.12.5

Consider the following schema:

Supplier(sid, sname,address)

Parts(pid, pname, color)

Cataloge(sid,pid,cost)

Write the relational algebraic queries for the following:

i) Find the sids of supplier who supply some red or some green parts

ii) Find the sids of supplier who supply every red or some green parts

iii) Find the pids of parts supplied by at least two different suppliers. (NOV/DEC 2023)

Solution:

i) $\rho(R1, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'}\ Parts) \bowtie Cataloge))$

$\rho(R2, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'}\ Parts) \bowtie Cataloge))$

$R1 \cup R2$

ii) $\rho(R1, \Pi_{sid,\ pid}Cataloge) / (\Pi_{pid}\ \sigma_{color='red'}\ Parts))$

$\rho(R2, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'\ Parts}) \bowtie Cataloge))$

$R1 \cup R2$

iii) $\rho(R1, Cataloge)$

$\rho(R2, Cataloge)$

$(\Pi_{R1.pid}\ \sigma_{R1.pid = R2.pid \wedge R1.sid!=R2.sid}\ (R1 \times R2)$

Example 1.12.6

Consider the relational database

employee (person-name, street, city)

works (person-name, company-name, salary)

company (company-name, city)

manager (person-name, manager-name)

where primary keys are underlined.

(a) Find the names of all employees who work for First Bank Corporation

(b) Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than 200,000 per annum.

(c) Find the names of all employees in this database who live in the same city as the company for which they work. (APR/MAY 2024)

Solution:

a) $\Pi_{person\text{-}name}(\sigma_{company\text{-}name = \text{"First Bank Corporation"}}(works))$

b) $\Pi_{person\text{-}name,\ street,\ city}(\sigma_{company\text{-}name = \text{"First Bank Corporation"} \wedge salary > 200000}\ (works \bowtie employee))$

c) $\Pi_{person\text{-}name}(works \bowtie employee \bowtie company))$

5. Explain about SQL fundamentals.

SQL Fundamentals

- Structure Query Language (SQL) is a database query language used for storing and managing data in Relational DBMS.
- Various parts of SQL are –

- **Data Definition Language (DDL):** It consists of a set of commands for defining relation schema, deleting relations, and modifying relation schemas.
- **Data Manipulation Language (DML):** It consists of set of SQL commands for inserting tuples into relational schema, deleting tuples from or modifying tuples in databases.
- **Integrity:** The SQL DDL includes commands for specifying integrity constraints. These constraints must be satisfied by the databases.
- **View definition:** The SQL DDL contains the commands for defining views for database.
- **Transaction control:** The SQL also includes the set of commands that indicate beginning and ending of the transactions.
- **Embedded SQL and Dynamic SQL:** There is a facility of including SQL commands in the programming languages like C, C++, COBOL or Java.
- **Authorization:** The SQL DDL includes the commands for specifying access rights to relations and views.

Data Abstraction

- The Basic data types used in SQL are –

(1) char(n):

- For representing the fixed length character string this data type is used.
- **For instance –** to represent name, designation, coursename, we use this data type.
- Instead of char, we can also use **character**.
- The n is specified by the user.

(2) varchar(n):

- The varchar means character varying.
- That means – for denoting the variable length character strings this data type is used.
- The n is user specified maximum character length.

(3) int:

- For representing the numeric values without precision, the int data type is used.

(4) numeric:

- For representing, a fixed-point number with user-specified precision this data type is used.

- The number consists of m digits plus sign k digits are to the right of precision.
- **For instance,** the numeric (3,2) allows 333.11 but it does not allow 3333.11

(5) smallint:

- It is used to store small integer value. It allows machine dependent subset of integer type.

(6) real:

- It allows the floating point, double precision numbers.

(7) float(n):

- For representing the floating-point number with precision of at least n digits this data type is used.

6.List and explain various DDL, DML, DCL commands in detail with examples. (NOV/DEC 2023)

Basic Schema Definition

- In this section, we will discuss various SQL commands for creating the schema definition.
- There are three types of SQL Languages -

1. DDL commands:

- DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.
- It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

Examples of DDL commands are:

- CREATE – is used to create the database or its objects such as table, function, views and so on.
- 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.
- COMMENT –is used to add comments to the data dictionary.
- RENAME –is used to rename an object existing in the database.

2. DML commands:

- DML stands for Data Manipulation Language. These commands deal with manipulation of data present in the database.

- **Examples of DML commands are:**
  1. **SELECT –** is used to retrieve data from the database.
  2. **INSERT –** is used to insert data into a table.
  3. **UPDATE –** is used to update existing data within a table.
  4. **DELETE –** is used to delete records from a database table.

3. DCL commands:

- It stands for Data Control Language.
- It includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
- **Examples of DCL commands:**
  1. GRANT-gives user's access privileges to database.
  2. REVOKE-withdraw user's access privileges given by using the GRANT command.
- Let us discuss various commonly used SQL commands that help in building the basic schema.

(1) Create Table: The database relation can be created by using the **create table** command.

Syntax

create table table_name;

Example

create table Student
(RollNo int,
Name varchar (10),
Marks numeric (3,2),
Primary key (RollNo));

The primary key attribute must be non-null and unique.

(2) Insert: The insert command is used to insert data into the table. There are two syntaxes of inserting data into SQL

Syntax

i) Insert into table_name (column1, column2, column3, ...)
values (value1, value2, value3, ...);
ii) insert into table_name
values (value1, value2, value3, ...);

Example

(i) insert into Student (RollNo,Name,Makrs) **values**(101,'AAA',56.45)
(ii) **insert into** Student values(101,'AAA',56.45)

(3) Delete: This command is used to delete the existing record.

Syntax

delete from table_name

where condition;

Example

Delete from student

where RollNo=10

(4) Alter: The **alter table** statement is used to add, delete, or modify columns in an existing table.

The **alter table** statement is also used to add and drop various constraints on an existing table.

Syntax for adding a columns

alter table table_name

add column_name datatype;

Example

Alter table student

Add address **varchar (20**

Syntax for dropping column

Alter table table_name

drop column column_name;

Example

Alter table student

drop column address;

Basic Structure of SQL Queries

The **basic form** of SQL queries is

SELECT-FROM-WHERE. The syntax is as follows:

SELECT [DISTINCT] *target-list*

FROM *relation-list*

WHERE *qualification*

- **SELECT:** This is one of the fundamental query commands of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.
- **FROM:** This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.

- **WHERE:** This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.
- **Relation-list:** A list of relation names(tables)
- **target-list:** A list of attributes of relations from relation list(tables)
- **qualification:** Comparisons of attributes with values or with other attributes combined using AND, OR and NOT.
- **DISTINCT** is an optional keyword indicating that the answer should not contain duplicates. Normally if we write the SQL without DISTINCT operator then it does not eliminate the duplicates.

Example

```
SELECT sname
FROM Student
WHERE age>18
```

- The above query will return names of all the students from student table where age of each student is greater than 18.

Queries on Multiple Relations

- Many times, it is required to access multiple relations(tables) to operate on some information.
- **For example,** consider two tables as **Student** and **Reserve**.

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

sid	isbn	day
1	005	07-07-18
2	007	03-03-18
3	009	

Query: Find the names of students who have reserved the books with book isbn

```
Select Student.sname,Reserve.isbn
From Student, Reserve
Where Student.sid=Reserve.sid
```

Use of SQL Join

- The SQL Joins clause is used to combine records from two or more tables in a database.
- A JOIN is a means for combining fields from two tables by using values common to each.

- **Example:** Consider two tables for using the joins in SQL. Note that **cid** is common column in following tables.

Student

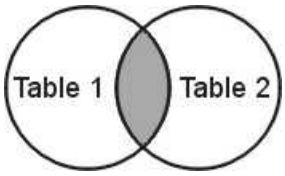
sid	cid	sname
1	101	Ram
2	101	Shyam
3	102	Seeta
4	NULL	Geeta

Reserve

cid	cname
101	Pune
102	Mumbai
103	Chennai

1) Inner Join:

- The most important and frequently used of the joins is the INNER JOIN. They are also known as an EQUIJOIN.
- The INNER JOIN creates a new result table by combining column values of two tables (Table1 and Table2) based upon the **join-predicate.**
- The query compares each row of table1 with each row of Table2 to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. It can be represented as :



Syntax:

- The basic syntax of the INNER JOIN is as follows.

```
SELECT Table1.column1, Table2.column2...
FROM Table1
INNER JOIN Table2
ON Table1.common_field = Table2.common_field;
```

Example:

- For above given two tables namely Student and City, we can apply inner join. It will return the record that are matching in both tables using the common column **cid**.
- The query will be

SELECT *

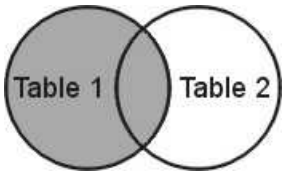
FROM Student **Inner Join** City on Student.cid=City.cid

- The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai

2) Left Join:

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table.
- This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- It can be represented as –



- **Syntax:** The basic syntax of a LEFT JOIN is as follows.

SELECT

SELECT Table1.column1, Table2.column2...

FROM Table1

LEFT JOIN Table2

ON Table1.common_field = Table2.common_field;

- **Example:** For above given two tables namely Student and City, we can apply Left join. It will Return all records from the left table, and the matched records from the right table using the common column **cid.** The query will be

SELECT *

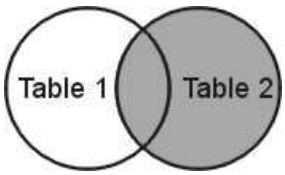
FROM Student **Left Join** City on Student.cid=City.cid

- The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL

3) Right Join:

- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.
- This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- It can be represented as follows:



- **Syntax:** The basic syntax of a RIGHT JOIN is as follow -

SELECT Table1.column1, Table2.column2...
FROM Table1
RIGHT JOIN Table2
ON Table1.common_field = Table2.common_field;

- **Example:** For above given two tables namely Student and City, we can apply Right join. It will return all records from the right table, and the matched records from the left table using the common column **cid.** The query will be

SELECT *

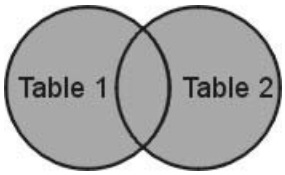
FROM Student **Right Join** City on Student.cid=City.cid

- The result will be –

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
NULL	NULL	NULL	103	Chennai

4) Full Join:

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.
- It can be represented as



- **Syntax:** The basic syntax of a FULL JOIN is as follows:

SELECT Table1.column1, Table2.column2...

FROM Table1 FULL JOIN Table2 ON Table1.common_field =Table2.common_field;

- The result will be -
- **Example:** For above given two tables namely **Student** and **City**, we can apply Full join. It will return returns rows when there is a match in one of the tables using the common column **cid.** The query will be -

SELECT *

FROM Student **Full Join** City on Student.cid=City.cid

- The result will be –

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL
NULL	NULL	NULL	103	Chennai

7. Explain the six clauses in the syntax of SQL query and show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional.

Additional Basic Operations

1. The Rename Operation:

- The SQL **AS** is used to assign temporarily a new name to a table column or table(relation) itself.
- One reason to rename a relation is to replace a long relation name with a shortened version that is more convenient to use elsewhere in the query. **For example –** "Find the names of students and isbn of book who reserve the books".

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	007	03-03-18
3	009	05-05-18

```
Select S.sname,R.isbn
From Student as S, Reserve as R
Where S.sid=R.sid
```

- In above case we could shorten the names of tables Student and Reserve as S and R respectively.
- Another reason to rename a relation is a case where we wish to compare tuples in the same relation. We then need to take the Cartesian product of a relation with itself.

- **For example –** If the query is – Find the names of students who reserve the book of isbn 005.
- Then the SQL statement will be –

Select S.sname,R.isbn

From Student as S, Reserve as R

Where S.sid=R.sid and S.isbn=005

2) Attribute Specification in Select clause:

- The symbol * is used in select clause to denote **all attributes.** For example – To select all the records from Student table we can write

Select* from Student

3) Ordering the display of tuples:

- For displaying the records in particular order, we use **order by** clause.
- The general syntax with ORDER BY is

SELECT column_name(s)

FROM table_name

WHERE condition

ORDER BY column_name(s)

- **Example:** Consider the Student table as follows –

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

Query: Find the names of students from highest marks to lowest

Select sname

From Student

Order By marks

- We can also use the **desc** for descending order and **asc** for ascending order.
- **For example:** In order to display names of the students in descending order of city – we can specify

Select sname

From Student

Order by city **desc**;

(4) Where clause Predicate:

- The **between** operator can be used to simplify the where clause which is used to denote the value be less than or equal to some value and greater than or equal to some other value. For example – of we want the names of the students whose marks are between 80 and 90 then SQL statement will be

Select name
From Students
Where marks between 80 and 90;

- Similarly, we can make use of the comparison operators for various attributes.
- **For example -** If the query is – Find the names of students who reserve the book of isbn 005.
- Then the SQL statement will be –

Select sname
From Student, Reserve
Where (Student.sid,Reserve.isbn)=(Reserve.sid,005);

- We can use AND, OR and NOT operators in the Where clause. For filtering the records based on more than one condition, the AND and OR operators can be used.
- The NOT operator is used to demonstrate when the condition is not TRUE.
- Consider following sample database – **Students** database, for applying AND, OR and NOT operators.

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

Syntax of AND

SELECT column1, column2, ...
FROM table_name
WHERE condition1 **AND** condition2 **AND** condition3 ...;

Example: Find the student having name “AAA” and lives in city “Pune”

SELECT *

FROM Students

Where sname='AAA' **AND** city='Pune'

Output

sid	sname	marks	city
1	AAA	60	Pune

Syntax OR

SELECT column1, column2, ...

FROM table_name

WHERE condition1 **OR** condition2 **OR** condition3 ...;

Example: Find the student having name "AAA" OR lives in city "Pune"

SELECT *

FROM Students

Where sname='AAA' **OR** city='Pune'

Output

sid	sname	marks	city
1	AAA	60	Pune
3	CCC	90	Pune

Syntax NOT

SELECT column1, column2, ...

FROM table_name

WHERE NOT condition

Example: Find the student who do not have city "Pune"

SELECT *

FROM Students

Where NOT city='Pune'

Output

sid	sname	marks	city
2	BBB	70	Mumbai
4	DDD	55	Mumbai

8. Explain about string operations in SQL with its various operations

String Operations

- For string comparisons, we can use the comparison operators =, <, >, <=,>=, <> with the ordering of strings determined alphabetically as usual.
- SQL also permits a variety of functions on character strings such as concatenation suing operator||, extracting substrings, finding length of string, converting strings to uppercase (using function **upper(s)**) and lowercase (using function **lower(s)**), removing spaces at the end of string (using function(trim(s)) and so on.
- Pattern matching can also be performed on strings using two types of special characters –
    1. Percent (%): It matches zero, one or multiple characters
    2. Underscore (_): The _ character matches any single character.
    3. The percentage and underscore can be used in combinations.
    4. Patterns are case sensitive. That means upper case characters do not match lowercase characters or vice versa.
- **For instance:**
    1. 'Data%' matches any string beginning with "Data", For instance it could be with "Database", "DataMining","DataStructure"
    2. '_ _ _' matches any string of exactly three characters.
    3. '_ _ _ %'matches any string of at least length 3 characters.
    4. The **LIKE** clause can be used in **WHERE** clause to search for specific patterns.
- **For example** – Consider following **Employee Database**

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
2	Mohsin	Manager	2-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan
6.	Archana	Purchase	6-Jan

(1) Find all the employee with EmpName starting with "s"

SQL Statement:

SELECT * FROM Employee

WHERE EmpName LIKE 's%'

Output

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan

(2) Find the names of employee whose name begin with S and end with a

SQL Statement:

SELECT EmpName FROM Employee

WHERE EmpName LIKE 'S%a'

Output

EmpName
Supriya
Sonia

(3) Find the names of employee whose name begin with S and followed by exactly four characters

SELECT EmpName FROM Employee

WHERE EmpName LIKE 'S_ _ _ _ '

Output

EmpName
Sunil
Sonia
Suraj

9. Explain Set Operations in SQL with examples.

Set Operations

1) UNION:

- To use this UNION clause, each SELECT statement must have

  i) The same number of columns selected

ii) The same number of column expressions

iii) The same data type and

iv) Have them in the same order

- This clause is used to combine two tables using UNION operator. It replaces the OR operator in the query. The union operator eliminates duplicate while the union all query will retain the duplicates.

Syntax

The basic syntax of a UNION clause is as follows –

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

UNION

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

- Here, the given condition could be any given expression based on your requirement.

Consider Following relations –

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

- **Example:** Find the names of the students who have reserved the 'DBMS' book or 'OS' Book
- The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
UNION
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

2) Intersect: The common entries between the two tables can be represented with the help of Intersect operator. It replaces the **AND** operator in the query.

Syntax

The basic syntax of a INTERSECT clause is as follows –

```
SELECT column1 [, column2]
FROM table1 [, table2]
[WHERE condition]
```

INTERSECT

```
SELECT column1 [, column2]
FROM table1 [, table2]
[WHERE condition]
```

Example: Find the students who have reserved both the 'DBMS' book and 'OS' Book The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
```

INTERSECT

SELECT S.sname

FROM Student S, Rese

rve R, Book B

WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'

3) Except: The EXCEPT clause is used to represent the set-difference in the query. This query is used to represent the entries that are present in one table and not in other.

Syntax:

The basic syntax of a EXCEPT clause is as follows –

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

EXCEPT

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

Example: Find the students who have reserved both the 'DBMS' book but not reserved 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

SELECT S.sid, S.sname

FROM Student S, Reserve R, Book B

WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'

EXCEPT

SELECT S.sname

FROM Student S, Reserve R, Book B

WHERE S.sid=R.sid AND R.isbn=B.isbn AND

B.bname='OS'

10. Explain aggregate functions in SQL with example.

Aggregate Functions

- An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.
- SQL offers five built-in aggregate functions:
    1. Average: avg

2. Minimum: min
3. Maximum: max
4. Total: sum
5. Count:

Basic Aggregation

- The aggregate functions that accept an expression parameter can be modified by the keywords DISTINCT or ALL. If neither is specified, the result is the same as if ALL were specified.

DISTINCT	Modifies the expression to include only distinct values that are not NULL
ALL	Includes all rows where expression is not NULL

- **Syntax of all the Aggregate Functions**

AVG ([DISTINCT | ALL] expression)

COUNT (*)

COUNT ([DISTINCT | ALL] expression)

MAX ([DISTINCT | ALL] expression)

MIN ([DISTINCT | ALL] expression)

SUM ([DISTINCT | ALL] expression)

- The **avg** function is used to compute average value. For example – To compute average marks of the students we can use

SQL Statement

SELECT AVG (marks)

FROM Students

- The **Count** function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates. **For example,** Consider following table

Test

id	value
11	100
22	200
33	300
NULL	400

SQL Statement

SELECT COUNT (*)

FROM Test

Output

4

SELECT **COUNT (ALL** id)

FROM Test

Output

3

- The **min** function is used to get the minimum value from the specified column.
- **For example** – Consider the above created **Test** table

SQL Statement

SELECT Min(value)

FROM Test

Output

100

- The **max** function is used to get the maximum value from the specified column.
- **For example –** Consider the above created **Test** table

SQL Statement

SELECT Max(value)

FROM Test

Output

400

- The **sum** function is used to get total sum value from the specified column. **For example –** Consider the above created **Test** table

SQL Statement

SELECT sum(value)

FROM Test

Output

1000

Use of Group By and Having Clause

(i) Group By:

- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement.
- Optionally it is used in conjunction with aggregate functions.
- The queries that contain the GROUP BY clause are called grouped queries
- This query returns a single row for every grouped item.
- **Syntax:**

SELECT column_name(s)

FROM table_name

WHERE condition

GROUP BY column_name(s)

The general syntax with ORDER BY is

SELECT column_name(s)

FROM table_name

WHERE condition

GROUP BY column_name(s)

ORDER BY column_name(s)

- **Example:** Consider the Student table as follows –

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

- **Query:** Find the total marks of each student in each city

SELECT SUM (marks), city

FROM Student

GROUP BY city

Output

SUM(marks)	city
150	Pune
125	Mumbai

(ii) Having Clause:

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.
- Syntax:

SELECT column-names
FROM table-name
WHERE condition
GROUP BY column-names
HAVING condition

- **Example:** Consider the Student table as follows –

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai
5	EEE	84	Chennai

Query: Find the total marks of each student in the city named 'Pune' and 'Mumbai' only

SELECT SUM (marks), city
FROM Student
GROUP BY city
HAVING city IN ('Pune', 'Mumbai')

Output

- The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

11. Write about Nested Queries in detail. Or Explain in brief about Subqueries and Co-related queries. (NOV/DEC 2023)

Nested Queries

In nested queries, a **query is written inside a query**. The result of inner query is used in execution of outer query.

There are two types of nested queries:

i) Independent Query:

- In independent nested queries, query execution starts from innermost query to outermost queries.
- The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.
- Various operators like IN, NOT IN, ANY, ALL etc are used in writing independent nested queries.
- For example - Consider three tables namely **Student, City and Student_City as follows –**

Student

sid	sname	phone
1	Ram	1111
2	Shyam	2222
3	Seeta	3333
4	Geeta	4444

City

cid	cname
101	Pune
102	Mumbai
103	Chennai

Student_City

sid	cid
1	101
1	103
2	101
3	102
4	102
4	103

- **Example 1 -** If we want to find out **sid** who live in city 'Pune' or 'Chennai'.

We can then write independent nested query using **IN** operator. Here we can use the IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Step 1: Find cid for cname='Pune' or 'Chennai'. The query will be

```
SELECT cid
FROM City
WHERE cname='Pune' or 'Chennai'
```

Step 2: Using cid obtained in step 1 we can find the sid. The query will be

```
SELECT sid
FROM Student_City
WHERE cid IN
(SELECT cid FROM City WHERE cname='Pune' or cname='Chennai')
```

The inner query will return a set with members 101 and 103 and outer query will return those **sid** for which **cid** is equal to any member of set (101 and 103 in this case). So, it will return 1, 2 and 4.

- **Example 2:** If we want to find out **sname** who live in city 'Pune' or 'Chennai'.

SELECT sname FROM Student WHERE sid IN
(SELECT sid FROM Student_City WHERE cid IN
(SELECT cid FROM City WHERE cname='Pune' or cname='Chennai'))

ii) Co-related Query:

In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query. For example,

If we want to find out sname of Student who live in city with cid as 101, it can be done with the help of co-related nested query as:

SELECT sname FROM Student S WHERE EXISTS
(SELECT * FROM Student_City SC WHERE S.sid=SC.sid and SC.cid=101)

Here For each row of **Student** S, it will find the rows from **Student_City** where S.sid = SC.sid and SC.cid=101.

If for a **sid** from **Student S,** atleast a row exists in **Student_City** SC with **cid**=101, then inner query will return true and corresponding **sid** will be returned as output.

Modification of Databases

- The modification of database is an operation for making changes in the existing databases. Various operations of modification of database are – insertion, deletion and updation of databases.

1. Deletion: The **delete** command is used to delete the existing record.

Syntax

delete from table_name

where condition;

Example

delete from student

where RollNo=10

2. Insertion: The insert command is used to insert data into the table. There are two syntaxes of inserting data into SQL

Syntax

(i) Insert into table_name (column1, column2, column3, ...)
values (value1, value2, value3, ...);

(ii) insert into table_name
values (value1, value2, value3, ...);

Example

(i) insert into Student (RollNo, Name, Marks) values(101,'AAA',56.45)

(ii) insert into Student values(101,'AAA',56.45)

3. Update: The update statement is used to modify the existing records in the table.

update table_name

set column1=value1, column2=value2, …

where condition;

Example:

Delete student

Set Name='WWW'

where RollNo=101

Example: Write the DDL, DML, DCL for the students database. Which contains student details: name, id, DOB, branch, DOJ.

Course details: Course name, Course id, Stud.id, Faculty name, id, marks

Solution:

DDL Commands

```
CREATE TABLE Student
(
stud_name varchar (20),
stud_id int (3),
DOB varchar (15),
branch varchar (10),
DOJ varchar (15),
);
CREATE TABLE Course
(course_name varchar (20),
course_id int (5),
stud_id int (3),
facult_name varchar (20),
faculty_id varchar (5),
marks real
);
```

DML Commands

The commands which we will use here are insert and select. The insert command is used to insert the values into database tables. Using the select command, the database values can be displayed.

(1) Inserting values into Student table

insert into Student (stud_name, stud_id, DOB, branch, DOJ)
values ('AAA',11,'01-10-1999', 'computers','5-3-2018')
insert into Student (stud_name, stud_id, DOB, branch, DOJ)
values ('BBB',12,'24-5-1988', 'Mechanical','17-2-2016')
insert into Student (stud_name, stud_id, DOB, branch, DOJ)
values ('CCC',13,'8-1-1990', 'Electrical','22-9-2017')

(2) Inserting values into Course table

insert into Course (course_name, course_id, stud_id, faculty_name, faculty_id, marks)
values ('Basic',101,11,'Archana','F001','50')
insert into Course (course_name, course_id, stud_id, faculty_name, faculty_id, marks)
values ('Intermediate',102,12,'Rupali','F002','70')
insert into Course (course_name, course_id, stud_id, faculty_name, faculty_id, marks)
values ('Advanced',103,13,'Sunil','F003','100')

(3) Displaying records of Student table

Select * **from** Student;

(4) Displaying records of Course table

Select * from Course;

DCL Commands

- The DCL command is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges.
- We will use the command **GRANT.**
- To allow a user to create tables in the database, we can use the below command,

Grant create table to user1;

Example: Write the following queries in relational algebra and SQL

(i) Find the names of employee who have borrowed a book published by McGraw Hill

(ii) Find the names of employees who have borrowed all books published by McGraw-Hill

Solution:

We will assume the databases as –

member (memb_no, name, dob)

books (isbn, title, authors, publisher)

borrowed (memb_no, isbn, date)

Relational Algebra:

(i)

$$\Pi_{name}((\sigma_{publisher='McGraw\ Hill'}\ books) \bowtie borrowed \bowtie member)$$

SQL:

```
SELECT name
FROM member
WHERE meber.memb_no=borrowed.memb_no
AND books.isbn=borrowed.isbn
AND books.publisher='McGraw Hill';
```

Relational Algebra:

(ii)

$$\rho\ (Tempname, (\pi_{memb\_no,isbn}\ borrowed) / \pi_{isbn}\ (\sigma_{publisher='McGraw\ Hill'}\ books)))$$

$$\pi_{name}(Tempname \bowtie member)$$

SQL:

```
SELECT distinct M.name
FROM Member M,
WHERE NOT EXIST
(
(SELECT isbn
FROM books
WHERE publisher = 'McGrawHill'
)
EXCEPT
(SELECT isbn
```

```
FROM borrowed R
WHERE R.memb_no = M.memb_no
)
)
```

Example: Assume the following table.

Degree (degcode, name, subject)

Candidate (seatno, degcode, name, semester, month, year, result)

Marks (seatno, degcode, semester, month, year, papcode, marks)

[degcode – degree code, name – name of the degree (Eg. MSc.), subject – subject of the course (Eg. Physis), papcode – paper code (Eg. A1)]

Solve the following queries using SQL;

Write a SELECT statement to display,

(i) all the degree codes which are there in the candidate table but not present in degree table in

the order of degcode.

(ii) the name of all the candidates who have got less than 40 marks in exactly 2 subjects.

(iii) the name, subject and number of candidates for all degrees in which there are less than 5

candidates.

(iv) the names of all the candidate who have got highest total marks in MSc. Maths.

Solution:

```
(i) SELECT C.degcode
FROM Candidate C,
WHERE NOT EXISTS
(SELECT D.degcode
FROM Degree D
WHERE D.degcode=C.degcode)
ORDER by C.degcode
(ii) SELECT C.name
FROM Candidate C, Degree D, Marks M
WHERE
C.seatno=M.seatno AND C.degcode=D.degcode AND C.degcode=M.degcode
AND M.marks<40
GROUP BY C.seatno
```

HAVING count(D.subject)=2;

(iii) SELECT D.name,D.subject,count(*)

FROM degree D, Candidate C

WHERE D.degcode=C.degcode

HAVING (SELECT count (*) FROM Candidate <5);

(iv)SELECT C.name

FROM Candidate C, Degree D, Marks M

WHERE

D.degname='MSc' AND D.subject='Maths' AND C.degcode=D.degcode AND

C.seatno=M.seatno AND

M.marks= (SELECT max(M.marks) FROM Marks M)

Example: Consider a student registration database comprising of the below given table schema.

Student File

Student Number	*Student Name*	*Address*	*Telephone*

Course File

Course Number	*Description*	*Hours*	*Professor Number*

Professor File

Professor Number	*Name*	*Office*

Registration File

Student Number	*Course Number*	*Date*

Consider a suitable sample of tuples / records for the above-mentioned tables and write DML statements (SQL) to answer for the queries listed below.

i) Which courses does a specific professor teach?

ii) What courses are taught by two specific professors?

iii) Who teaches a specific course and where is his/her office?

iv) For a specific student number, in which courses is the student registered and what is his/her name?

v) Who are the professors for a specific student?

vi) Who are the students registered in a specific course?

Solution:

(i)

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
HAVING count(DISTINCT P.name)=2
```

(ii)

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iii)

```
SELECT P.name,P.office, C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iv)

```
SELECT S.StudentNumber,S.StudentNumber,C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber AND
C.CourseNumber=R.CourseNumber
```

(v)

```
SELECT S.StudentName, P.Name
FROM Student S, Course C, Professor P, Registration R
WHERE C.ProfessorNumber=P.ProfessorNumber
AND C.CourseNumber=R.CourseNumber
AND S.StudentNumber=R.StudentNumber
GROUP BY P.ProfessorNumber
```

(vi)

```
SELECT S.StudentName, C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber
AND R.CourseNumber=C.CourseNumber
GROUP BY C.CourseNumber
```

12. Write in detail about Advanced SQL Features.

Advanced SQL Features

Embedded SQL

- The programing module in which the SQL Statements are embedded is called Embedded SQL module.
- It is possible to embed SQL statements inside the programming language such as C, C++, PASCAL, Java and so on.
- It allows the application languages to communicate with DB and get requested result.
- The high-level languages which support embedding SQLs within it are also known as **host language.**
- An embedded SQL program must be processed by a special preprocessor prior to compilation. The preprocessor replaces embedded SQL requests with host-language declarations and procedure calls that allow runtime execution of the database accesses. Then, the resulting program is compiled by the host-language compiler. This is the main distinction between embedded SQL and JDBC or ODBC.
- **Example of Embedded SQL –** Following program prompts the user for an order number, retrieves the customer number, salesperson, and status of the order, and displays the retrieved information on the screen.

```
int main () {
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
int OrderID; /* Employee ID (from user) */
int CustID; /* Retrieved customer ID */
char SalesPerson[10] /* Retrieved salesperson name */
char Status [6] /* Retrieved order status */
EXEC SQL END DECLARE SECTION;
/* Set up error processing */
EXEC SQL WHENEVER SQLERROR GOTO query_error;
EXEC SQL WHENEVER NOT FOUND GOTO bad_number;
/* Prompt the user for order number */
printf ("Enter order number: ");
scanf_s("%d", &OrderID);
/* Execute the SQL query */
EXEC SQL SELECT CustID, SalesPerson, Status
```

```
FROM Orders
WHERE OrderID =: OrderID
INTO: CustID, :SalesPerson, :Status;
/* Display the results */
printf ("Customer number: %d\n", CustID);
printf ("Salesperson: %s\n", SalesPerson);
printf ("Status: %s\n", Status);
exit ();
query_error:
printf ("SQL error: %ld\n", sqlca->sqlcode);
exit ();
bad_number:
printf ("Invalid order number.\n");
exit ();
}
```

Features of Embedded SQL

(1) It is easy to use.

(2) It is ANSI/ISO standard programming language.

(3) It requires less coding

(4) The pre-compiler can optimize execution time by generating stored procedures for the Embedded SQL statements.

(5) It is identical over different host languages, hence writing applications using different programming languages is quite easy.

Dynamic SQL

- Dynamic SQL is a programming technique which allows to build the SQL statements dynamically at runtime.
- Dynamic SQL statements are not embedded in the source program but stored as strings of characters that are manipulated during a program's runtime.
- These SQL statements are either entered by a programmer or automatically generated by the program.
- Dynamic SQL statements also may change from one execution to the next without manual intervention.
- Dynamic SQL facilitates automatic generation and manipulation of program modules for efficient automated repeating task preparation and performance.

- Dynamic SQL facilitates the development of powerful applications with the ability to create database objects for manipulation according to user input.
- The simplest way to execute a dynamic SQL statement is with an EXECUTE IMMEDIATE statement. This statement passes the SQL statement to the DBMS for compilation and execution.

Example: Consider the relation student (Reg.No., name, mark, and grade). Write embedded dynamic SQL program in C language to retrieve all the students' records whose mark is more than 90.

Solution:

```
int main () {
/* Begin program */
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION
int Reg_No;
char name [10][10];
float marks;
char grade;
EXEC SQL END DECLARE SECTION
EXEC SQL WHENEVER SQLERROR STOP
EXEC SQL SELECT Reg_No,name,marks,grade
FROM Student
WHERE marks>90
INTO: Reg_No,:name,:marks,:grade;
/* Display the results */
printf ("Registration number: %d\n", Reg_No);
printf ("Name: %s\n", name);
printf ("Marks: %f\n", marks);
printf ("Grade: %c\n", grade);
exit ();
EXEC SQL DISCONNECT
/* End program */
}
```

13. Write about Domain and Key Constraint in detail.

Domain and Key Constraint

Domain Constraint

- A domain is defined as the set of all unique values permitted for an attribute. For example, a domain of date is the set of all possible valid dates, a domain of Integer is all possible whole numbers, and a domain of day-of-week is Monday, Tuesday ... Sunday.
- This in effect is defining rules for a particular attribute. If it is determined that an attribute is a date then it should be implemented in the database to prevent invalid dates being entered.
- Domain constraints are user defined data type and we can define them like this:
- Domain constraint = Data type + Constraints
- The constraints can be specified using NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT.
- For example –

```
Create domain id_value integer
constraint id_test
check (value > 100); □ cheking if stud_id value is greater than 100
create table student (
stu_id id_value PRIMARY KEY,
stu_name CHAR (30),
stu_age integer
);
```

Key Constraint

- A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.
- For example - Consider the students relation and the constraint that no two students have the same student id. This IC is an example of a key constraint.
- The **definition** of key constraints contains **two parts -**
- Two distinct tuples in a legal instance (an instance that satisfies all Integrity Constraints including the key constraint) **cannot have identical values** in all the fields of a key.
- No subset of the set of fields in a key is a **unique identifier** for a tuple.
- The **first part** of the definition means that, in any legal instance, the values in the key fields uniquely identify a tuple in the instance. When specifying a key constraint, the DBA or user must be sure that this constraint will not prevent them from storing a 'correct' set of tuples. For example, several

students may have the same name, although each student has a unique student id. If the name field is declared to be a key, the DBMS will not allow the Students relation to contain two
tuples describing different students with the same name.

- The **second part** of the definition means, for example, that the set of fields {RollNo, Name} is not a key for Students, because this set properly contains the key {RollNo}. The set {RollNo, Name} is an example of a superkey, which is a set of fields that contains a key.
- The key constraint can be specified using SQL as follows -
- In SQL, we can declare that a subset of the columns of a table constitute a key by using the UNIQUE constraint.
- At most one of these candidate keys can be declared to be a primary key, using the PRIMARY KEY constraint. For example -
  CREATE TABLE Student (RollNo integer,
  Name CHAR (20),
  age integer,
  UNIQUE(Name,age),
  CONSTRAINT StudentKey PRIMARY KEY(RollNo))
- This definition says that **RollNo** is a **Primary key** and Combination of **Name and age is also a key.**

14. Explain the concept of view along with its operations.

- Views in SQL are kind of virtual tables.
- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.
- A view can either have all the rows of a table or specific rows based on certain condition.

1.CREATING VIEW

We can create a view using CREATE VIEW statement. The Syntax is

CREATE VIEW name_of_view AS
SELECT coloumn1, coloumn1, ...
FROMtable_name1, table_name2, ...
WHERE condition;

Example

Creating a view using single table:

Consider table Employee and create a view

EmployeeDetails whose Salary is<10000

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	poonam	8000
104	Sharda	15000
105	Monika	7000

Salary Table

```
CREATE VIEW EmployeeDetails(EmpID,EName)AS
SELECT E.EmpID, E.EName
FROM Employee E
WHERE E.Salary > 10000
```

The Output will be-

EmpID	EName
102	Madhura
103	Poonam
105	Monika

Creating a view from multiple table: In this example we will create a view from two tables Employee and Department

Employee table

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	poonam	8000
104	Sharda	15000
105	Monika	7000

Department table

EmpID	EName
101	Accounts
104	Sales
105	Sales

Now we need to create a view named Employee_dept_Details in which the name and salary of employees belonging to sales department is displayed. The SQL statements will then be-

CREATE VIEW Employee_dept_Details(EName,Salary) ASSELECT E.EName,E.SalaryRROM Employee E, Department DWHERE E.EmplD=D.EmpID AND D.DName='Sales'

The output will be

EName	Salary
Sharda	15000
Monika	7000

2.View Update

- The SQL UPDATE VIEW command can be used to modify the data of a view.
- All views are not updatable. So, UPDATE Command is not applicable to all views.
- An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.
- **Syntax for updating view**
- UPDATE < view_name >SET<Column1>=<value1>, <column2>=<value2>, WHERE <Condition>;
- **Example**

UPDATE VIEW Employee_dept _Details

SET Salary=1000WHERE EName=Monika';

This view can be viewed by using following query

SELECT * FROM Employee dept Details;

Rules for updating the views

The view can be updating the view

(1) The view can be defined based on one and only one table.

(2) The SELECT statement should not have DISTINCT keyword.

(3) The View should not have all NOT NULL values.

(4) The view should not be created from nested and complex queries.

(5) The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.

(6) The view should not have any field made out of aggregate functions.

(7) Any selected output fields of the view must not use constants, strings or value expressions.

3.Dropping View: For deleting a view, the DROP command is used.

Syntax: DROP VIEW view _name;

Example: DROP VIEW Employee _dept _Details;

15. Write about Transaction Control Statements in SQL.

- The COMMIT, ROLLBACK and SAVEPOINT are collectively considered as Transaction commands

1. COMMIT

The COMMIT command is used to save permanently any transaction to database. When we perform, Read or Write Operations to the database then

those changes scan be undone by rollback operations. To make these changes permanent, we should make use of commit.

2. ROLLBACK

The ROLLBACK command is used to undo transactions that have not already saved to database. **For Example-** Consider the database table as

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Student table

Following command will delete the record from the database, but if weimmediately perform ROLLBACK, then the deletion is undone.

For instance-

DELETE FROM student

WHERE Rollno = 2;

ROLLBACK

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

3.SAVEPOINT

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. The SAVEPOINT can be created as

SAVEPOINT_savepoint_name;

Then we can ROLLBACK to SAVEPOINT as

For example-Consider Student table as follows-

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Consider the following commands

SQL> SAVEPOINT S1

SQL>DELETE FROM student

Where Roll=2;

SQL>SAVEPOINT S2

SQL>DELETE FROM Student

Where RollNo=3;

SQL>SAVEPOINT S3

SQL>DELETE FROM Student

Where RollNo=4

SQL>DELETE FROM student

Where RollNo=5

SQL>ROLLBACK TO S3;

The resultant table will be: -

RollNo	Name
1	AAA
2	BBB
3	CCC

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai,

Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / III SEM

UNIT III – RELATIONAL DATABASE DESIGN AND NORMALIZATION

SYLLABUS:

ER and EER-to-Relational mapping – Update anomalies – Functional dependencies – Inference rules – Minimal cover – Properties of relational decomposition – Normalization (upto BCNF).

PART A

1. **What are the problems caused by redundancy?**
   1. **Problems caused by Redundancy:** Following problems can be caused by redundancy -

      i. Redundant Storage: Some information is stored repeatedly.

      ii. Update Anomalies: If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

      iii. Insertion Anomalies: Due to insertion of new record repeated information get added to the relation schema.

      iv. Deletion Anomalies: Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

2. **Define functional dependency.**
   2. Let P and Q be sets of columns, then: P functionally determines Q written P->Q if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q
   3. In other words, the functional dependency holds if

      TI. P = T2.P, then TI. Q = T2Q

- Where notation T1.P projects the tuple TI onto the attribute in P.

PREPARED BY: Mrs. S. JAYABHARATHI AP/AI&DS

3. Why certain functional dependencies are called trivial functional dependencies?

- A functional dependency FD:X->Y is called trivial if Y is a subset of X. This kind of dependency is called trivial because it can be derived from common sense. If one "side" is a subset of the other, it's considered trivial. The left side is considered the determinant and the right the dependent.
- **For example** (A, B) ->B is a trivial functional dependency because B is a subset of AB Since (A, B) ->B includes B, the value of 8 can be determined. It's a trivial functional dependency because determining B is satisfied by its relationship to A, B.

4. Define inference rules (Armstrong's Axioms).

- The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F+.
- The closure set of all functional dependency can be computed using basic three rules which are also called as Armstrong's Axioms.

These are as follows -

i) **Reflexivity** : If $X \supseteq Y$, then $X \rightarrow Y$

ii) **Augmentation** : If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

iii) **Transitivity** : If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union** : If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- **Decomposition** : If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

5. Define Minimal cover.

- A minimal cover for a set F of FDs is a set G of FDs such that:

  1) Every dependency in G is of the form X->A, where A is a single attribute.

  2) The **closure F+** is equal to the **closure G+**.

  3) If we obtain a set H of dependencies from G by deleting one or more dependencies or by deleting attributes from a dependency in G, then F+ H+.

6. Define normalization or What is meant by Normalization of data? (NOV/DEC 2023)

- Normalization is the process of reorganizing data in a database so that it meets **two basic requirements**:

  1) There is **no redundancy** of data (all data is stored in only one place).

  2) **Data dependencies** are logical (all related data items are stored together).

7. State anomalies of INF.

- All the insertion, deletion and update anomalies are in INF relation.

First Normal Form

- The table is said to be in INF if it follows following rules-
  1. It should only have single (atomic) valued attributes/columns.
  2. Values stored in a column should be of the same domain
  3. All the columns in a table should have unique names.
  4. And the order in which data is stored, does not matter.

8. Write about Prime and Non-Prime Attributes

- **Prime attribute:** An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute:** An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.
- **Example:** Consider a Relation R- (A, B, C, D) and candidate key as AB.
  1. **Prime attributes:** A, B
  2. **Non-Prime attributes:** C, D

9. Define the Second Normal Form

- For a table to be in the Second Normal Form, following conditions must be followed
  1. It should be in the First Normal form.
  2. It should not have partial functional dependency.

10. Give the Concept of Transitive Dependency OR Define Trivial functional Dependency (APR/MAY 2024)

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.

- **For example,** X->Z is a transitive dependency if the following functional dependencies hold true:
  1. X->Y
  2. Y->Z

11. Write about Third Normal Form

- A table is said to be in the Third Normal Form when,
  1. It is in the Second Normal form. (i.e. it does not have partial functional dependency)
  2. It doesn't have transitive dependency.
- In other words, 3NF can be defined as: A table is in 3NF if it is in 2NF and for each functional dependency

  X-> Y
- At least one of the following conditions hold:
  1. X is a super key of table
  2. Y is a prime attribute of table

12. Define Boyce/Codd Normal Form (BCNF) or Write a note on BCNF Normal Form. (APR/MAY 2023)

- 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.
- For a table to be in BCNF, following conditions must be satisfied:
  1. R must be in 3rd Normal Form
  2. For each functional dependency (X ~ Y), X should be a super Key.
- In simple words if Y is a prime attribute, then X cannot be non-prime.

13. Show that if a relation is in BCNF, then it is also in 3NF.

- Boyce and Codd Normal Form is a higher version of the Third Normal form.
- A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

- When the table is in BCNF then it doesn't have partial functional dependency as well as transitive dependency. Hence, it is true that if relation is in BCNF then it is also in 3NF.

14. What is decomposition? Or List the desirable properties of Relational Decomposition. (NOV/DEC 2023)

- Decomposition is the process of breaking down one table into multiple tables.
- **Formal definition of decomposition is:**
  - A decomposition of relation schema R consists of replacing the relation schema by two relation schemas that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.

15. Why it is necessary to decompose a relation?

- Decomposition is the process of breaking down one table into multiple tables.
- The decomposition is used for eliminating redundancy.

16. Explain at least two desirable properties of decomposition.

- There are two properties associated with decomposition and those are

  1) Loss-less Join or non-Loss Decomposition: When all information found in the original database is preserved after decomposition, we call it as loss less or non-loss decomposition.

  2) Dependency Preservation: This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

17. What are the primary goals of relational database design?

- There are two primary goals of relational database design –
  1. To generate a set of relation schemas that allow us to store information without unnecessary redundancy.
  2. To allows us to retrieve information easily.

18. What are the differences between 3NF and BCNF?

3NF	**BCNF**
3NF stands for Third Normal Form.	CNF stands for Boyce Codd Normal Form.
The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold: (i) X is a super key, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved BCNF.
Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to obtain in BCNF.
3NF can be achieved without losing any information from the old table.	For obtaining BCNF we may lose some information from old table.

19. Define the concept of Extraneous Attributes.

- An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies.
- The formal definition of extraneous attributes is as follows:
- Consider a set F of functional dependencies and the functional dependency α-> βin F
  1. Attribute A is extraneous in α if A € α, and F logically implies
     (F – {α->β}) ∪ {(α –> A) ->β
  2. Attribute A is extraneous in β if A € β and the set of functional dependencies (F – {α->β}) ∪ {(α-> (β– A)} logically implies F.

20.List the steps involved in Query Processing. (APR/MAY 2024)

- Parsing and translation
- Optimization
- Evaluation

21. Write an efficient relational algebraic expression for the following query: SELECT B1. BANKNAME FROM BANK AS B1, B2 WHERE B1. ASSETS> B2. ASSETS AND B2. BANKLOCATION='TAMILNADU'. (APR/MAY 2024)

Solution:

PART B

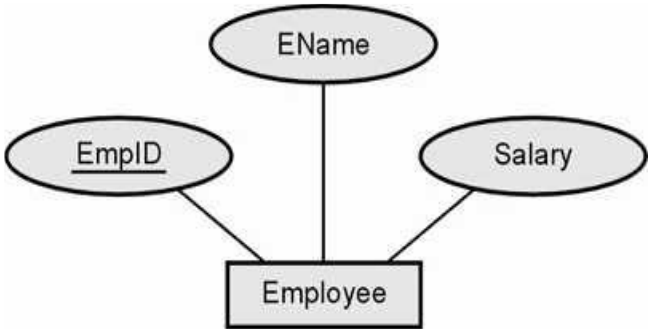
1. **Discuss the corresponding between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER model. (NOV/DEC 2022)**

ER to Relational Mapping

- In this section we will discuss how to map various ER model constructs to Relational Model construct.

Mapping of Entity Set to Relationship

- An entity set is mapped to a relation in a straightforward way.
- Each attribute of entity set becomes an attribute of the table.
- The primary key attribute of entity set becomes an entity of the table.
- **For example -** Consider following ER diagram as shown in figureure 3.1.



Figureure 3.1 ER Diagram for Employee

- The converted employee table is as follows –

EmpID	EName	Salary
201	Poonam	30000
202	Ashwini	35000
203	Sharda	40000

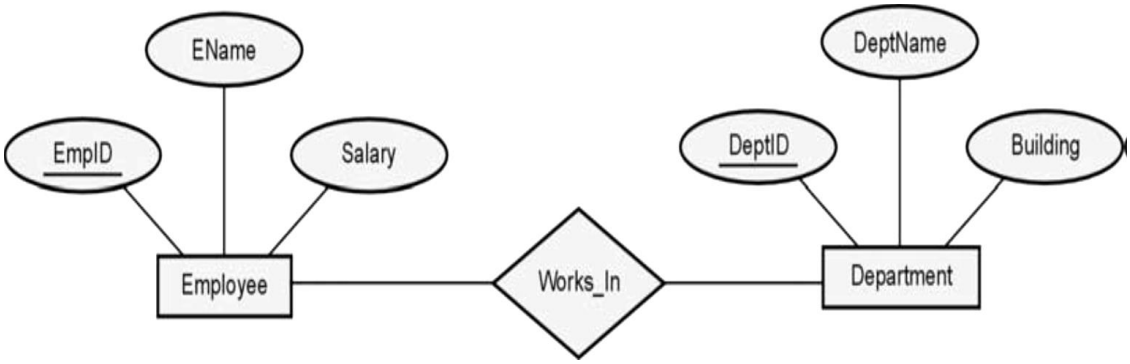
- The SQL statement captures the information for above ER diagram as follows:

```
CREATE TABLE Employee (EmpID CHAR (11),
EName CHAR (30),
Salary INTEGER,
PRIMARY KEY(EmpID))
```

Mapping Relationship Sets (without Constraints) to Tables

- Create a table for the relationship set.
- Add all primary keys of the participating entity sets as fields of the table.

- Add a field for each attribute of the relationship.
- Declare a primary key using all key fields from the entity sets.
- Declare foreign key constraints for all these fields from the entity sets.
- **For example -** Consider following ER model as shown in figureure 3.2.



Figureure 3.2 Mapping Relationship Sets (without Constraints) to Tables

- The SQL statement captures the information for relationship present in above ER diagram as follows -

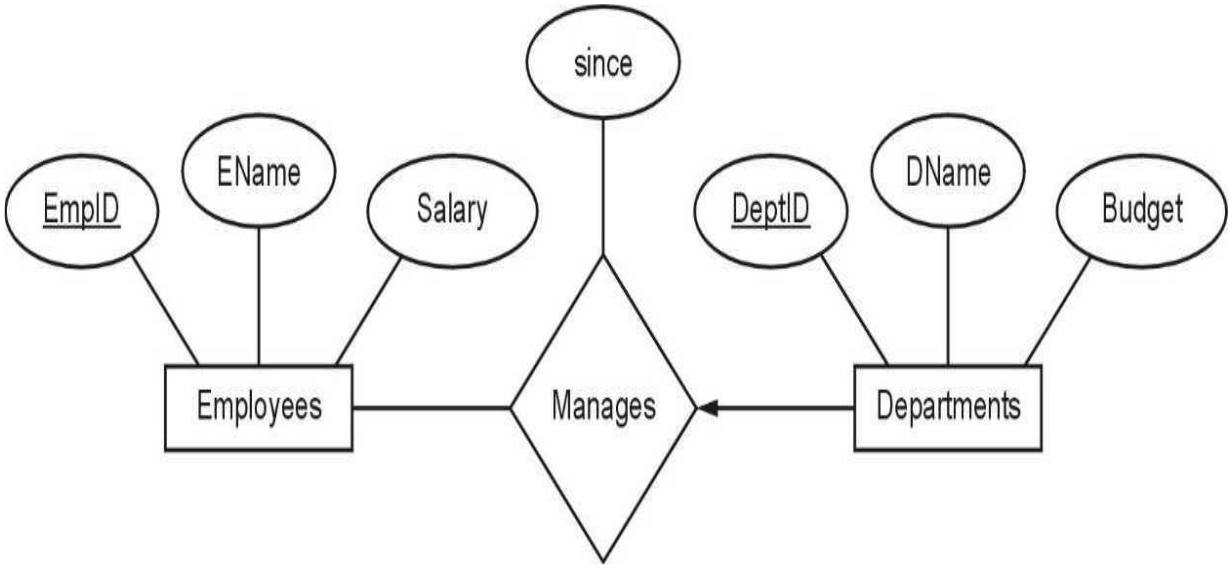
```
CREATE TABLE Works_In (EmpID CHAR (11),
DeptID CHAR (11),
EName CHAR (30),
Salary INTEGER,
DeptName CHAR (20),
Building CHAR (10),
PRIMARY KEY (EmpID, DeptID),
FOREIGN KEY (EmpID) REFERENCES Employee,
FOREIGN KEY (DeptID) REFERENCES Department
)
```

Mapping Relationship Sets (With Constraints) to Tables

- If a relationship set involves **n** entity sets and some **m** of them are linked **via arrows** in the ER diagram, the key for anyone of these **m** entity sets **constitutes a key** for the relation to which the relationship set is mapped.
- Hence, we have **m** candidate keys, and one of these should be designated as the **primary key.**
- There are two approaches used to convert a relationship sets with key constraints into table.

Approach 1:

- By this approach the relationship associated with more than one entities is separately represented using a table. For example - Consider following ER diagram as shown in figureure 3.3. Each **Dept has at most one manager, according to the key constraint on Manages.**



Figureure 3.3 Mapping Relationship Sets (With Constraints) to Tables

- Here the constraint is each department has at the most one manager to manage it.
- Hence, no two tuples can have same DeptID. Hence there can be a separate table named **Manages** with DeptID as Primary Key. The table can be defined using following SQL statement

  CREATE TABLE Manages (EmpID CHAR (11),
  DeptID INTEGER,
  Since DATE,
  PRIMARY KEY(DeptID),
  FOREIGN KEY (EmpID) REFERENCES Employees,
  FOREIGN KEY (DeptID) REFERENCES Departments)

Approach 2:

- In this approach, it is preferred **to translate a relationship set with key constraints.**
- It is a superior approach because, it avoids creating a distinct table for the relationship set.

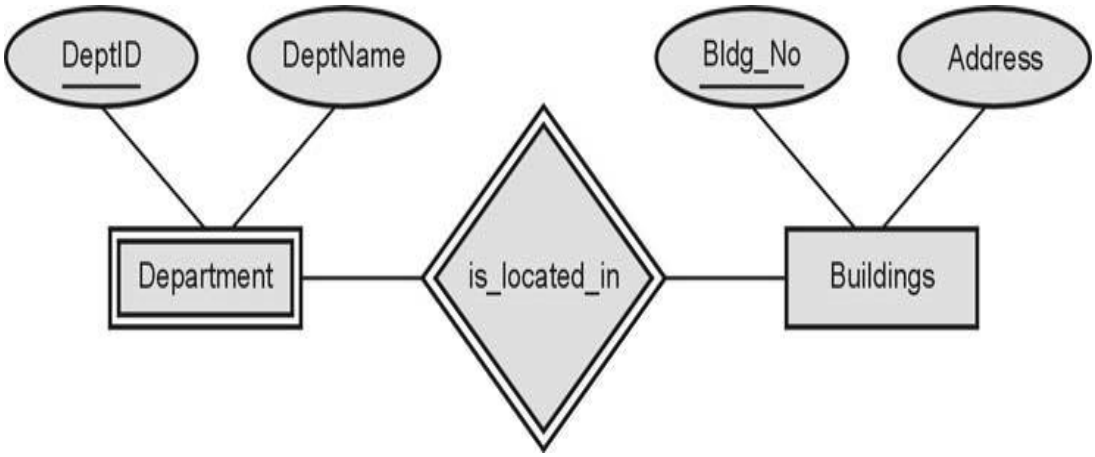
- The idea is to include the information about the relationship set in the table corresponding to the entity set with the key, taking advantage of the key constraint.
- This approach eliminates the need for a separate Manages relation, and queries asking for a department's manager can be answered without combining information from two relations.
- The only drawback to this approach is that space could be wasted if several departments have no managers.
- The following SQL statement, defining a **Dep_Mgr** relation that captures the information in both **Departments** and **Manages,** illustrates the second approach to translating relationship sets with key constraints:

```
CREATE TABLE Dep_Mgr (DeptID INTEGER,
DName CHAR (20),
Budget REAL,
EmpID CHAR (11),
since DATE,
PRIMARY KEY (DeptID),
FOREIGN KEY (EmpID) REFERENCES Employees)
```

Mapping Weak Entity Sets to Relational Mapping

- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity. Following steps are used for mapping Weka Entity Set to Relational Mapping
  1. Create a table for the weak entity set.
  2. Make each attribute of the weak entity set a field of the table.
  3. Add fields for the primary key attributes of the identifying owner.
  4. Declare a foreign key constraint on these identifying owner fields.
  5. Instruct the system to automatically delete any tuples in the table for which there are no owners.

- **For example -** Consider following ER model as shown in figureure 3.4.



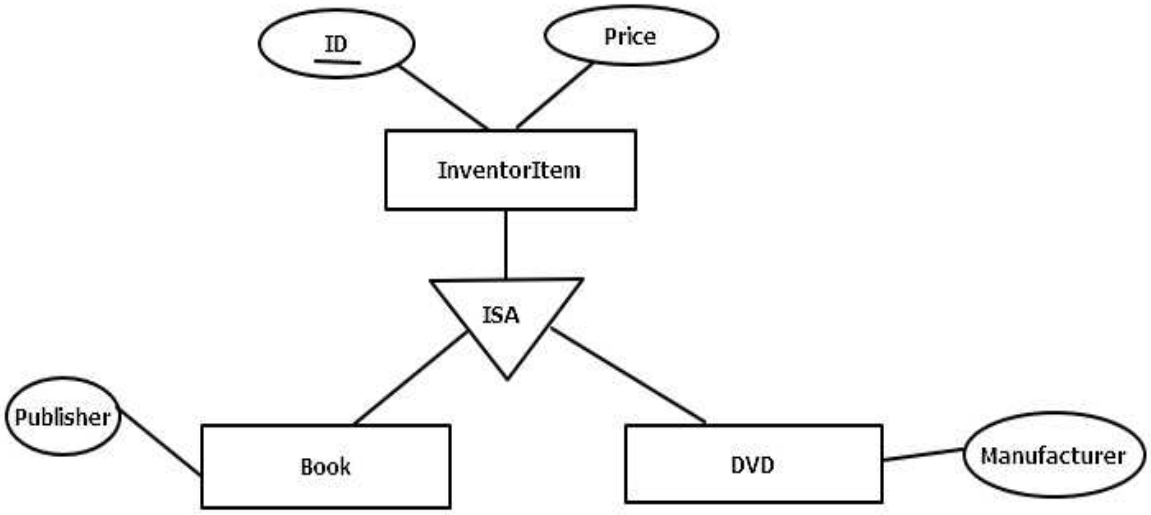
Figureure 3.4 Mapping Weak Entity Sets to Relational Mapping

- Following SQL Statement illustrates this mapping

```
CREATE TABLE Department (DeptID CHAR (11),
DeptName CHAR (20),
Bldg_No CHAR (5),
PRIMARY KEY (DeptID,Bldg_No),
FOREIGN KEY(Bldg_No) References Buildings on delete cascade
        )
```

Mapping of Specialization / Generalization (EER Construct) to Relational Mapping

- The Specialization/Generalization relationship (Enhanced ER Construct) can be mapped to database tables(relations) using three methods as shown in figureure 3.5. To demonstrate the methods, we will take the – InventoryItem, Book, DVD



Figureure 3.5 Mapping of Specialization / Generalization (EER Construct) to Relational Mapping

Method 1: All the entities in the relationship are mapped to individual tables
InventoryItem (ID, name)
Book (ID, Publisher)
DVD (ID, Manufacturer)
Method 2: Only subclasses are mapped to tables. The attributes in the superclass are duplicated in all subclasses. **For example -**
Book (ID, name, Publisher)
DVD (ID, name, Manufacturer)
Method 3: Only the superclass is mapped to a table. The attributes in the subclasses are taken to the superclass. **For example -**
InventoryItem (ID, name, Publisher, Manufacturer)

- This method will introduce **null** values. When we insert a **Book** record in the table, the **Manufacturer** column value will be null. In the same way, when we insert a **DVD** record in the table, the **Publisher** value will be null.

2. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. Also construct appropriate tables for the ER diagram you have drawn.

Solution:

Relational Mapping

patients (P_id, name, insurance, date-admitted, date-checked-out)

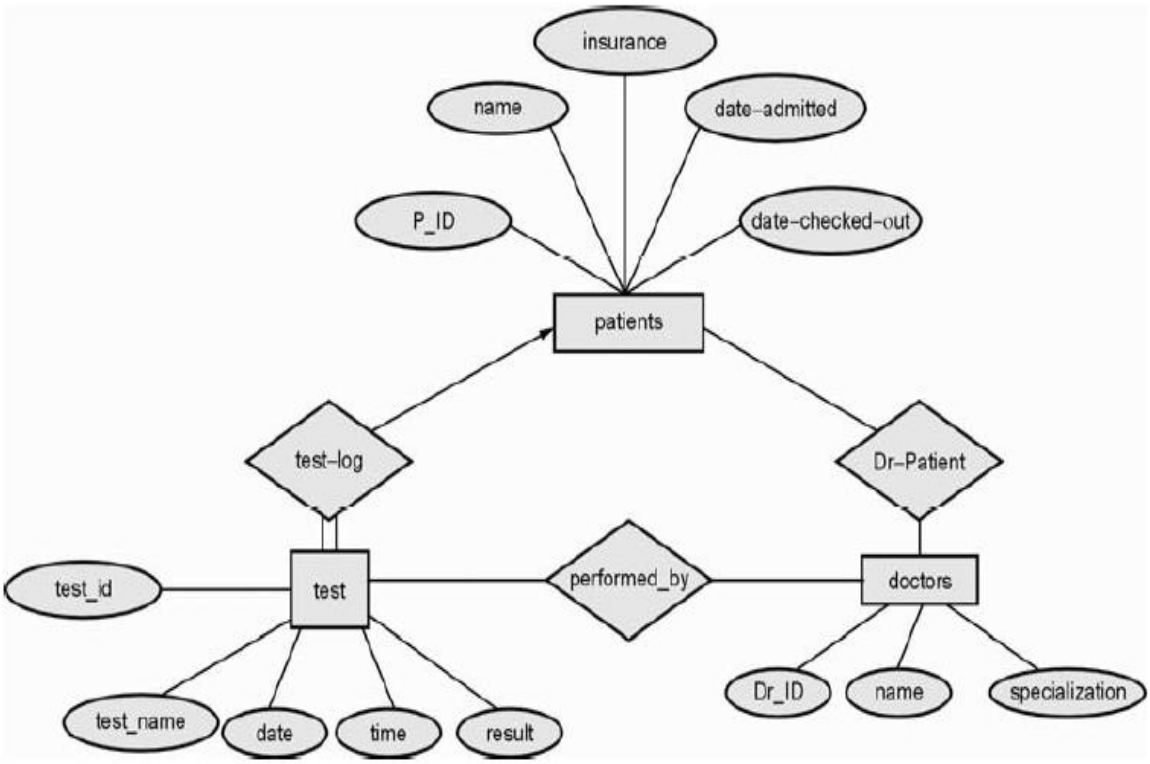
doctors (Dr_id, name, specialization)

test (testid, testname, date, time, result)

doctor-patient (P_id, Dr_id)

test-log (testid, P_id) performed-by (testid, Dr_id)

ER Diagram -



Figureure 3.6 E-R diagram for a hospital

3. Discuss about Functional Dependencies with an example. or Illustrate in detail about Functional Dependencies with relevant examples. (APR/MAY 2023) OR Explain in detail about closure set of Functional dependencies and closure set of attributes. (NOV/DEC 2023)

Functional Dependencies

Definition:

- Let P and Q be sets of columns, then: P functionally determines Q, written P → Q if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.
- In other words, the functional dependency holds if

  T1.P = T2.P, then T1. Q=T2.Q
- Where notation T1.P projects the tuple T1 onto the attribute in P.
- **For example:** Consider a relation in which the roll of the student and his/her name is stored as follows:

R	**N**
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Table which holds functional dependency i.e. R->N

- Here, R->N is true. That means the functional dependency holds true here. Because for every assigned RollNuumber of student there will be unique name.
- **For instance:** The name of the Student whose RollNo is 1 is AAA. But if we get two different names for the same roll number then that means the table does not hold the functional dependency.
- Following is such table –

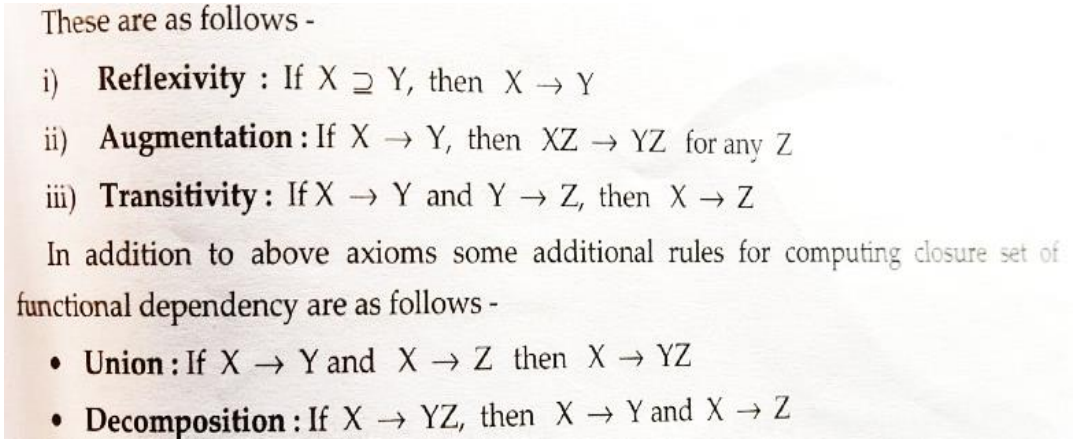
R	N
1	AAA
2	BBB
3	CCC
1	XXX
2	YYY

Table which does not hold functional dependency

- In above table for RollNumber 1 we are getting two different names - “AAA” and “XXX”. Hence here it does not hold the functional dependency.

Computing Closure Set of Functional Dependency

- The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F+.
- The closure set of functional dependency can be computed using basic three rules which are also called as Armstrong's Axioms.



These are as follows -

i) **Reflexivity** : If $X \supseteq Y$, then $X \rightarrow Y$

ii) **Augmentation** : If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

iii) **Transitivity** : If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union** : If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- **Decomposition** : If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

4.Compute the closure of the following set of functional dependencies for a relation scheme R (A, B, C, D, E), F={A->BC, CD->E, B->D, E->A) (APR/MAY 2024)

Solution: Consider F as follows

A->BC

CD->E

B->D

E->A

- The closure can be written for each attribute of relation as follows
- (A)+ = **Step 1:** {A} -> the attribute itself

  Step 2: {ABC} as A->BC

  Step 3: {ABCD} as B->D

  Step 4: {ABCDE} as CD->E

  Step 5: {ABCDE} as E->A and A is already present
- Hence (A)+ ={ABCDE}
- (B)+ = **Step 1:** {B}

  Step 2: {BD} as B->D

  Step 3: {BD} as there is no BD pair on LHS of F
- Hence (B)+ ={BD}
- (C)+ = **Step 1:** {C}

  Step 2: {C} as there is no single C on LHS of F
- Hence (C)+ ={C}

- (D)+ = **Step 1:** {D}

  Step 2: {D} as there is no BD pair on LHS of F
- Hence (D)+ ={D}
- (E)+ = **Step 1:** {E}

  Step 2: {EA} as E->A

  Step 3: {EABC} as A->BC

  Step 4: {EABCD} as B->D

  Step 5: {EABCD} as CD->E and E is already present
- By rearranging we get {ABCDE}
- Hence (E)+ ={ABCDE}
- (CD)+ = **Step 1:** {CD}

  Step 2: {CDE}

  Step 3: {CDEA}

  Step 4: {CDEAB}
- By rearranging we get {ABCDE}
- Hence (CD)+ ={ABCDE}

5.Compute the closure of the following set of functional dependencies for a relation scheme R (A, B, C, D, E), F={A->BC, CD->E, B->D, E->A) and find the candidate key.

Solution: For finding the closure of functional dependencies –

- Consider F as follows

  A->BC

  CD->E

  B->D

  E->A
- The closure can be written for each attribute of relation as follows
- (A)+ = **Step 1:** {A} -> the attribute itself

  Step 2: {ABC} as A->BC

  Step 3: {ABCD} as B->D

  Step 4: {ABCDE} as CD->E

  Step 5: {ABCDE} as E->A and A is already present
- Hence (A)+ ={ABCDE}
- (B)+ = **Step 1:** {B}

  Step 2: {BD} as B->D

Step 3: {BD} as there is no BD pair on LHS of F

- Hence (B)+ ={BD}
- (C)+ = **Step 1:** {C}

  Step 2: {C} as there is no single C on LHS of F
- Hence (C)+ ={C}
- (D)+ = **Step 1:** {D}

  Step 2: {D} as there is no BD pair on LHS of F
- Hence (D)+ ={D}
- (E)+ = **Step 1:** {E}

  Step 2: {EA} as E->A

  Step 3: {EABC} as A->BC

  Step 4: {EABCD} as B->D

  Step 5: {EABCD} as CD->E and E is already present
- By rearranging we get {ABCDE}
- Hence (E)+ ={ABCDE}
- (CD)+ = **Step 1:** {CD}

  Step 2: {CDE}

  Step 3: {CDEA}

  Step 4: {CDEAB}
- By rearranging we get {ABCDE}
- Hence (CD)+ ={ABCDE}
- We can identify candidate from the given relation schema with the help of functional dependency. For that purpose, we need to compute the closure set of attributes. Now we will find out the closure set which can completely identify the relation R (A, B, C, D).

  Let, (A)+ = {ABCDE}

  (B)+ = {BD}

  (C)+ = {C}

  (D)+ = {D}

  (E)+ = {ABCDE}

  (CD)+ = {ABCDE}
- Clearly, only (A)+, (E)+ and (CD)+ gives us {ABCD} i.e. complete relation R. Hence, these are the candidate keys.

6.Discuss the concept of Canonical Cover or Minimal Cover.

Canonical Cover or Minimal Cover

Formal Definition: A minimal cover for a set F of FDs is a set G of FDs such that:

1) Every dependency in G is of the form X->A, where A is a single attribute.

2) The **closure F+** is equal to the **closure G+**.

3) If we obtain a set H of dependencies from G by deleting one or more dependencies or by deleting attributes from a dependency in G, then F+ H+.

Concept of Extraneous Attributes

Definition: An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies. The formal definition of extraneous attributes is as follows:

- Consider a set F of functional dependencies and the functional dependency α-> βin F
  - Attribute A is extraneous in α if A € α, and F logically implies (F – {α->β}) ∪ {(α –> A) ->β
  - Attribute A is extraneous in β if A € β and the set of functional dependencies (F – {α->β}) ∪ {(α-> (β– A)} logically implies F.

Algorithm for computing Canonical Cover for set of functional Dependencies F

Fc = F

repeat

Use the union rule to replace any dependencies in Fc of the form

$\alpha 1 \rightarrow \beta 1$ and $\alpha 1 \rightarrow \beta 2$ and $\alpha 1 \rightarrow \beta 1\beta 2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β.

/* The test for extraneous attributes is done using Fc, not F */

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$ in F_c .

until (F_c does not change)

7.Consider the following functional dependencies over the attribute set R(ABCDE) for finding minimal cover FD = {A->C, AC->D, B->ADE}

Solution:

Step 1: Split the FD such that R.H.S contain single attribute. Hence, we get

A->C

AC->D

B->A

B->D

B->E

Step 2: Find the **redundant entries** and **delete** them. This can be done as follows -

- **For A->C:** We find (A)+ by assuming that we delete A->C temporarily. We get (A)+={A}. Thus, from A it is not possible to obtain C by deleting A->C. This means we cannot delete A->C
- **For AC->D:** We find (AC)+ by assuming that we delete AC->D temporarily. We get (AC)+={AC}. Thus, by such deletion it is not possible to obtain D. This means we cannot delete AC->D
- **For B->A:** We find (B)+ by assuming that we delete B->A temporarily. We get (B)+={BDE}. Thus, by such deletion it is not possible to obtain A. This means we cannot delete B->A
- **For B->D:** We find (B)+ by assuming that we delete B->D temporarily. We get (B)+={BEACD}. This shows clearly that even if we delete B->D we can obtain D. This means we can delete B->A. Thus, it is redundant.
- **For B->E:** We find (B)+ by assuming that we delete B->E temporarily. We get (B)+={BDAC}. Thus, by such deletion it is not possible to obtain E. This means we cannot delete B->E
- To summarize we get now

  A->C

  AC->D

  B->A

  B->E

- Thus R.H.S gets simplified.

  Step 3: Now we will simplify L.H.S.

  Consider AC->D. Here we can split A and C. For that we find closure set of A and C.

  (A)+ = (AC)

  (C)+ = (C)

- Thus, C can be obtained from both A as well as C. That also means we need not have to have AC on L.H.S. Instead, only A can be allowed and C can be eliminated. Thus, after simplification we get

  A->D

- To summarize we get now

A->C

A->D

B->A

B->E

- Thus L.H.S gets simplified.

  Step 4: The simplified L.H.S. and R.H.S can be combined together to form

  A->CD

  B->AE

- This is a **minimal cover** or **Canonical cover** of functional dependencies.

8.Write about Update Anomalies in detail

Concept of Redundancy and Anomalies

Definition:

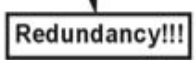
- Redundancy is a condition created in database in which same piece of data is held at two different places.
- Redundancy is at the root of several problems associated with relational schemas.

Problems caused by redundancy: Following problems can be caused by redundancy

1. **Redundant storage:** Some information is stored repeatedly.
2. **Update anomalies:** If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.
3. **Insertion anomalies:** Due to insertion of new record repeated information get added to the relation schema.
4. **Deletion anomalies:** Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

- **Example:** Following example illustrates the above discussed anomalies or redundancy problems.
- Consider following Schema in which all possible information about Employee is stored.

EmpIID	EName	Salary	DeptID	DeptName	DeptLoc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai



1) Redundant storage: Note that the information about **DeptID**, **DeptName** and **DeptLoc** is repeated.

2) Update anomalies: In above table if we change **DeptLoc** of Pune to Chennai, then it will result **inconsistency** as for DeptID 101 the DeptLoc is Pune. Or otherwise, we need to **update multiple copies** of **DeptLoc** from Pune to Chennai. Hence, this is an update anomaly.

3) Insertion anomalies: For above table if we want to add new tuple say (5, EEE,50000) for **DeptID 101** then it will cause repeated information of (101, XYZ,Pune) will occur.

4) Deletion anomalies: For above table, if we delete a record for **EmpID 4,** then automatically information about the **DeptID 102, DeptName PQR** and **DeptLoc Mumbai** will get deleted and one may not be aware about **DeptID 102**. This causes deletion anomaly

9.Write about Properties of Relational Decomposition with an example. OR Discuss the procedure used for loss-less decomposition with an example. (APR/MAY 2024)

Decomposition

- Decomposition is the process of breaking down one table into multiple tables.
- **Formal definition of decomposition is** –
    - A decomposition of relation Schema R consists of replacing the relation Schema by two relation schemas that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.
    - For example - Consider the following table

Employee_Department table as follows –

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	Hyderabad	25000	D005	Human Resource

- We can decompose the above relation Schema into two relation schemas as **Employee (Eid, Ename, Age, City, Salary)** and **Department (Deptid, Eid, DeptName)**. as follows -

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	Hyderabad	25000

Department Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

- The decomposition is used for eliminating redundancy.
- **For example:** Consider following relation **Schema R** in which we assume that the grade determines the salary, the redundancy is caused **Schema R**

Name	eid	deptname	Grade	Salary
AAA	121	Accounts	2	8000
AAA	132	Sales	3	7000
BBB	101	Marketing	4	7000
CCC	106	Purchase	2	8000

Redundancy!!!

- Hence, the above table can be decomposed into two Schema S and T as follows:

Schema S

Name	eid	deptname	Grade
AAA	121	Accounts	2
AAA	132	Sales	3
BBB	101	Marketing	4
CCC	106	Purchase	2

Schema T

Grade	Salary
2	8000
3	7000
4	7000
2	8000

Problems Related to Decomposition:

- Following are the potential problems to consider:

  1) Some queries become more **expensive**.

  2) Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!

  3) Checking some dependencies may require joining the instances of the decomposed relations.

  4) There may be loss of information during decomposition.

Properties Associated with Decomposition

- There are two properties associated with decomposition and those are

  1) Loss-less Join or non-Loss Decomposition: When all information found in the original database is preserved after decomposition, we call it as loss less or non-loss decomposition.

  2) Dependency Preservation: This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

Non-loss Decomposition or Loss-less Join (APR/MAY 2024)

- **The lossless join can be defined using following three conditions:**

  i) Union of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

  Att(R1) ∪ Att(R2) = Att(R)

  ii) Intersection of attributes of R1 and R2 must not be NULL.

  Att(R1) ∩ Att(R2) ≠ Φ

  iii) Common attribute must be a key for at least one relation (R1 or R2)

  Att(R1) ∩ Att(R2) -> Att(R1) **OR**

  Att(R1) ∩ Att(R2) -> Att(R2)

10.Consider the following relation R (A, B, C, D) and FDs A->BC, is the decomposition of R into R1(A, B, C), R2(A, D). Check if the decomposition is lossless join or not.

Solution:

Step 1: Here Att(R1) ∪ Att(R2) = Att(R) i.e. R1(A, B, C) ∪ R2(A, D) = (A, B, C, D) i.e. R. Thus, first condition gets satisfied.

Step 2: Here R1 ∩ R2={A}. Thus Att(R1) ∩ Att(R2) ≠ Φ. Here the second condition gets satisfied.

Step 3: Att(R1) ∩ Att(R2) -> {A}. Now (A)+= {A, B, C} € attributes of R1. Thus, the third condition gets satisfied.

- This shows that the given decomposition is a **lossless join**.

11.Consider the following relation R (A, B, C, D, E, F) and FDs A->BC, C->A, D->E, F->A, E->D is the decomposition of R into R1(A, C, D), R2(B, C, D), and R3(E, F, D). Check for lossless.

Solution:

Step 1: R1 R2 R3=R. Here the first condition for checking lossless join is satisfied as (A, C, D) ∪ (B, C, D) ∪ (E, F, D) = {A, B, C, D, E, F} which is nothing but R.

Step 2: Consider R1∩ R2={CD} and R2∩R3={D}. Hence second condition of intersection not being Φ gets satisfied.

Step 3: Now, consider R1(A, C, D) and R2(B, C, D). We find R1∩R2={CD} (CD)+ = {ABCDE} € attributes of R1 i.e. {A, C, D}. Hence condition 3 for checking lossless join for R1 and R2 gets satisfied.

Step 4: Now, consider R2(B, C, D) and R3(E, F, D). We find R2∩R3={D}. (D)+= {D, E} which is neither complete set of attributes of R2 or R3. [Note that F is missing for being attribute of R3].

- Hence it is not **lossless join decomposition.** Or in other words we can say it is a **lossy decomposition.**

12.Suppose that we decompose schema R=(A,B,C,D,E) into (A,B,C) (C,D,E). Show that it is not a lossless decomposition.

Solution:

Step 1: Here we need to assume some data for the attributes A, B, C, D, and E. Using this data we can represent the relation as follows –

Relation R

A	B	C	D	E
a	1	x	p	q
b	2	x	r	s

Relation R1 = (A, B, C)

A	B	C
a	1	x
b	2	x

Relation R2 = (C, D, E)

C	D	E
x	p	q
x	r	s

Step 2: Now we will join these tables using natural join, i.e. they join based on common attribute C. We get R1 ⋈ R2 as

A	B	C	D	E
a	1	x	p	q
a	1	x	r	s
b	2	x	p	q
b	2	x	r	s

- Here we get more rows or tuples than original relation R.
- Clearly R1 ⋈ R2 □ R. Hence it is not lossless decomposition.

13.Explain the concept of Dependency Preservation

Dependency Preservation

- **Definition:** A Decomposition D = {R1, R2, R3…. Rn} of R is dependency preserving for a set F of Functional dependency if - (F1 ∪ F2 ∪ … ∪ Fm) = F.
- If decomposition is not dependency-preserving, some dependency is lost in the decomposition.
- **Example1: Consider the relation R (A, B, C) for functional dependency set {A -> B and B -> C} which is decomposed into two relations R1 = (A, C) and R2 = (B, C). Then check if this decomposition dependency preserving or not.**

  Solution: This can be solved in following steps:

  Step 1: For checking whether the decomposition is dependency preserving or not we need to check following condition

  F+ = (F1 F2) +

  Step 2: We have with us the F+ = {A ->B and B-> C}

  Step 3: Let us find (F1) + for relation R1 and (F2) + for relation R2

R1(A,C)	R2(B,C)
A->A Trivial	B->B Trivial
C->C Trivial	C->C Trivial
A->C ∵ In (F)+A->B->C and it is Nontrivial	B->C ∵ In (F)+ B->C and it is Non-Trivial
AC->AC Trivial	BC->BC Trivial
A->B but is not useful as B is not part of R1 set	We can not obtain C->B
We can not obtain C->A	

Step 4: We will eliminate all the trivial relations and useless relations. Hence, we can obtain R1 and R2 as

R1(A,C)	R2(B,C)
A->C Nontrivial	B->C Non-Trivial

(F1∪ F2) + = {A->C, B->C} {A->B, B->C} i.e.(F)+

- Thus, the condition specified in step 1 i.e. F+= (F1 F2) + **is not true.** Hence it is **not dependency preserving decomposition.**

Example2: Let relation R (A, B, C, D) be a relational schema with following functional dependencies {A->B, B->C, C->D, and D->B}. The decomposition of R into (A, B), (B, C) and (B, D). Check whether this decomposition is dependency preserving or not.

Solution:

Step 1: Let (F)+ = {A->B, B->C, C->D, D->B}.

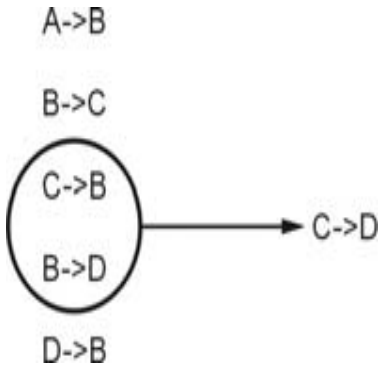
Step 2: We will find (F1) +, (F2) +, (F3) + for relations R1(A, B), R2(B, C) and R3(B, D) as follows –

R1(A,B)	R2(B,C)	R3(B,D)
A->A Trivial	B->B Trivial	B->B Trivial
B->B Trivial	C->C Trivial	D->D Trivial
A->B ∵ (F)+ and it's non Trivial	B->C ∵ (F)+ and it's non Trivial	B-> D ∵ (F)+ as and B->C->D and it's non Trivial
B->A can not be obtained	C->B ∵ In (F)+ and C->D->C and it is Nontrivial	D->B ∵ (F)+ and it's non Trivial
AB->AB	BC->BC Trivial	BD->BD Trivial

Step 3: We will eliminate all the trivial relations and useless relations. Hence, we can obtain R1 ∪ R2 ∪ R3 as

R1(A,B)	R2(B,C)	R2(B,D)
A->B	B->C	B-> D
	C->B	D->B

Step 4: As from above FD's we get



Step 5: This proves that **F+= (F1 F2 F3) +**.

- Hence given **decomposition is dependency preserving**.

14.Explain about Normalization in detail. Or Exemplify in detail about First, Second and Third Normal Forms with relevant examples. (APR/MAY 2023) OR Explain various Normal Forms in database management systems which are required for fulfilling normalization requirements of an organization. (NOV/DEC 2022) NOV/DEC 2023)

Normal Forms

- Normalization is the process of reorganizing data in a database so that it meets **two basic requirements**:

  1) There is **no redundancy** of data (all data is stored in only one place).

  2) **data dependencies** are logical (all related data items are stored together)
- The normalization is important because it allows database to take up **less disk space**.
- It also helps in increasing the **performance**.

First Normal Form

- The table is said to be in 1NF if it follows following rules -

  i) It should only have single (atomic) valued attributes/columns.

  ii) Values stored in a column should be of the same domain

  iii) All the columns in a table should have unique names.

  iv) And the order in which data is stored, does not matter.
- Consider following Student table

Student

sid	sname	Phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

- As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. We can make it in 1NF. The conversion is as follows –

sid	sname	Phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

Second Normal Form

- Before understanding the second normal form let us first discuss the concept of partial functional dependency and prime and non-prime attributes.

Concept of Partial Functional Dependency

- Partial dependency means that a nonprime attribute is functionally dependent on part of a candidate key.
- **For example:** Consider a relation R (A, B, C, D) with functional dependency {AB->CD, A->C}
- Here (AB) is a candidate key because
  (AB)+ = {ABCD}={R}
- Hence {A, B} are prime attributes and {C, D} are non-prime attribute. In A->C, the non-prime attribute C is dependent upon A which is actually a part of candidate key AB.
- Hence due to A->C we get partial functional dependency.

Prime and Non-Prime Attributes

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

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

- **Example: Consider a Relation** R= {A, B, C, D} and candidate key as AB.
  1. Prime attributes: A, B
  2. Non-Prime attributes: C, D

The Second Normal Form

- For a table to be in the Second Normal Form, following conditions must be followed
  i) It should be in the First Normal form.
  ii) It should not have partial functional dependency**.**
- **For example:** Consider following table in which every information about the student is maintained in a table such as student id(sid), student name(sname), courseid(cid) and course name(cname).

Student_Course

sid	sname	cid	cname
1	AAA	101	C
2	BBB	102	C++
3	CCC	101	C
4	DDD	103	Java

- This table is not in 2NF. For converting above table to 2NF we must follow the following steps -

  Step 1: The above table is in 1NF.

  Step 2: Here **sname** and **sid** are associated similarly **cid** and **cname** are associated with each other. Now if we delete a record with **sid=2**, then automatically the course C++ will also get deleted. Thus,

  sid->sname or **cid->cname** is a partial functional dependency, because **{sid,cid}** should be essentially a candidate key for above table. Hence to bring the above table to 2NF we must decompose it as follows:

Student

sid	sname	cid
1	AAA	101
2	BBB	102
3	CCC	101
4	DDD	103

Here candidate key is (sid,cid) and (sid,cid)->sname

Course

cid	cname
101	C
102	C++
101	C
103	Java

Here candidate key is cid
Here cid->cname

- Thus, now table is in 2NF as there is no partial functional dependency.

Third Normal Form

- Before understanding the third normal form let us first discuss the concept of transitive dependency, super key and candidate key.

Concept of Transitive Dependency

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. **For example -** X -> Z is a transitive dependency if the following functional dependencies hold true:

  X->Y

  Y->Z

Concept of Super key and Candidate Key

Super key: A super key is a set or one of more columns (attributes) to uniquely identify rows in a table.

Candidate key: The minimal set of 0attributes which can uniquely identify a tuple is known as candidate key. For example, consider following table

RegID	RollNo	Sname
101	1	AAA
102	2	BBB
103	3	CCC
104	4	DDD

Superkeys

- {RegID}
- {RegID, RollNo}
- {RegID,Sname}
- {RollNo,Sname}
- {RegID, RollNo,Sname}

Candidate Keys

- {RegID}
- {RollNo}

Third Normal Form

- A table is said to be in the Third Normal Form when,

  i) It is in the Second Normal form. (i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

- In other words, 3NF can be defined as: A table is in 3NF if it is in 2NF and for each functional dependency

  X-> Y

- at least one of the following conditions hold:

  i) X is a super key of table

  ii) Y is a prime attribute of table

- For example: Consider following table **Student_details** as follows –

sid	sname	zipcode	cityname	state
1	AAA	11111	Pune	Maharashtra
2	BBB	22222	Surat	Gujarat
3	CCC	33333	Chennai	Tamilnadu
4	DDD	44444	Jaipur	Rajastan
5	EEE	55555	Mumbai	Maharashtra

- Here,

  Super keys: {sid}, {sid,sname},{sid,sname,zipcode},{sid,zipcode,cityname}… and so on.

  Candidate keys: {sid}

  Non-Prime attributes: {sname,zipcode,cityname,state}

  The dependencies can be denoted as

  sid->sname

  sid->zipcode

  zipcode->cityname

  cityname->state

- The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows:

Student

sid	sname	zipcode
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Zip

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Boyce / Codd Normal Form (BCNF)

- 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.
  Or in other words,
- For a table to be in BCNF, following conditions must be satisfied:
  i) R must be in 3rd Normal Form
  ii) For each functional dependency (X → Y), X should be a super Key.
- In simple words if Y is a prime attribute, then X cannot be non-prime attribute.
- **For example -** Consider following table that represents that a student enrolment for the course -

Enrolment Table

Sid	Course	Teacher
1	C	Ankita
1	Java	Poonam
2	C	Ankita
3	C++	Supriya
4	C	Archana

- From above table following observations can be made:
  - One student can enroll for multiple courses. For example, student with sid=1 can enroll for C as well as Java.
  - For each course, a teacher is assigned to the student.
  - There can be multiple teachers teaching one course for example course C can be taught by both the teachers namely - Ankita and Archana.
  - The candidate key for above table can be (sid,course), because using these two columns we can find.
  - The above table holds following dependencies

    (sid,course)->Teacher

    Teacher->course
- The above table is not in BCNF because of the dependency **teacher->course**. Note that the teacher is not a super key or in other words, **teacher** is a non-prime attribute and **course** is a prime attribute and non-prime attribute derives the prime attribute.
- To convert the above table to BCNF we must decompose above table into Student and Course tables

Student

sid	Teacher
1	Ankita
1	Poonam
2	Ankita
3	Supriya
4	Archana

Course

Teacher	course
Ankita	C
Poonam	Java
Ankita	C
Supriya	C++
Archana	C

- Now the table is in BCNF.

Multivalued Dependencies and Fourth Normal Form

Concept of Multivalued Dependencies

- A table is said to have multi-valued dependency, if the following conditions are true,

  1) For a dependency A B, if for a single value of A, **multiple values** of B exist, then the table may have multi-values dependency.

  2) Also, a table should have **at-least 3 columns** for it to have a multi-valued dependency.

  3) And, for a relation R (A, B, C), if there is a multi-valued **dependency between**, A and B, then B and C should be independent of each other.
- If all these conditions are true for any relation(table), it is said to have multi-valued dependency.
- In simple terms, if there are two columns A and B - and for column A if there are multiple values of column B then we say that MVD exists between A and B.
- The multivalued dependency is denoted by ->>
- If there exists a multivalued dependency then the table is **not in 4th normal form.**
- **For example:** Consider following table for information about student

Student

sid	Course	Skill
1	C C++	English German
2	Java	English French

- Here sid =1 leads to multiple values for courses and skill. Following table shows this

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

- Here, **sid** and **course** are dependent but the **Course** and **Skill** are independent. The multivalued dependency is denoted as:

  sid ->>Course

  sid->>Skill

Fourth Normal Form

Definition: For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1) It should be in the Boyce-Codd Normal Form (BCNF).

2) And, the table should not have any multi-valued dependency.

- **For example:** Consider following student relation which is not in 4NF as it contains multivalued dependency.

Student Table

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

- Now to convert the above table to 4NF we must decompose the table into following two tables.

Student_Course Table

Key: (sid,Course)

sid	Course
1	C
1	C++
2	Java

Student_Skill Table

Key: (sid,Skill)

sid	Skill
1	English
1	German
2	English
2	French

- Thus, the tables are now in 4NF.

Join Dependencies and Fifth Normal Form

Concept of Join Dependencies

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over C is equal to relation R, then we can say that a Join Dependency (JD) exists.

- Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.
- A JD ⋈ {R1, R2..., Rn} is said to hold over a relation R if R1, R2....., Rn is a lossless-join decomposition.
- The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.
- Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

Concept of Fifth Normal Form

- The database is said to be in 5NF if -
  i) It is in 4th Normal Form
  ii) If we can decompose table further to eliminate redundancy and anomalies and when we rejoin the table, we should not be losing the original data or get a new record (**join Dependency Principle**)
- The fifth normal form is also called as **project join normal form.**
- **For example -** Consider following table

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Britania	Biscuits

- Here we assume the keys as {Seller, Company, Product}
- The above table has multivalued dependency as Seller {Company, Product}. Hence, table is not in 4th Normal Form. To make the above table in 4th normal form, we decompose above table into two tables as

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuits

- The above table is in 4th Normal Form as there is no multivalued dependency. But it is not in 5th normal form because if we join the above two table we may get,

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Amul	Biscuits
Sunil	Britania	Icecream
Sunil	Britania	Biscuits

Newly added records which are not present in original table

- To avoid the above problem, we can decompose the tables into three tables as Seller_Company, Seller_Product, and Company Product table

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuit

Company_Product

Company	Product
Godrej	Cinthol
Dabur	Honey
Dabur	HairOil
Dabur	RoseWater
Amul	Icecream
Britania	Biscuit

- Thus, the table in in 5th normal form.

15. Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F= {{A, B} C, A {D, E}, B F, F {G, H}, D {I, J} }

1. What is the key for R? Demonstrate it using the inference rules.

2. Decompose R into 2NF, then 3NF relations.

Solution: Let, $A \rightarrow DE$ (given)

$\therefore A \rightarrow D, A \rightarrow E$

As $D \rightarrow I\ J, A \rightarrow I\ J$

Using union rule we get

$A \rightarrow DEIJ$

As $A \rightarrow A$

we get $A \rightarrow ADEIJ$

Using augmentation rule we compute AB

$AB \rightarrow ABDEIJ$

But $AB \rightarrow C$ (given)

$\therefore AB \rightarrow ABCDEIJ$

$B \rightarrow F$ (given) $F \rightarrow GH \therefore B \rightarrow GH$ (transitivity)

$\therefore AB \rightarrow AGH$ is also true

- Similarly, $AB \rightarrow AF \because B \rightarrow F$ (given)
- Thus, now using union rule

$AB \rightarrow ABCDEFGHIJ$

$\therefore$ AB is a key

- The table can be converted to 2NF as

  R_1 = (A, B, C)

  R_2 = (A, D, E, I, J)

  R_3 = (B, F, G, H)

- The above 2NF relations can be converted to 3NF as follows

  R_1 = (A, B, C)

  R_2 = (A, D, E)

  R_3 = (D, I, J)

  R_4 = (B, E)

  R_5 = (E, G, H).

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai,

Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / III SEM

UNIT IV – TRANSACTION MANAGEMENT

SYLLABUS:

Transaction concepts – properties – Schedules – Serializability – Concurrency Control – Two phase locking techniques.

PART A

1. Define Transaction with an example. (APR/MAY 2024)

- A transaction can be defined as a **group of tasks** that form a single logical unit.
- **For example -** Suppose we want to withdraw ` 100 from an account then we will follow following operations:

  1) Check account balance

  2) If sufficient balance is present request for withdrawal.

  3) Get the money

  4) Calculate Balance = Balance –100

  5) Update account with new balance.

2. List out the ACID properties. Or What are the ACID properties? (NOV/DEC 2023)

- Atomicity
- Consistency
- Isolation
- Durability

PREPARED BY: Mrs. S. JAYABHARATHI AP/AI&DS

3.State the Atomicity property of a Transaction.

- This property states that each transaction must be considered as a **single unit** and **must be completed fully or not completed at all**.
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not be in a state 'executing'**.

4.What are states of transaction?

- Various states of transaction are
  (1) Active
  (2) Partially Committed
  (3) Failed
  (4) Aborted
  (5) Committed.

5.What is meant by concurrency control?

- A mechanism which ensures that simultaneous execution of more than one transaction does not lead to any database inconsistencies is called concurrency control mechanism.

6.State the need for concurrency control. (or) Why is it necessary to have control of concurrent execution of transactions? How is it made possible?

- Following are the purposes of concurrency control-
  1. To ensure isolation
  2. To resolve read-write or write-write conflicts
  3. To preserve consistency of database

7.List commonly used concurrency control techniques.

- The commonly used concurrency control techniques are -
  i) Lock
  ii) Timestamp
  iii) Snapshot Isolation

8.What is meant by serializability? How it is tested?

- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.
- It is tested using precedence graph technique.

9.What are the two-phases in locking?

- There are two phases in two-phase locking protocol:

  a. Growing Phase (Locking Phase): It is a phase in which the transaction may obtain locks but does not release any lock.

  b. Shrinking Phase (Unlocking Phase): It is a phase in which the transaction may release the locks but does not obtain any new lock.

10.What is the difference between shared lock and exclusive lock?

Shared Lock	**Exclusive lock**
Shared lock is used for when the transaction wants to perform read operation.	Exclusive lock is used when the transaction wants to perform both mad and write operation.
Multiple shared lock can be set on a transaction simultaneously.	Only one exclusive lock can be placed on a data item at a time
Using shared lock data item can be viewed.	Using exclusive lock data can be inserted or deleted.

11.What benefit does strict two-phase locking provide? What disadvantages result? (APR/MAY 2024)

Benefits

1. This ensure that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits and preventing other transaction from reading that data.

2. This protocol solves dirty read problem.

Disadvantage

1. Concurrency is reduced.

12. What is serializable schedule?

- The schedule in which the transactions execute one after the other is called serial schedule.
- It is consistent in nature.
- **For example:** Consider following two transactions T1 and T2

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

- All the operations of transaction T1 on data items A and then B executes and then in transaction T2 all the operations on data items A and B execute.
- The R stands for Read operation and W stands for write operation.

13. When are two schedules conflict equivalent?

- Two schedules are conflict equivalent if:
- They contain the same set of the transaction.
- Every pair of conflicting actions is ordered the same way.
- **For example –**

Non Serial Schedule

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Serial Schedule

T1	T2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

14. What does time to commit mean?

- The COMMIT command is used to save permanently any transaction to database.
- When we perform, Read or Write operations to the database then those changes can be undone by rollback operations.
- To make these changes permanent, we should make use of commit.

15. Define Topological Sorting.

- A serializability order of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting.**

16. Define Dirty Read or Uncommitted Read Problem.

- The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction.

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A-20
	W(A)
	Commit
Commit	

Dirty read

17. What are the Types of Locks? Or What are the different modes of Locks? (NOV/DEC 2023)

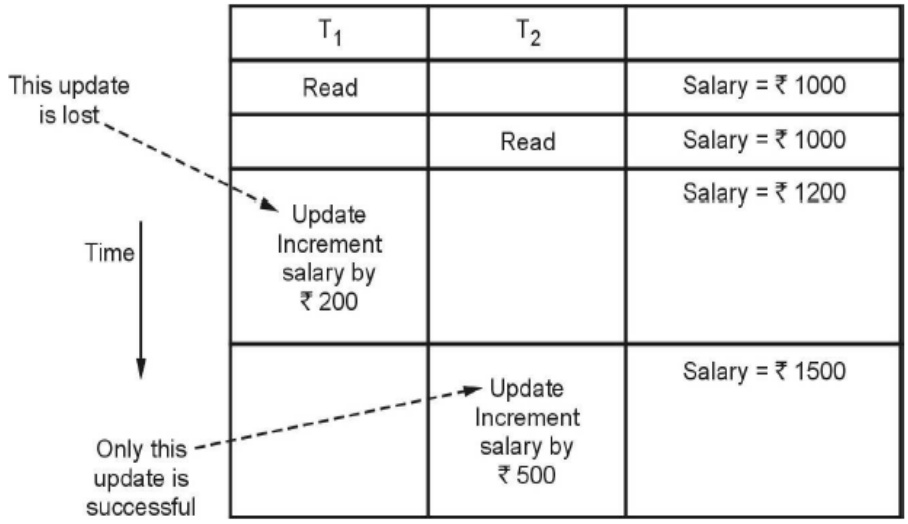
- There are two types of locks used –

i) Shared Lock: The shared lock is used for reading data items only. It is denoted by **Lock-S**. This is also called as **read lock.**

ii) Exclusive Lock: The exclusive lock is used for both read and write operations. It is denoted as **Lock-X.** This is also called as **write lock.**

18. Define Lost Update Problem with example.

- This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.
- **For example** – Consider following transactions



19. Define Rigorous Two-Phase Locking:

- This is stricter two-phase locking protocol. Here all locks are to be held until the transaction commits. The transactions can be serialized in the order in which they commit.

20. Define Strict Two-Phase locking.

- The strict 2PL protocol is a basic two-phase protocol but **all the exclusive mode locks** be **held until the transaction commits**.
- That means in other words all the **exclusive locks** are **unlocked** only after the **transaction is committed**.

PART B

1.Write a short note on Transaction Concepts. Also, Describe the ACID properties of transaction processing. (APR/MAY 2023)

Transaction Concepts.

Definition:

- A transaction can be defined as a **group of tasks** that form a single logical unit.
- **For example -** Suppose we want to withdraw ` 100 from an account then we will follow following operations:
  1) Check account balance
  2) If sufficient balance is present request for withdrawal.
  3) Get the money
  4) Calculate Balance = Balance –100
  5) Update account with new balance.
- The above mentioned **four steps** denote **one transaction**.
- In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database.

ACID Properties

1) Atomicity:

- This property states that each transaction must be considered as a **single unit** and **must be completed fully or not completed at all**.
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not be in a state 'executing'**.
- **For example -** In above mentioned withdrawal of money transaction all the five steps must be completed fully or none of the step is completed. Suppose if transaction gets failed after step 3, then the customer will get the money but the balance will not be updated accordingly. The state of database should be either at before ATM withdrawal (i.e customer without withdrawn money) or after ATM withdrawal (i.e. customer with money and account updated). This will make the system in consistent state.

2) Consistency:

- The database must remain in consistent state after performing any transaction.

For example: In ATM withdrawal operation, the balance must be updated appropriately after performing transaction. Thus, the database can be in consistent state.

3) Isolation:

- In a database system where **more than one transaction** are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.
- **For example:** If a bank manager is checking the account balance of particular customer, then manager should see the balance either before withdrawing the money or after withdrawing the money. This will make sure that each individual transaction is completed and any other dependent transaction will get the consistent data out of it. Any failure to any transaction will not affect other transaction in this case. Hence it makes all the transactions consistent.

4) Durability:

- The database should be **strong enough** to handle any **system failure**.
- If there is any set of insert /update, then it should be able to handle and commit to the database.
- If there is any **failure**, the database should be able to **recover** it to the consistent state.
- **For example:** In ATM withdrawal example, if the system failure happens after Customer getting the money, then the system should be strong enough to update Database with his new balance, after system recovers. For that purpose, the system has to keep the **log of each transaction and its failure**. So, when the system recovers, it should be able to know when a system has failed and if there is any pending transaction, then it should be updated to Database.

2. Sketch the state transition diagram for a database transaction and explain the flow. (APR/MAY 2023)

Transaction States

- Each transaction has following five states as shown in figure 4.1

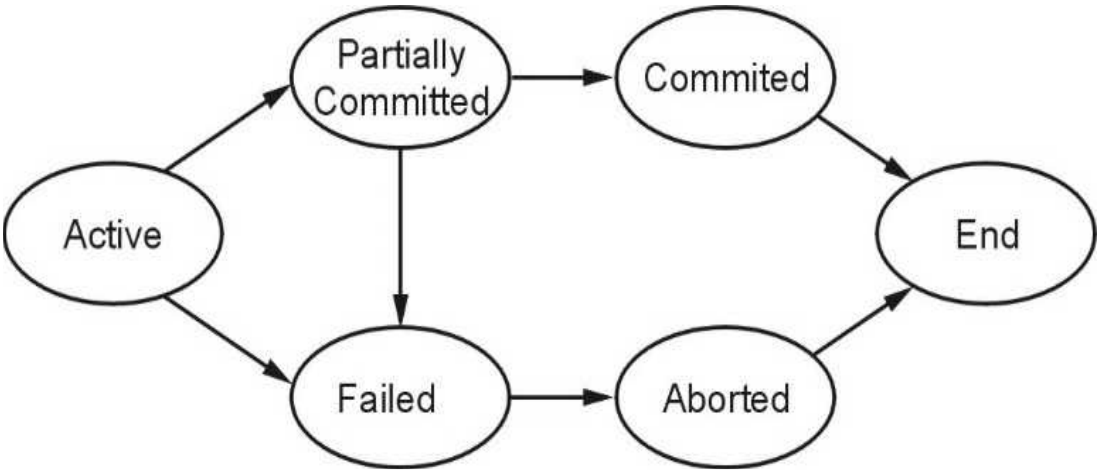


Figure 4.1 Transaction States

1) Active: This is the first state of transaction. For example: insertion, deletion or updating of record is done here. But data is not saved to database.

2) Partially Committed: When a transaction executes its final operation, it is said to be in a partially committed state.

3) Failed: A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

4) Aborted: If a transaction is failed to execute, then the database recovery system will make sure that the database is in its previous consistent state. If not, it brings the database to consistent state by aborting or rolling back the transaction.

5) Committed: If a transaction executes all its operations successfully, it is said to be committed. This is the last step of a transaction, if it executes without fail.

3. Define a transaction. Then discuss the following with relevant examples:

1. **A read only transaction**
2. **A read write transaction**
3. **An aborted transaction**

Transaction

- A transaction can be defined as a **group of tasks** that form a single logical unit.

Solution:

1. Read only transaction

T1
Read(A)
Read(B)
Display(A-B)

2. A read write transaction

T1
Read(A)
A=A+100
Write(A)

3. An aborted transaction

T1	T2	
Read(A)		Assume A=100
A=A+50		A=150
Write(A)		
	Read(A)	A=150
	A=A+100	A=250
RollBack		A=100 (restore back to original value which is before Transaction T1)
	Write(A)	

4. Define Schedules and also explain the types of Schedules.

Schedules

- Schedule is an order of multiple transactions executing in concurrent environment.
- Following figure 4.2 represents the types of schedules.

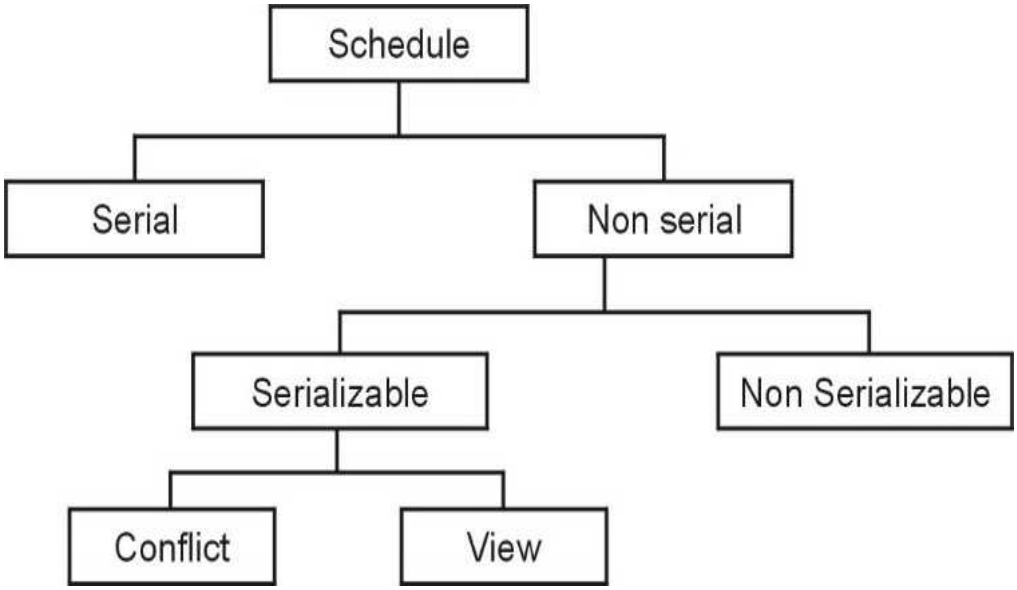


Figure 4.2 Types of Schedules

Serial Schedule:

- The schedule in which the transactions execute one after the other is called serial schedule. It is consistent in nature.
- **For example:** Consider following two transactions T1 and T2.

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

- All the operations of transaction T1 on data items A and then B executes and then in transaction T2 all the operations on data items A and B execute.
- The **R stands for Read** operation and **W stands for write** operation.

Non-Serial Schedule:

- The schedule in which operations present within the transaction are intermixed. This may lead to conflicts in the result or inconsistency in the resultant data.
- **For example** - Consider following two transactions,

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(B)
R(A)	
W(B)	
	R(B)
	W(B)

- The above transaction is said to be **non-serial** which result in inconsistency or conflicts in the data.

Serializability

- When multiple transactions run concurrently, then it may lead to inconsistency of data (i.e. change in the resultant value of data from different transactions).
- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

- **For example:**

T_1	A	B	T_2
Initial Value	100	100	
A=A-10			
W(A)			
B=B+10			
W(B)			
	90	110	
			A=A-10
			W(A)
	80	110	

- In above transactions initially T1 will read the values from database as A=100, B=100 and modify the values of A and B. But transaction T2 will read the modified value i.e. 90 and will modify it to 80 and perform write operation. Thus, at the end of transaction T1 value of A will be 90 but at end of transaction T2 value of A will be 80. Thus, conflicts or inconsistency occurs here. This sequence can be converted to a sequence which may give us consistent result. This process is called **serializability**.

Difference between Serial Schedule and Serializable Schedule

<table>
<tr><th>Serial Schedule</th><th>Serializable Schedule</th></tr>
<tr><td>No concurrency is allowed in serial schedule.</td><td>Concurrency is allowed in serializable schedule.</td></tr>
<tr><td>In serial schedule, if there are two transactions executing at the same time and no interleaving of operations are permitted, then following can be the
possibilities of execution –
(i) Execute all the operations of transactions T1 in a sequence and then execute all the operations of
transactions T2 in a sequence.
(ii) Execute all the operations of transactions T2 in a sequence and then execute all the operations of
transactions T1 in a sequence.</td><td>In serializable schedule, if there are two transactions executing at the same time and interleaving of operations is allowed there can be
different possible orders of executing an individual operation of the transactions.</td></tr>
<tr><td>Example of Serial Schedule
<table><tr><th>T1</th><th>T2</th></tr><tr><td>Read(A)</td><td></td></tr><tr><td>A=A-50</td><td></td></tr><tr><td>Write(A)</td><td></td></tr><tr><td>Read(B)</td><td></td></tr><tr><td>B=B+100</td><td></td></tr><tr><td>Write(B)</td><td></td></tr><tr><td></td><td>Read(A)</td></tr><tr><td></td><td>A=A+10</td></tr><tr><td></td><td>Write(A)</td></tr></table></td><td>Example of Serializable Schedule
<table><tr><th>T1</th><th>T2</th></tr><tr><td>Read(A)</td><td></td></tr><tr><td>A=A-50</td><td></td></tr><tr><td>Write(A)</td><td></td></tr><tr><td></td><td>Read(B)</td></tr><tr><td></td><td>B=B+100</td></tr><tr><td></td><td>Write(B)</td></tr><tr><td>Read(B)</td><td></td></tr><tr><td>Write(B)</td><td></td></tr></table></td></tr>
</table>

5.Define Serializability. Also, Explain Conflict serializability and view serializability in detail.

Serializability

- When multiple transactions run concurrently, then it may lead to inconsistency of data (i.e. change in the resultant value of data from different transactions).
- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.
- There are **two types of serializability:** conflict serializability and view Serializability

Conflict Serializability

Definition: Suppose T1 and T2 are two transactions and I1 and I2 are the instructions in T1 and T2 respectively. Then these two transactions are said to be conflict Serializable, if both the instruction access the data item d, and at least one of the instructions is write operation.

What is conflict? In the definition **three conditions** are specified for a conflict in conflict serializability –

1) There should be **different transactions**
2) The **operations** must be performed on **same data** items
3) **One of the operations** must be the **Write(W**) operation

- We can test a given schedule for conflict serializability by constructing a **precedence graph** for the schedule, and by searching for absence of cycles in the graph.
- Precedence graph is a directed graph, consisting of G= (V, E) where V is set of vertices and E is set of edges. The set of vertices consists of all the transactions participating in the schedule. The set of edges consists of all edges Ti →Tj for which one of three conditions holds:
    1. Ti executes write(Q) before Tj executes read(Q).
    2. Ti executes read(Q) before Tj executes write(Q).
    3. Ti executes write(Q) before Tj executes write(Q).
- A serializability order of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting.**

Testing for serializability

- Following method is used for testing the serializability: To test the conflict serializability we can draw a graph G= (V, E) where V = vertices which represent the number of transactions. E = edges for conflicting pairs.

  Step 1: Create a node for each transaction.

  Step 2: Find the conflicting pairs (RW, WR, WW) on the same variable (or data item) by different transactions.

  Step 3: Draw edge for the given schedule. Consider following cases

  1. Ti executes write(Q) before Tj executes read(Q), then draw edge from Ti to Tj.
  2. Ti executes read(Q) before Tj executes write(Q), then draw edge from Ti to Tj
  3. Ti executes write(Q) before Tj executes write (Q), then draw edge from Ti to Tj

  Step 4: Now, if precedence graph is cyclic then it is a non-conflict serializable schedule and if the precedence graph is acyclic then it is conflict serializable schedule.

6.Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable? Explain why or why not.

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

Solution:

Step 1: To check whether the schedule is conflict serializable or not we will check from **top to bottom**. Thus, we will start reading from top to bottom as

T1: R(A) -> T1: W(A) ->T2: R(A) -> T2: R(B) ->T1: R(B)->T1: W(B)

Step 2: We will find **conflicting operations**. Two operations are called as conflicting operations if all the following conditions hold true for them

i) Both the **operations belong to different transactions**.

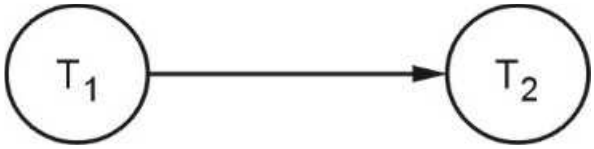
ii) Both the operations are on **same data item**.

iii) **At least one** of the two operations is a **write operation**

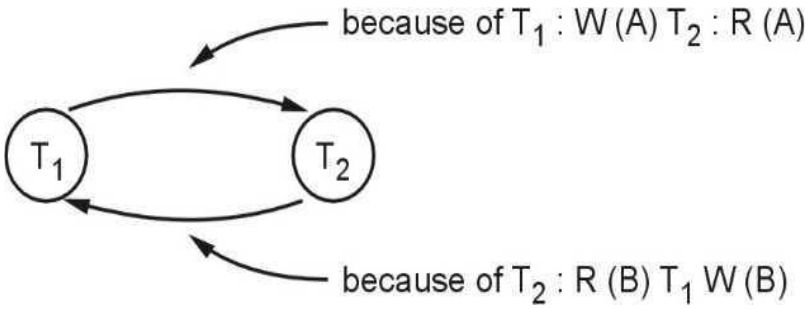
- From above given example in the top to bottom scanning we find the conflict as T1: W(A) ->T2: R(A).
    - i) Here note that there are two different transactions T1 and T2,
    - ii) Both work on same data item i.e. A and
    - iii) One of the operations is write operation

Step 3: We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are two transactions, there will be two nodes namely T1 and T2.

Step 4: Draw the edge between conflicting transactions. For example, in above given scenario, the conflict occurs while moving from T1: W(A) to T2: R(A). Hence edge must be from T1 to T2.



Step 5: Repeat the step 4 while reading from top to bottom. Finally, the **precedence graph** will be as follows



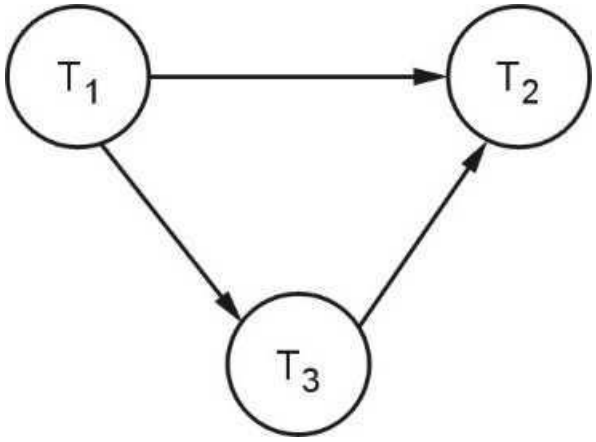
Step 6: Check if **any cycle** exists in the graph. Cycle is a path using which we can start from one node and reach to the same node. If the is cycle found then schedule is not conflict serializable. In the step 5 we get **a graph with cycle**, that means given schedule is **not conflict serializable.**

7. Check whether following schedule is conflict serializable or not. If it is not conflict serializable then find the serializability order.

T_1	T_2	T_3
R(A)		
	R(B)	
		R(B)
	W(B)	
W(A)		
		W(A)
	R(A)	
	W(A)	

Solution:

Step 1: We will read from top to bottom, and build a precedence graph for conflicting entries:



Step 2: As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable**. Hence, we can convert this non serial schedule to serial schedule. For that purpose, we will follow these steps to find the serializable order.

Step 3: A **serializability order** of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting.**

Step 4: Find the vertex which has no incoming edge which is T**1**. Finally find the vertex having no outgoing edge which is T**2**. So in between them is T**3**. Hence the order will be $T_1 – T_3 – T_2$

View Serializability

- If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.
- **View Equivalent Schedule:** Consider two schedules S1 and S2 consisting of transactions

  T_1 and T_2 respectively, then schedules S_1 and S_2 are said to be view equivalent schedule if it satisfies following three conditions :

  - If transaction T_1 reads a data item A from the database initially in schedule S_1, then in schedule S_2 also, T_1 must perform the initial read of the data item X from the database. This is same for all the data items. In other words - the initial reads must be same for all data items.
  - If data item A has been updated at last by transaction T_i in schedule S_1, then in schedule S_2 also, the data item A must be updated at last by transaction T_i.
  - If transaction Ti reads a data item that has been updated by the transaction T_j in schedule S_1, then in schedule S_2 also, transaction T_i must read the same data item that has been updated by transaction T_j. In other words the Write-Read sequence must be same.

Steps to check whether the given schedule is view serializable or not

Step 1: If the schedule is conflict serializable then it is surely view serializable because conflict serializability is a restricted form of view serializability.

Step 2: If it is not conflicting serializable schedule then check whether there exists any blind write operation. The blind write operation is a write operation without reading a value. If there does not exist any blind write then that means the given schedule is not view serializable. In other words, if a blind write exists then that means schedule may or may not be view conflict.

Step 3: Find the view equivalence schedule.

8. Consider the following schedules for checking if these are view serializable or not.

T_1	T_2	T_3
		W(C)
	R(A)	
	W(B)	
R(C)		
		W(B)
W(B)		

Solution:

i) The initial read operation is performed by T2 on data item A or by T1 on data item C. Hence, we will begin with T2 or T1. We will choose T2 at the beginning.

ii) The final write is performed by T1 on the same data item B. Hence T1 will be at the last position.

iii) The data item C is written by T3 and then it is read by T1. Hence T3 should appear before T1. Thus, we get the order of schedule of view serializability as

T2 – T1 – T3

9. Consider following two transactions:

T1: read(A)
read(B)
if A=0 then B: =B+1;
write(B)
T2: read(B);
read(A);
if B=0 then A: =A+1;
write(A)

Let consistency requirement be A=0 V B=0 with A=B=0 the initial values.

1) Show that every serial execution involving these two transactions preserves the consistency of the Database?

2) Show a concurrent execution of T1 and T2 that produces a non-serializable schedule?

3) Is there a concurrent execution of T1 and T2 that produces a serializable schedule?

Solution:

1) There are two possible executions: **T1 ->T2 or T2->T1**

Consider case T**1**->T**2** then

A	B
0	0
0	1
0	1

$A \vee B = A \text{ OR } B = F \vee T = T.$

- This means consistency is met.
- Consider case T**2**->T**1** then

A	B
0	0
1	0
1	0

$A \vee B = A \text{ OR } B = F \vee T = T.$

- This means consistency is met.

2) The concurrent execution means interleaving of transactions T**1** and T**2**. It can be

T_1	T_2
R(A)	
	R(B)
	R(A)
R(B) If A=0 then B=B+1	If B=0 then A=A+1 W(A)
W(B)	

- This is a non-serializable schedule.

3) There is no concurrent execution resulting in a serializable schedule.

10. Test serializability of the following schedule:

i) r1(x); r3(x); w1(x); r2(x); w3(x) ii) r3(x); r2(x); w3(x); r1(x); w1(x)

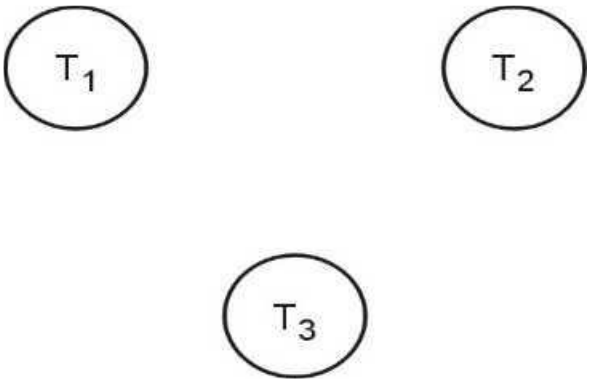
Solution:

i) r1(x); r3(x); w1(x); r2(x); w3(x)

- The r**1** represents the read operation of transaction T**1**, w**3** represents the write operation on transaction T**3** and so on. Hence from given sequence the schedule for three transactions can be represented as follows:

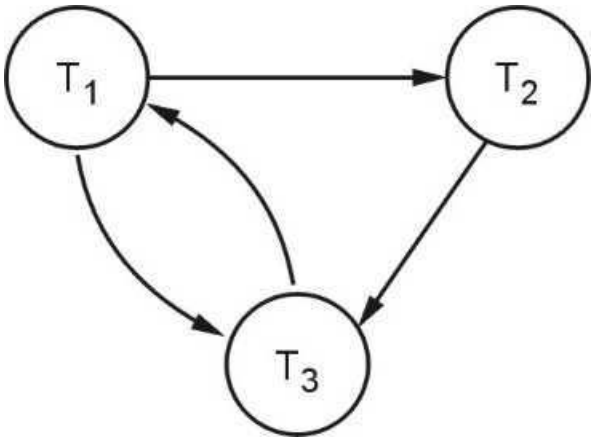
T_1	T_2	T_3
$r_1(x)$		
		$r_3(x)$
$w_1(x)$		
	$r_2(x)$	
		$w_3(x)$

Step 1: We will use the precedence graph method to check the serializability. As there are three transactions, three nodes are created for each transaction.



Step 2: We will read from top to bottom. Initially we read r**1**(x) and keep on moving bottom in search of write operation. Here all the transactions work on same data items i.e. x. Now we get a write operation in T**3** as w**3**(x). Hence the dependency is from T**1** to T**3**.

- Therefore, we draw edge from T**1** to T**3**.
- Similarly, for r**3**(x) we get w**1**(x) pair. Hence there will be edge from T**3** to T**1**. Continuing in this fashion we get the precedence graph as



Step 3: As cycle exists in the above precedence graph, we conclude that it is **not serializable.**

ii) r3(x); r2(x); w3(x); r1(x); w1(x)

- From the given sequence the schedule can be represented as follows:

T_1	T_2	T_3
		$r_3(x)$
	$r_2(x)$	
		$w_3(x)$
$r_1(x)$		
$w_1(x)$		

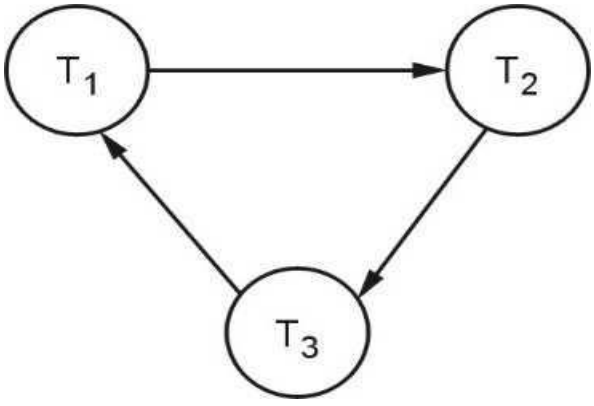
Step 1: Read the schedule from top to bottom for pair of operations. For r**3**(x) we get w**1**(x) pair. Hence edge exists from T**3** to T**1** in precedence graph.

There is a pair from r**2**(x): w**3**(x). Hence edge exists from T**2** to T**3**.

There is a pair from r**2**(x): w**1**(x). Hence edge exists from T**2** to T**1**.

There is a pair from w**3**(x): r**1**(x). Hence edge exists from T**3** to T**1**.

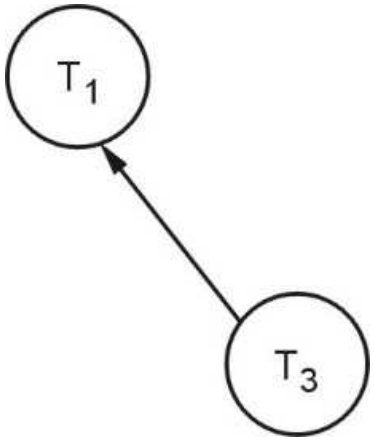
Step 2: The precedence graph will then be as follows –



Step 3: As there is no cycle in the above graph, the given schedule is **serializable.**

Step 4: The serializability order for consistent schedule will be obtained by applying topological sorting on above drawn precedence graph. This can be achieved as follows,

Sub-Step 1: Find the node having no incoming edge. We obtain T**2** is such a node. Hence, T**2** is at the beginning of the serializability sequence. Now delete T**2**. The Graph will be



Sub-Step 2: Repeat sub-Step 1, We obtain T3 and T1 nodes as a sequence. Thus, we obtain the sequence of transactions as T2, T3 and T1. Hence the serializability order is

r2(x); r3(x); w3(x): r1(x); w1(x)

11. Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name.

S1: T1: R(X), T2: R(X), T1: W(Y), T2: W(Y) T1: R(Y), T2: R(Y)

S2: T3: W(X), T1: R(X), T1: W(Y), T2: R(Z), T2: W(Z) T3: R(Z)

For each of the schedules, answer the following questions:

i) What is the precedence graph for the schedule?

ii) Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules?

iii) Is the schedule view-serializable? If so, what are all the view equivalent serial schedules?

Solution:

i) We will find **conflicting operations**. Two operations are called as conflicting operations if all the following conditions hold true for them-

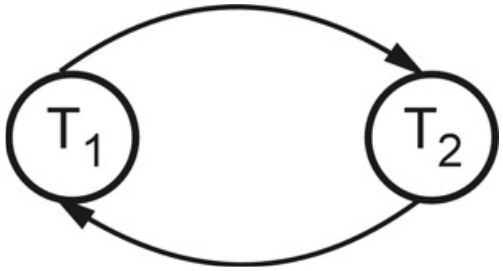
- Both the **operations belong to different transactions**.
- Both the operations are on **same data item**.
- At least one of the two operations is a write operation

For S1: From above given example in the top to bottom scanning we find the conflict as

T1: W(Y), T2: W(Y) and

T2: W(Y), T1: R(Y)

- Hence, we will build the precedence graph. Draw the edge between conflicting transactions. For example, in above given scenario, the conflict occurs while moving from T1: W(Y) to T2: W(Y).
- Hence edge must be from T1 to T2. Similarly for second conflict, there will be the edge from T2 to T1.



For S2: The conflicts are

T3: W(X), T1: R(X)

T2: W(Z), T3: R(Z)

- Hence the precedence graph is as follows –

(ii)

- S1 is **not conflict-serializable** since the dependency **graph has a cycle**.

S2 is conflict-serializable as the dependency graph is acyclic.

The order T2-T3- **T1** is the only equivalent **serial order**.

(iii)

- S1 is **not view serializable**.

- S2 is trivially view-serializable as it is conflict serializable. The **only serial order** allowed is **T2-T3-T1.**

12. Write in detail about Recoverable Schedule and Cascadeless Schedule. (APR/MAY 2024)

Transaction Isolation and Atomicity

- The serializable schedule can be made consistent by applying conflict serializability or view serializability.
- The serializable order makes a transaction isolation. But during the execution of concurrent transactions in a given schedule, some of the transaction may get failed (may be due to hardware or software failure)
- If the transaction gets failed, we need to undo the effect of this transaction to ensure the atomicity property of transaction. This makes the schedule acceptable even after the failure.
- The schedule can be made acceptable by two techniques namely – Recoverable Schedule and Cascadeless schedule.

Recoverable Schedule

Definition:

- A recoverable schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the commit operation of Tj
- **For example:** Consider following schedule, consider A=100

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A-20
	W(A)
	Commit
some transaction…	
Commit	

Failure

- The above schedule is inconsistent if failure occurs after the commit of T2.
- It is because T2 is **dependable transaction** on T1. A transaction is said to be dependable if it contains a **dirty read.**
- The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction.

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A-20
	W(A)
	Commit
Commit	

Dirty read

- Now if the dependable transaction i.e. T2 is committed first and then failure occurs then if the transaction T1 makes any changes then those changes will not be known to the T2. This leads to non-recoverable state of the schedule.
- To make the schedule recoverable we will apply the rule that - commit the independent transaction before any dependable transaction.

- In above example independent transaction is T1, hence we must commit it before the dependable transaction i.e. T2.
- The recoverable schedule will then be -

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A-20
	W(A)
Commit	
	Commit

Cascadeless Schedule

Definition:

- If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written that data item is committed or aborted, then such a schedule is known as a cascadeless schedule.
- The cascadeless schedule allows only **committed Read** operation.
- **For example:**

T_1	T_2	T_3
R(A)		
A=A+50		
W(A)		
Commit		
	R(A)	
	A=A-20	
	W(A)	
	Commit	
		R(A)
		W(A)

- In above schedule at any point if the failure occurs due to commit operation before every Read operation of each transaction, the schedule becomes recoverable and atomicity can be maintained.

13. Define Concurrency Control. Also, Discuss the violations caused by each of the following: dirty read, non-repeatable read and phantoms with suitable example. or Discuss the concurrency control mechanism in detail with suitable example. (NOV/DEC 2023) (APR/MAY 2024)

Introduction to Concurrency Control

- One of the fundamental properties of a transaction is **isolation**.
- When several transactions execute concurrently in the database, however, the isolation property may no longer be preserved.
- A database can have multiple transactions running at the same time. This is called **concurrency.**
- To preserve the isolation property, the system must control the interaction among the concurrent transactions; this control is achieved through one of a variety of mechanisms called **concurrency control schemes.**

Definition of concurrency control:

- A mechanism which ensures that simultaneous execution of more than one transaction does not lead to any database inconsistencies are called **concurrency control mechanism.**
- The concurrency control can be achieved with the help of various protocols such as - lock based protocol, Deadlock handling, Multiple Granularity, Timestamp based protocol, and validation-based protocols.

Need for Concurrency

- Following are the purposes of concurrency control –
    - o **To ensure isolation**
    - o **To resolve read-write or write-write conflicts**
    - o **To preserve consistency of database**
- Concurrent execution of transactions over shared database creates several data integrity and consistency problems – these are

(1) Lost Update Problem: This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

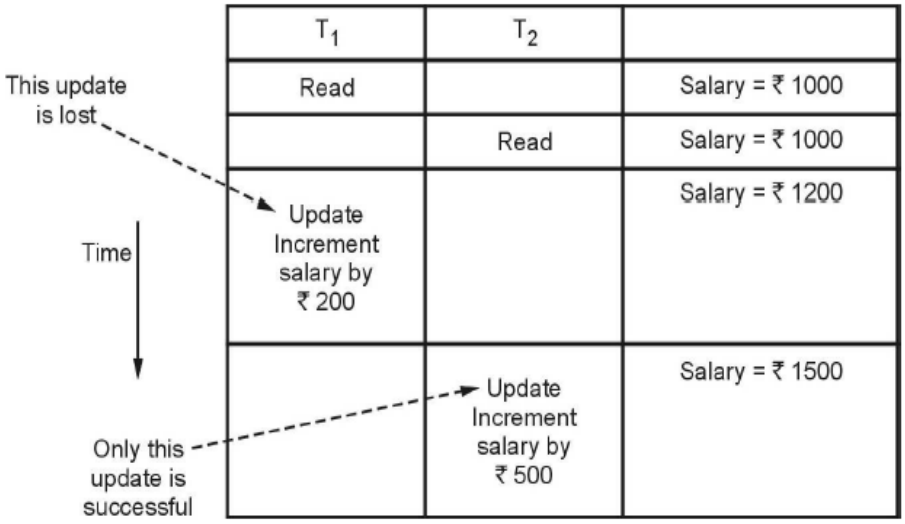
For example – Consider following transactions

(1) Salary of Employee is read during transaction T1.

(2) Salary of Employee is read by another transaction T2.

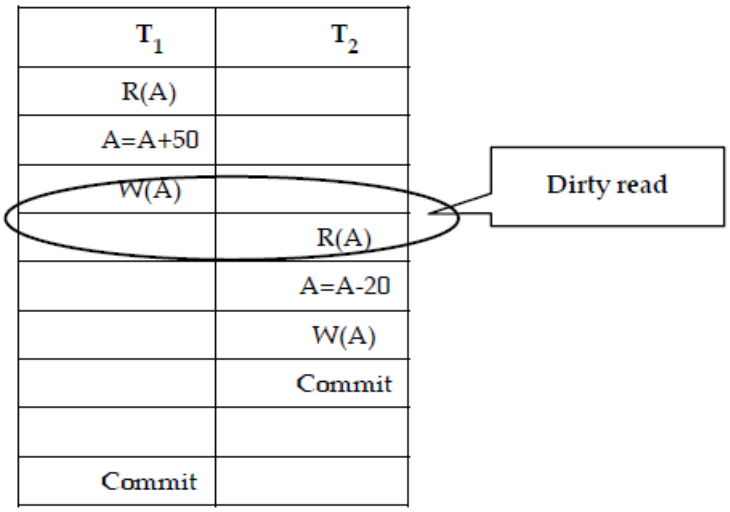
(3) During transaction T1, the salary is incremented by ` 200

(4) During transaction T2, the salary is incremented by ` 500

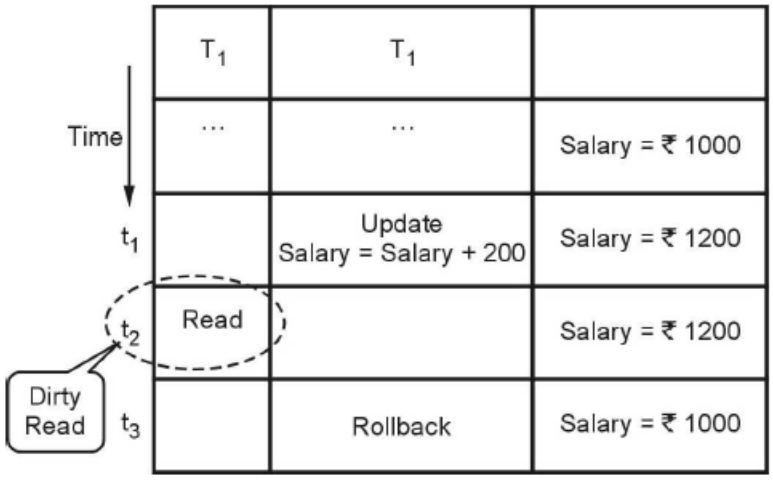


- The result of the above sequence is that the update made by transaction T1 is completely lost. Therefor this problem is called as lost update problem.

(2) Dirty Read or Uncommitted Read Problem: The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction.



- For example – Consider following transactions -
- Assume initially salary is = ` 1000



(1) At the time t1, the transaction T2 updates the salary to `1200

(2) This salary is read at time t2 by transaction T1. Obviously, it is 1200

(3) But at the time t3, the transaction T2 performs Rollback by undoing the changes made by T1 and T2 at time t1 and t2.

(4) Thus, the salary again becomes = ` 1000. This situation leads to **Dirty Read or Uncommitted Read** because here the read made at time t2**(immediately after update of another transaction)** becomes a dirty read.

(3) Non-repeatable read Problem

- This problem is also known as **inconsistent analysis problem**. This problem occurs when a particular transaction sees two different values for the same row within its lifetime. **For example –**

Time	T_1	T_2	
t_1	Read		Salary = ₹ 1000
t_2		Update salary from ₹ 1000 to ₹ 1200	Salary = ₹ 1200
t_3		Commit	
t_4	Read		Salary = ₹ 1200

(1) At time t1, the transaction T1 reads the salary as ` 1000

(2) At time t2 the transaction T2 reads the same salary as ` 1000 and updates it to `1200

(3) Then at time t3, the transaction T2 gets committed.

(4) Now when the transaction T1 reads the same salary at time t4, it gets different value than what it had read at time t1. Now, transaction T1 cannot repeat its reading operation. Thus, inconsistent values are obtained. Hence the name of this problem is non-repeatable read or inconsistent analysis problem.

(4) Phantom read Problem

- The phantom read problem is a special case of non-repeatable read problem.
- This is a problem in which one of the transactions makes the changes in the database system and due to these changes, another transaction cannot read the data item which it has read just recently.
- **For example –**

Time	T_1	T_2	
t_1	Read		Salary = ₹ 1000
t_2		Read	Salary = ₹ 1000
t_3	Delete salary		No salary
t_4		Read	"Can not find salary"

(1) At time t1, the transaction T1 reads the value of salary as 1000

(2) At time t2, the transaction T2 reads the value of the same salary as 1000

(3) At time t3, the transaction T1 deletes the variable salary.

(4) Now at time t4, when T2 again reads the salary, it gets error. Now transaction T2 cannot identify the reason why it is not getting the salary value which is read just few times back.

- This problem occurs due to changes in the database and is called phantom read problem.

14. State and explain the lock-based concurrency control with suitable example. Locking Protocols (APR/MAY 2024).

Why Do we Need Locks?

- One of the methods to ensure the **isolation** property in transactions is to require that data items be accessed in a mutually exclusive manner. That means, while one transaction is accessing a data item, no other transaction can modify that data item.
- The most common method used to implement this requirement is to allow a transaction to access a data item only if it is currently holding a **lock on that item.**
- Thus, the lock on the operation is required to ensure the isolation of transaction.

Simple Lock Based Protocol

Concept of Protocol: The lock-based protocol is a mechanism in which there is exclusive use of **locks** on the data item for current transaction.

Types of Locks: There are two types of locks used –

i) Shared Lock: The shared lock is used for reading data items only. It is denoted by **Lock-S**. This is also called as **read lock.**

ii) Exclusive Lock: The exclusive lock is used for both read and write operations. It is denoted as **Lock-X.** This is also called as **write lock.**

- The **compatibility matrix** is used while working on set of locks. The **concurrency control manager** checks the compatibility matrix before granting the lock. If the two modes of transactions are compatible to each other then only the lock will be granted.
- In a set of locks may consists of shared or exclusive locks. Following matrix represents the compatibility between modes of locks.

	S	X
S	T	F
X	F	F

- Here T stands for True and F stands for False. If the control manager gets the compatibility mode as True then it grants the lock otherwise the lock will be denied.
- **For example:** If the transaction T1 is holding a shared lock in data item A, then the control manager can grant the shared lock to transaction T2 as compatibility is True. But it cannot grant the exclusive lock as the compatibility is false. In simple words if transaction T1 is reading a data item A then same data item A can be read by another transaction T2 but cannot be written by another transaction.
- Similarly, if an exclusive lock (i.e. lock for read and write operations) is hold on the data item in some transaction then no other transaction can acquire Shared or exclusive lock as the compatibility function denotes F. That means of some transaction is writing a data item A then another transaction cannot read or write that data item A.
- Hence the **rule of thumb is**

  i) **Any number** of transactions can hold **shared lock** on an item.

  ii) But **exclusive lock can be hold by only one transaction**.
- **Example of a schedule denoting shared and exclusive locks:** Consider following schedule in which initially A=100. We deduct 50 from A in T1 transaction and read the data item A in transaction T2. The scenario can be represented with the help of locks and concurrency control manager as follows:

Exclusive Lock

Shared Lock

T_1	T_2	Concurrency control manager
Lock-X(A)		
		Grant X(A,T1) because in T1 there is write operation.
R(A)		
A=A-50		
W(A)		
Unlock(A)		
	Lock-S(A)	
		Grant S(A,T2) because in T2 there is Read operation
	R(A)	
	Unlock(A)	

15. Illustrate two phase locking protocol with an example. or Explain in detail about Two-phase locking and Timestamp based protocol in concurrency control. (APR/MAY 2023) (NOV/DEC 2022)

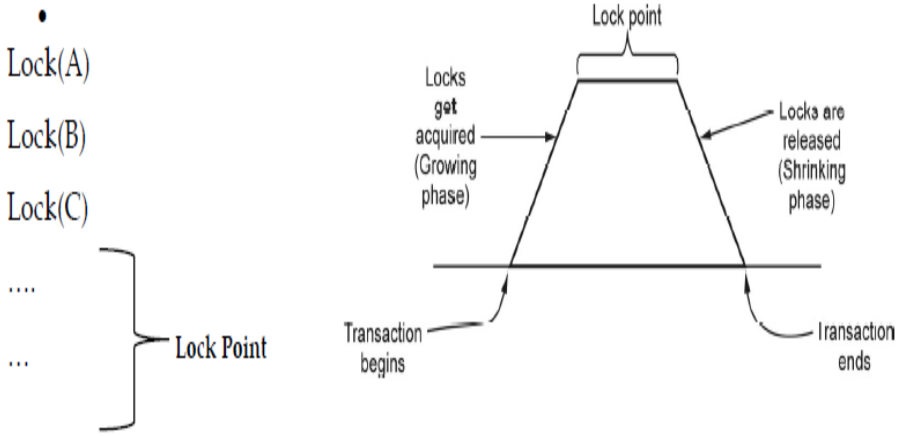
Two Phase Locking

The Two-phase locking is a protocol in which there are two phases:

i) Growing phase (Locking phase): It is a phase in which the transaction may obtain locks but does not release any lock.

ii) Shrinking phase (Unlocking phase): It is a phase in which the transaction may release the locks but does not obtain any new lock.

Lock Point: The **last lock position** or **first unlock position** is called lock point. For example,



Unlock(A)

Unlock(B)

Unlock(C)

For example – Consider following transactions

T1	T2
Lock-X(A)	Lock-S(B)
Read(A)	Read(B)
A=A-50	Unlock-S(B)
Write(A)	
Lock-X(B)	
Unlock-X(A)	
B=B+100	Lock-S(A)
Write(B)	Read(A)
Unlock-X(B)	Unlock-S(A)

- The important rule for being a two-phase locking is - **All Lock operations precede all the unlock operations.**
- In above transactions T1 is in two phase locking mode but transaction T2 is not in two phase locking.
- Because in T2, the Shared lock is acquired by data item B, then data item B is read and then the lock is released.
- Again, the lock is acquired by data item A, then the data item A is read and the lock is then released.
- Thus, we get lock-unlock-lock-unlock sequence. Clearly this is not possible in two phase locking.

16. Prove that two phase locking guarantees serializability. Or Explain in detail about two phase locking and how does it guarantees serializability. (NOV/DEC 2023)

Solution:

- Serializability is mainly an issue of handling write operation. Because any inconsistency may only be created by **write** operation.
- Multiple reads on a database item can happen parallelly.
- 2-Phase locking protocol restricts this unwanted read/write by applying **exclusive lock**.

- Moreover, when there is an **exclusive lock** on an item it will **only be released in shrinking phase**. Due to this restriction, there is no chance of getting any inconsistent state.
- The serializability using two phase locking can be understood with the help of following example
- Consider two transactions

T_1	T_2
R(A)	
	R(A)
R(B)	
W(B)	

Step 1: Now we will **apply two phase locking**. That means we will **apply locks in growing** and **shrinking** phase

T_1	T_2
Lock-S(A)	
R(A)	
	Lock-S(A)
	R(A)
Lock-X(B)	
R(B)	
W(B)	
Unlock-X(B)	
	Unlock-S(A)

- Note that above schedule is serializable as it prevents interference between two transactions.
- The serializability order can be obtained based on the **lock point**. The lock point is either last lock operation position or first unlock position in the transaction.
- The last lock position is in T1, then it is in T2. Hence the serializability will be T1->T2 based on lock points. Hence, the **serializability sequence** can be **R1(A); R2(A); R1(B); W1(B)**

Limitations of Two-Phase Locking Protocol

- The two-phase locking protocol leads to two problems – deadlock and cascading roll back.
- **Deadlock:** The deadlock problem cannot be solved by two phase locking. Deadlock is a situation in which when two or more transactions have got a lock and waiting for other locks currently held by one of the other transactions.
- For example,

T1	T2
Lock-X(A)	Lock-X(B)
Read(A)	Read(B)
A=A-50	B=B+100
Write(A)	Write(B)
Delayed, wait for T2 to release Lock on B	Delayed, wait for T1 to release Lock on A

(2) Cascading Rollback: Cascading rollback is a situation in which a single transaction failure leads to a series of transaction rollback. For example –

T1	T2	T3
Read(A)		
Read(B)		
C=A+B		
Write(C)		
	Read(C)	
	Write(C)	
		Read(C)

- When T1 writes value of C then only T2 can read it. And when T2 writes the value of C then only transaction T3 can read it. But if the transaction T1 gets failed then automatically transactions T2 and T3 gets failed.
- The simple two-phase locking does not solve the cascading rollback problem. To solve the problem of cascading Rollback two types of two-phase locking mechanisms can be used.

17. Explain the Types of Two-Phase Locking in detail.

(1) Strict Two-Phase Locking:

- The strict 2PL protocol is a basic two-phase protocol but **all the exclusive mode locks** be **held until the transaction commits**.
- That means in other words all the **exclusive locks** are **unlocked** only after the **transaction is committed**.
- That also means that if T1 has exclusive lock, then T1 will release the exclusive lock only after commit operation, then only other transaction is allowed to read or write.
- **For example –**
- Consider two transactions

T_1	T_2
W(A)	
	R(A)

- If we apply the locks then

T_1	T_2
Lock-X(A)	
W(A)	
Commit	
Unlock(A)	
	Lock-S(A)
	R(A)
	Unlock-S(A)

- Thus, only after commit operation in T1, we can unlock the exclusive lock.

- This ensures strict serializability.
- Thus, compared to basic two-phase locking protocol, the advantage of strict 2PL protocol is it ensures strict serializability.

Rigorous Two-Phase Locking:

- This is stricter two-phase locking protocol. Here all locks are to be held until the transaction commits. The transactions can be serialized in the order in which they commit.
- **For example –**
- Consider transactions

T_1
R(A)
R(B)
W(B)

- If we apply the locks then

T_1
Lock-S(A)
R(A)
Lock-X(B)
R(B)
W(B)
Commit
Unlock(A)
Unlock(B)

- Thus, the above transaction uses rigorous two-phase locking mechanism.

18. Consider the following two transactions:

T1: read(A)

Read(B);

If A=0 then B=B+1;

Write(B)

T2: read(B); read(A)

If B=0 then A=A+1

Write(A)

Add lock and unlock instructions to transactions T1 and T2, so that they observe two-phase locking protocol. Can the execution of these transactions result in deadlock?

Solution:

T1	T2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Read(B)	Read(A)
if A=0 then B=B+1	if B=0 then A=A+1
Write(B)	Write(A)
Unlock(A)	Unlock(B)
Commit	Commit
Unlock(B)	Unlock(A)

- This is lock-unlock instruction sequence help to satisfy the requirements for strict two-phase locking for the given transactions.
- The execution of these transactions result in deadlock. Consider following partial execution scenario which leads to deadlock.

T1	T2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Now it will wait for T2 to release exclusive lock on A	Now it will wait for T1 to release exclusive lock on B

19. Write about Lock Conversion with an example.

- Lock conversion is a mechanism in two phase locking mechanism - which allows conversion of shared lock to exclusive lock or exclusive lock to shared lock.

Method of Conversion:

First Phase:

- can acquire a lock-S on item
- can acquire a lock-X on item
- can convert a lock-S to a lock-X **(upgrade)**

Second Phase:

- can release a lock-S
- can release a lock-X
- can convert a lock-X to a lock-S **(downgrade)**

- This protocol assures serializability. But still relies on the programmer to insert the various locking instructions.
- For example – Consider following two transactions –

T_1	T_2
R(A)	R(A)
R(B)	R(B)
...	
R(C)	
...	
W(A)	

- Here, if we start applying locks, then we must apply the **exclusive lock** on data item A, because we have to read as well as write on data item A.
- Another transaction T2 does not get shared lock on A until transaction T1 performs write operation on A.
- Since transaction T1 needs exclusive lock only at the end when it performs write operation on A, it is better if T1 could initially lock A in shared mode and then later change it to exclusive mode lock when it performs write operation. In such situation, the lock conversion mechanism becomes useful.

- When we convert the shared mode lock to exclusive mode then it is called **upgrading** and when we convert exclusive mode lock to shared mode then it is called **downgrading**.
- Also note that upgrading takes place only in **growing phase** and downgrading takes place only in shrinking phase.
- Thus, we can refine above transactions using lock conversion mechanism as follows –

T1	T2
Lock-S(A)	
R(A)	
	Lock-S(A)
	R(A)
Lock-S(B)	
	Lock-S(B)
	R(B)
	Unlock(A)
	Unlock(B)
…	
Lock-S(C)	
R(C)	
…	
Upgrade(A)	
W(A)	
Unlock(A)	
Unlock(B)	
Unlock(C)	

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai,

Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / III SEM

UNIT V– OBJECT RELATIONAL AND NO-SQL DATABASES

SYLLABUS:

Mapping EER to ODB schema - Object identifier-reference types-row types-UDTs-Subtypes and supertypes user-defined routines- Collection types - Object Query Language, No-SQL: CAP theorem -Document-based: MongoDB data model and CRUD operations: Column-based: Hbase data model and CRUD operations.

PART A

1. What is object identifier (OID)?

- Any real-world entity is modelled as object. Associated with each object there is a unique ID maintained which is called object identifier (OID).

2. Explain the structure of object

- Object is a fundamental unit of object-oriented programming in which data var and code is encapsulated in a single unit.
- Conceptually all interactions among the objects and rest of the system are carried out by using messages.

3. What is reference type?

- Reference types are all types other than value types. Variables of reference types store a reference to the memory address of the value.
- A reference type value can be stored in one table and used as a reference to specific row in some other table.

PREPARED BY: Mrs. S. JAYABHARATHI AP/AI&DS

4.What is row type?

- A row type is a sequence of name-datatype pair. It is used to provide a datatype to the rows in the table.
- Using row type complete row can be stored in a variable.

5. Compare between supertype and subtype. (APR/MAY 2023)

Supertype	Subtype
It an entity type that has got the relationship as parent to child with one or more subtypes.	The subtypes are group of subtype entity.
It contains attributes that are common to its subtypes.	It contains unique attributes but they are different from each subtype.

6. What is NoSQL?

- NoSQL stands for not only SQL. It is nontabular database system that store data differently than relational tables.
- There are various types of NoSQL databases such document, key-value, wide column and graph.
- Using NoSQL, we can maintain flexible schemas and these schemas can be scaled easily with large amount of data.

7.What is CAP theorem? or State the CAP theorem in NoSQL. (APR/MAY 2023)

- The CAP theorem states that it is not possible to guarantee all three of the desirable properties Consistency, availability and partition tolerance at the same time in a distributed system with data replication.

8.Enlist the features of MongoDB.

- It is a schema-less, document-based database system.
- It provides high performance data persistence.
- It supports multiple storage engines.
- It has a rich query language support.
- MongoDB provides high availability and redundancy with the help of replication. That means it creates multiple copies of the data and sends these

copies to a different server so that if one server fails, then the data is retrieved from another server.

9.Enlist the features of NoSQL.

- The NoSQL does not follow any relational mode
- It is either schema free or have relaxed schema. That means it does not require specific definition of schema.
- Multiple NoSQL databases can be executed in distributed fashion.
- It can process both unstructured and semi-structured data
- The NoSQL have higher scalability.

10.What is MongoDB?

- MongoDB is an open source, document-based database.
- It is developed and supported by a company named 10gen which is now known MongoDB Inc.
- The first ready version of MongoDB was released in March 2010.

11.Why MongoDB is needed?

- There are so many efficient RDBMS products available in the market, then why do we need MongoDB? Well, all the modern applications require big data, faster development and flexible deployment.
- This need is satisfied by the document-based database like MongoDB.

12.How the terms in MongoDB are different from SQL?

- The terms in SQL are treated differently in MongoDB.
- In MongoDB the data is not stored in tables, instead of that, the there is a concept called collection which is analogous to the tables.
- In the same manner the rows in RDBMS are called documents in MongoDB, likewise the columns of the record in RDBMS are called fields.

13.Define collections in MongoDB.

- MongoDB groups data together through collections. A collection is simply a grouping of documents that have the same or a similar purpose.
- A collection acts similarly to a table in a traditional SQL database, with one major difference.

- In MongoDB, a collection is not enforced by a strict schema; instead, documents in a collection can have a slightly different structure from one another as needed.
    - This reduces the need to break items in a document into several different tables, which is often done in SQL implementations.

14.Define the term document in MongoDB.

- A document is a representation of a single entity of data in the MongoDB database.
- A collection is made up of one or more related objects.
- A major difference between MongoDB and SQL is that documents are different from rows.
- Row data is flat, meaning there is one column for each value in the row.
- However, in MongoDB, documents can contain embedded subdocuments, thus providing a much closer inherent data model to your applications.

15.Enlist any four data types in MongoDB.

- Following are various types of data types supported by MongoDB.

1. Integer: This data type is used for storing the numerical value.

2. Boolean: This data type is used for implementing the Boolean values i.e. true or false.

3. Double: Double is used for storing floating point data.

4. String: This is the most commonly used data type used for storing the string values.

16.Explain how to create collection in MongoDB?

- After creating some sample database, we must create some collection inside that database.
- **Syntax**

  db.create Collection(name, options) where
- We can create collection explicitly using createCollection command name is the name of collection options is an optional field. This field is maximum number of documents and so on.

17.Define CRUD Operation

- After creating some sample database, we must create some collection inside that database.
- We can perform various operations such as insert, delete and update on this collection.
- These operations are called CRUD operations.
- CRUD stands for Create, Read. Update and Delete operation.

18.How to create Create Collection in MongoDB

Syntax

- We can create collection explicitly using createCollection command. db.create Collection(name, options)
- where name is the name of collection
- options are an optional field.
- This field is used to specify some parameters such as Size, maximum number of documents and so on.

19.Give the syntax for insert the document. The document is analogous to rows in database.

- The document is inserted within the collection. The document is analogous to rows in database
- **Syntax**

  db.collection name.insert({key, value }

20.How to insert multiple documents in a MongoDB database?

- It is possible to insert multiple documents at a time using a single command. Following command shows how to insert multiple documents in the existing collection

```
> var = allStudents = [
        {
"name": AAA,
'age": 20
},
{
“name”:BBB,
```

```
"age": 21
},
{
"name": CCC "age": 22
}
];
>db.Student
>db.Student details.insert(alliStudents)
```

21.How to delete Documents in MongoDB?

- For deleting the document, the remove command is used. This is the simplest command
- **Syntax**

  db.collection_name.remove(delete_criteria)

22.What are the methods used in Update the document in MongoDB?

- The methods used in Update the document in MongoDB

  1.updateOne,

  2. updateMany or bulkwrite.

23. What is column-oriented database model? Give example of it.

- The column-oriented database is a database system that stores the data table by column rather than by rows.
- The advantage of column-oriented Database is that we can access the data more efficiently when only subset of columns is used for querying.
- **For example -** HBase

24. Enlist the features of HBase data model.

- HBase is horizontally scalable. That means we can add any number of columns anytime.
- The data is stored in key value format.
- It is a platform for storing and retrieving data with random access.
- It does not enforce relationship within the data.
- It is designed to run on cluster of computers.

25. What are the advantages of NoSQL? (NOV/DEC 2023)

- Scalability
- Simplicity
- Less code
- Easy code

26. What are the advantages of Object-oriented model? (NOV/DEC 2023)

- Easy to save and retrieve data quickly.
- Complex data and wider variety of data types compared to MySQL datatypes.
- Code reusability.
- Improved reliability and flexibility.
- Real world modeling.

27.State denormalization. (APR/MAY 2024)

- Denormalization is a database optimization technique in which we add redundant data to one or more tables.
- It is an optimization technique that is applied after normalization.

28.Distinguish between total and partial rollback. (APR/MAY 2024)

TOTAL ROLLBACK	**PARTIAL ROLLBACK**
Abort the transaction and then restart it.	More effective to roll back transaction only as far as necessary to break deadlock.

Part B & C Questions

1.Explain about Object Database Concept and Mapping EER and ODB Schema. Or explain in detail about Object relational data model with diagram. (APR/MAY 2023) OR Explain mapping an EER schema to an ODB schema in detail. (NOV/DEC 2023) OR Describe the features of object-oriented data model. (APR/MAY 2024)

Object Database Concept

- Traditional applications are designed for business applications such as inventory, employee, university, bank, library, air-line reservation systems, and so on.
- These applications require relatively simple data types that are well suited to the relational data model.
- But database systems were applied to a wider range of applications, such as computer-aided design and geographical information systems.
- Hence such systems require complex data type and processing.
- The object-based database provides the solution to model the real-world object and their behaviour. It is an alternative to relational database model.

Mapping EER to ODB Schema

- Object oriented paradigm encapsulate data and corresponding methods in a single entity called object.
- The object-oriented data model is a logical data model just like ER model. Loosely object can be thought as an entity of E-R model.
- Object oriented data model helps in adapting object-oriented paradigm to database systems,
- In object-oriented database, information is represented in the form of objects.
- Object oriented databases are exactly same as object-oriented programming languages.
- If we can combine the features of relational model (transaction, concurrency, recovery) to object-oriented databases, the resultant model is called as object-oriented database model.
- The core object-oriented Data model consists of following object-oriented concepts-

Object and Object Identifier

- Any real-world entity is modelled as object. Associated with each object there is a unique ID maintained which is called object identifier (OID).

- Object identifiers used to uniquely identify objects. Object identifiers are unique. No two objects have the same identifier, each object has only one object identifier.

Attribute and method

- Every object has state and behaviour. The state is a set of values for attributes of the object.
- The behaviour is set of methods. **For example -** Object named circle has state radius and computing its area is a behaviour.

Class

- Class is a means of grouping all the objects which share the same set of attributes and methods.
- An object must belong to only one class as an instance of that class (instance-of relationship).
- A class is similar to an abstract data type.

Class Hierarchy and Inheritance

- The class hierarchy is used to represent the Inheritance property of object-oriented paradigm. The inheritance is a mechanism in which a new class is derived from existing class.

1.Object Structure

- Object is a fundamental unit of object-oriented programming in which data value and code is encapsulated in a single unit.
- Conceptually all interactions among the objects and rest of the system are carried out by using messages.
- Thus, the interface among various objects and rest of the system is messages. Messages can be implemented as procedure invocations.
    - In general, an object has associated with it:
    - A set of variables that contain the data for the object. The value of each variable is itself an object.
    - A set of messages to which the object responds.
- A set of methods, each of which is a body of code to implement each message; a method returns a value as the response to the message.
- Methods are programs written in general-purpose language with the following features-
    - Only variables in the object itself may be referenced directly.
    - Data in other objects are referenced only by sending messages.

- Methods can be read-only or update methods. Read-only methods do not change the value of the object.

- **For example**-the object Circle can have methods such as compute area)

2.Object Classes

- Similar objects are grouped into **classes**. In other words, object is an instance of a class. **For example,**

```
class Student {
/*variables*/
Int RollNo:
String Name:
String address;
/*Messages*/
String get name ();
String get_address();
Int get RollNo();
}
```

- Methods can be defined separately. **For example,**

```
int get roliNo()
{
Return RollNO;
}
```

3.Inheritance

- In object-oriented database schema, there can be large number of classes.
- Some classes are similar and can be derived from old existing classes.
- The classes are placed into specialization or Is-a hierarchy.
- **For example -** Consider following ER model as shown in figure 5.1 –

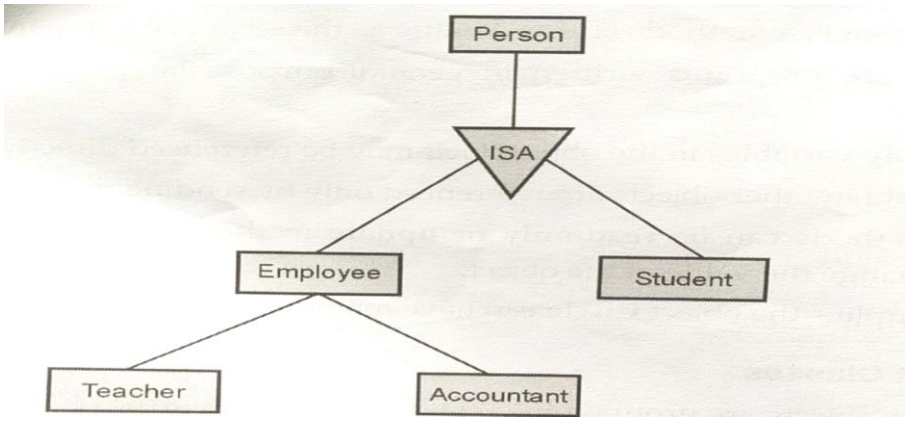


Figure 5.1 Inheritance

Class Hierarchy

- The class hierarchy can be defined in pseudo-code in which the variables associated with each class are as follows

```
class Person
{
String name;
String address;
}
class Student isa Person //Inheritance: Deriving child class
{
int RollNo;
String Course taken
}
class Employee isa Person
{
date join-date;
int salary;
}
class Teacher isa Employee
{
String Subject:
}
class Accountant isa Employee
{
int office no
}
```

- The keyword isa is used to indicate that a class is a specialization of another class.
- The specialization of a class is called subclasses. **For example** - Employee is a subclass of Person. Conversely, Person is a superclass of class Employee.
- Class hierarchy and inheritance of properties from more general classes.
- The important benefit of using inheritance is code-reusability. This can be achieved by means of substitutability property.
- **Substitutability** means any method of a class, say person, can be invoked equally well with any object belonging to any subclass, such as subclass Teacher of Person.

4.Multiple Inheritance

- In most cases, tree-structured organization of classes is adequate to describe applications.
- In such cases, all super classes of a class are ancestors of descendants of another in the hierarchy.
- However, there are situations that cannot be represented well in a tree-structured class hierarchy.
- To avoid the conflicts between two occurrences, we use multiple inheritance.
- Multiple inheritance is an ability of class to inherit variables from multiple super classes.
- The class subclass relationship is represented by directed acyclic graph (DAG) as shown in figure 5.2. **For example-**

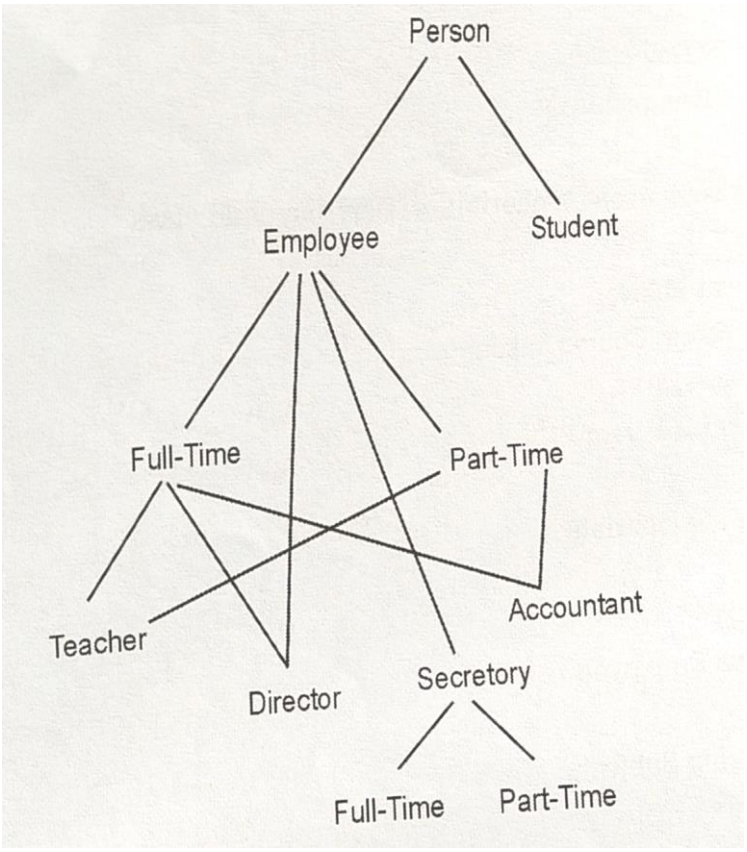


Figure 5.2 Multiple inheritance

- Handling name conflicts: When multiple inheritance is used, there is potential ambiguity if the same variable or method can be inherited from more than one superclass.

5. Object Identity

- For identifying the RDBMS record uniquely in the table, we use primary key associated with it.

- In object-oriented database management system, we use Object Id to identify the record.
- Associated with each object there is a unique ID maintained which is called object identifier (OID).
- An object retains its identity even if some or all the values of the variables or definitions of methods change over time.
- The concept of object identity does not apply to tuples of relational database. It is a stronger notion of identity.
- The difference between primary of RDBMS and object ID of OODBMS is that primary key is explicit; that means it is visible to the user whereas the object ID is implicit; that means it is hidden from external world.
- It has several forms of identity:
    - **Value:** The data value is used for identity.

    For example, the primary key of the tuple in a relational database.
    - **Name:** The user supplied name is used for identity. **For example,** file name can be used for identity of object.
    - **Built-in:** A notion of identity is built-into the data model or programming languages, and no user-supplied identifier is required.
- Object identity is typically implemented via a unique, system-generated OID. The value of the OID is not visible to the external user, but is used internally by the system to identify each object uniquely and to create and manage inter-object references.

6. Object Containment

- The objects that contain other objects are called composite objects or complex objects.
- There can be multiple levels of object containment. These levels can be represented using containment hierarchy.
- **For example,** the document contains the text in the form of paragraphs. Each paragraph contains word. Each word is made up of come characters.
- Also note that there can be some size and style for each paragraph, word and each character.
- The links between classes must be interpreted as is-part-of, rather than the is-a interpretation of links in an inheritance hierarchy.
- Containment allows data to be viewed at different granularities by different users. **For example -** document can be viewed by a reader for simply reading

purpose, the size and style of paragraphs can be viewed by DTP operator or proof editor for editing purpose.

- The containment hierarchy can be used to find all object contained in document object.
- In certain applications, an object may be contained in several objects. In such cases, the containment relationship is represented by a DAG rather than by a hierarchy.

2.Write a short note about

a) Reference types

b) Row Types

c) User Defined Types

d) Subtypes

e) Supertypes

Reference types

- Reference types are all types other than value types. Variables of reference types store a reference to the memory address of the value.
- A reference type value can be stored in one table and used as a reference to specific row in some other table.
- To use a reference type in an SQL statement, use REF (type-name), where type-name represents the referenced type.
- REF IS SYSTEM GENERATED in a CREATE TYPE statement indicates that the actual values of associated REF type are provided by the system.

Row Types

- A row type is a sequence of name-datatype pair. It is used to provide a datatype to the rows in the table.
- Using row type complete row can be stored in a variable. This variable can be passed as an argument to the routine or we can get the returned value from a function call within this variable.
- **For example,** Consider an Employee table containing Emp_Id and address and insert into the new table

CREATE TABLE Employee (empID CHAR (5).
address ROW (Street VARCHAR (30),
City VARCHAR (25).
Pincode ROW (city id VARCHAR (6). Region VARCHAR (10))));

INSERT INTO Employee VALUES ('Account 1234', ROW ('11 Shankar Rd', "Chennai", ROW ('ch11','Anand Nagar")));

User-defined type

- The user defined type (UDT) is a data type derived from an existing data type
- UDT allows the database developer to extend the built in types and to create a customized data type.
- There are two types of user defined types-

Distinct Type

- A distinct type is a user-defined data type that shares its internal representation with an existing built-in data type. For example-
  CREATE TYPE empNumberType AS VARCHAR (5) FINAL

Structured Type

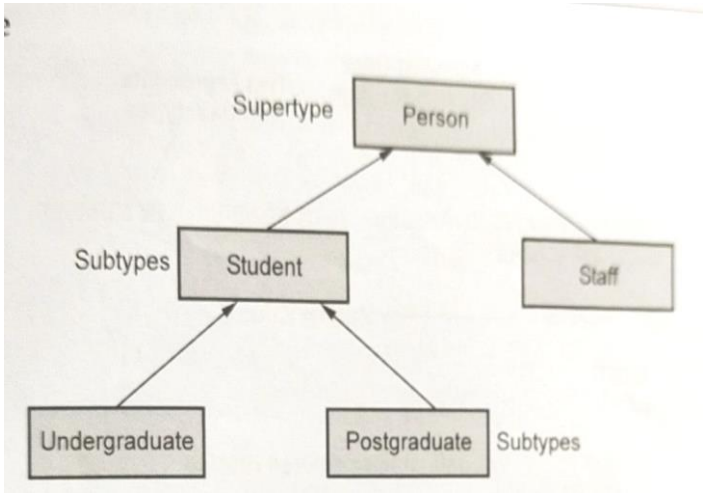
- A structured type is a user-defined data type containing one or more named attributes, each of which has a data type.
- Attributes are properties that describe an instance of a type.
- A structured type also includes a set of method specifications. Methods enable you to define behaviours for structured types. For example -
  CREATE TYPE person AS OBJECT (
  name VARCHAR (30),
  phone VARCHAR (20));
  CREATE TYPE purchase_order AS OBJECT (
  id NUMBER, contact person,
  MEMBER FUNCTION
  get value RETURN NUMBER);
- In above example we have defined an object named person and created a structure type purchase_order. The members of this structure are attributes such as id, contact and one member function named get_value
- The value of the attribute can be accessed using the dot ooperator. **For example,** if p is an instance of person, then,
  p.name = 'Anand'
- allows to assign name to the attribute of the person data type.

Supertype:

- It an entity type that has got the relationship as parent to child with one or more subtypes. It contains attributes that are common to its subtypes.

Subtype:

- The subtypes are group of subtype entity and have unique attributes but they are different from each subtype.



Figureure 5.3 Example of Subtype and Super type

- To create a subtype **StudentType** of the **supertype** PersonType we write:

```
CREATE TYPE StudentType UNDER PersonType AS (
rollNo VARCHAR (5),
name VARCHAR (10),
branchID CHAR (4))
```

Advantages of Subtype and super type

1. Avoids data redundancy, as each subtype contains only unique information. This in turn leads to a reduction in the size of the database itself.
2. Reduction of errors in programming operations on a table that corresponds to a specific subtype of an entity.
3. Flexibility in modifying the database structure. There is no need to modify the super type table if you add / change the structure of the subtype table.

3. Write about User-Defined Routines and collection types in detail.

User-Defined Routines

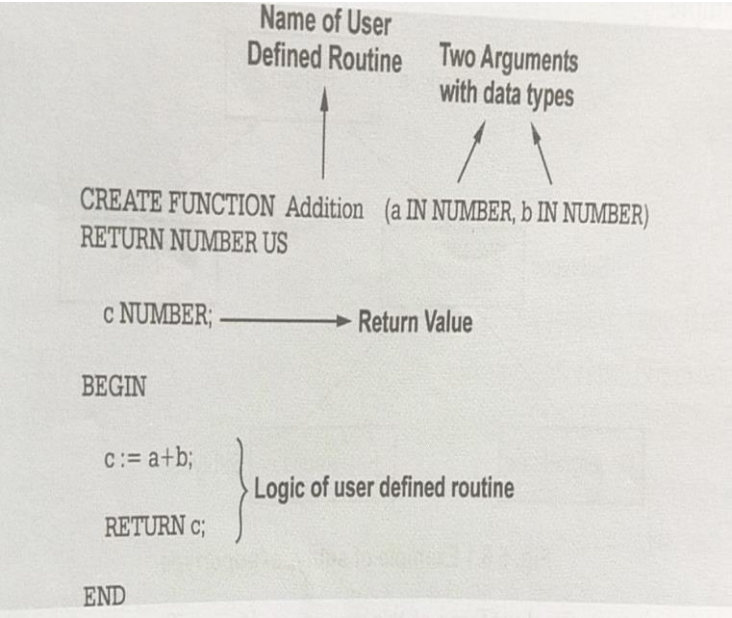
- User-defined routines are routines that users create themselves. User-defined routines provide a means for users to extend the SQL language.
- The User Defined routines capture the functionality of most commonly used arithmetic, string, and casting functions.
- However, these also allows to create routines to encapsulate logic of your own.

- User-defined procedures, functions and methods are created in the database by executing the appropriate CREATE statement for the routine type. These routine creation statements include:

  CREATE PROCEDURE

  CREATE FUNCTION

  CREATE METHOD



- The clauses specific to each of the CREATE statements define characteristics of the routine, such as the routine name, the number and type of routine arguments, and details about the routine logic.

Collection Types

- A collection is an ordered group of elements having the same data type.
- The collections are used to store multiple values in a single column of table.
- The collection add flexibility to the database design.
- Following are the ways by which the collection types can be used-
  1. **ARRAY:** It is an entity in that represents one dimensional array with maximum number of elements
  2. **LIST:** Ordered collection that allows duplicates.
  3. **SET:** Unordered collection that does not allow duplicates. **For example**-Array can be created as follows-

  VARCHAR (20) ARRAY (4):

  ARRAY ['Archana', 'Ashwini', 'Aishwarya', "Sharda']:

4. Explain Object Query Language (OQL) in Detail.

Object Query Language (OQL)

- Object Query Language (OQL) is a query language standard for object-oriented databases modelled after SQL.
- The following rules apply to OQL statements

  (1) All complete statements must be terminated by a semi-colon.

  (2) A list of entries in OQL is usually separated by commas but not terminated by a comma (,).

  (3) Strings of text are enclosed by matching quotation marks.
- Basic From of OQL: SELECT, FROM, WHERE
- **Syntax**

  SELECT <list of values>

  FROM <list of collections and variable assignments>

  WHERE <condition>
- Where the SELECT clause extracts those elements of a collection meeting a specific condition.
- By using the keyword DISTINCT duplicated elements in the resulting collection get eliminated.
- Collections in FROM can be either extents (persistent names sets) or expressions that evaluate to a collection (a set).
- **Example:** Give the names of people who are older than 30 years old:

  SELECT SName: p.name

  FROM p in People

  WHERE page > 30

DOT notations and Path expressions

- We use the dot notation and path expressions to access components of complex values. **For example,**

  ta.salary -> real

  t.students-> set of tuples of type tuple(name: string, fee: real) representing students

  t.salary-> real

5. What is NoSQL? What is the need for it. Enlist various feature of NoSQL. Also, Explain different types of NoSQL databases.

NoSQL

- NoSQL stands for not only SQL.
- It is nontabular database system that store data differently than relational tables.
- There are various types of NoSQL databases such as document, key-value, wide column and graph.
- Using NoSQL, we can maintain flexible schemas and these schemas can be scaled easily with large amount of data.

Need for NoSQL

- The NoSQL database technology is usually adopted for following reasons-

  1) The NoSQL databases are often used for handling big data as a part of fundamental architecture.

  2) The NoSQL databases are used for storing and modelling structured, semi-structured and unstructured data.

  3) For the efficient execution of database with high availability, NoSQL. is used.

  4) The NoSQL database is non-relational, so it scales out better than relational databases and these can be designed with web applications.

  5) For easy scalability, the NoSQL is used.

Features of NoSQL

1) The NoSQL does not follow any relational model.

2) It is either schema free or have relaxed schema. That means it does not require specific definition of schema.

3) Multiple NoSQL databases can be executed in distributed fashion.

4) It can process both unstructured and semi-structured data.

5) The NoSQL have higher scalability.

6) It is cost effective.

7) It supports the data in the form of key-value pair, wide columns and graphs.

Types of NoSQL:

- There are four types of NoSQL databases and those are-

  1. Key-value store
  2. Document store
  3. Graph based
  4. Wide column store

- Let us discuss them in detail.

1. Key-Value Store

- Key-value pair is the simplest type of NoSQL database.
- It is designed in such a way to handle lots of data and heavy load.
- In the key-value storage the key is unique and the value can be JSON, string or Binary objects.
- **For example-**

  {
  Customer:
  [
  {"id":1,"name":"Ankita"),
  {"id":2,"name":"Kavita"}
  ]
  }
- Here id, name are the keys and 1,2, "Ankita", "Prajkta" are the values corresponding to those keys.
- Key-value stores help the developer to store schema-less data.
- They work best for Shopping Cart Contents.
- The DynamoDB, Riak, Redis are some famous examples of key-value store

2. Document Store

- The document store makes use of key-value pair to store and retrieve data.
- The document is stored in the form of XML and JSON.
- The document stores appear the most natural among NoSQL. database types.
- It is most commonly used due to flexibility and ability to query on any field.
- **For example -**

  {
  "id": 101,
  "Name": "AAA",
  "City": "Pune"
  }
- MongoDB and CouchDB are two popular document-oriented NoSQL database

3. Graph

- The graph database is typically used in the applications where the relationships among the data elements is an important aspect.

- The connections between elements are called links or relationships.
- In a graph database, connections are first-class elements of the database, stored directly.
- In relational databases, links are implied, using data to express the relationships.
- The graph database has two components as shown in figure 5.4.

  1**) Node:** The entities itself. For example - People, student,

  2) Edge: The relationships among the entities.
- **For example-**

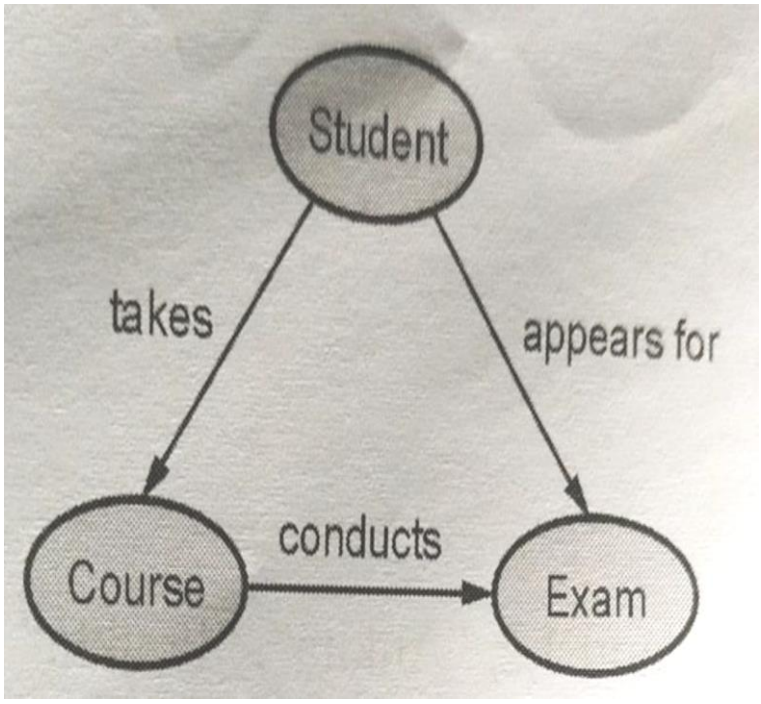


Figure 5.4 Graph

- Graph base database is mostly used for social networks, logistics, spatial data.
- The graph databases are Neo4J, Infinite Graph, OrientDB.

4. Wide Column Store

- Wide column store model is similar to traditional relational database.
- In this model, the columns are created for each row rather than having predefined by the table structure.
- In this model number of columns are not fixed for each record.
- Columns databases can quickly aggregate the value of a given column.

- **For example-**

Row ID	Columns...		
1	**Name**	**City**	
	Ankita	Pune	
2	**Name**	**City**	**email**
	Kavita	Mumbai	kavita123@gmail.com

- The column store databases are widely used to manage data warehouses, business intelligence, HBase, Cassandra are examples of column-based databases.

6. Give the difference between RDBMS and NoSQL.

RDBMS	**NoSQL**
The relational database system is based on relationships among the tables.	It is non-relational database system. It can be used in distributed environment
It is vertically scalable.	It is horizontally scalable.
It has predefined schema.	It does not have schema or it may have related schema.
It uses SQL to query the database.	It uses unstructured query language.
It is a table-based database.	It is document-based; graph based on key-value pair.
It emphasizes on ACID properties (Atomicity, consistency, isolation and durability)	It follows Brewers CAP theorem (Consistency, availability and partition tolerance)
Schema is fixed or rigid.	Schema is dynamic
Pessimistic.	Optimistic
Examples: MySQL, Oracle, PostgreSQL	**Examples:** MongoDB, Big Table, Redis

7. Write a short note on CAP Theorem

CAP Theorem

- Cap theorem is also called as brewer's theorem.
- The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores:
    1. **Consistency:** All reads receive the most recent write or an error.
    2. **Availability:** All reads contain data, but it might not be the most recent
    3. **Partition tolerance:** The system continues to operate despite network failures (i.e.; dropped partitions, slow network connections, or unavailable network connections between nodes.)
    - The CAP theorem states that it is not possible to guarantee all three of the desirable properties - consistency, availability, and partition tolerance at the same time in a distributed system with data replication.

8. Understanding NoSQL and MongoDB in Detail

NoSQL

- NoSQL stands for not only SQL.
- It is nontabular database system that store data differently than relational tables.
- There are various types of NoSQL databases such as document, key-value, wide column and graph.
- Using NoSQL, we can maintain flexible schemas and these schemas can be scaled easily with large amount of data.

Need for NoSQL

- The NoSQL database technology is usually adopted for following reasons-

    1) The NoSQL databases are often used for handling big data as a part of fundamental architecture.

    2) The NoSQL databases are used for storing and modelling structured, semi- structured and unstructured data.

    3) For the efficient execution of database with high availability, NoSQL. is used.

    4) The NoSQL database is non-relational, so it scales out better than relational databases and these can be designed with web applications.

    5) For easy scalability, the NoSQL is used.

Features of NoSQL

1) The NoSQL does not follow any relational model.

2) It is either schema free or have relaxed schema. That means it does not require specific definition of schema.

3) Multiple NoSQL databases can be executed in distributed fashion.

4) It can process both unstructured and semi-structured data.

5) The NoSQL have higher scalability.

6) It is cost effective.

7) It supports the data in the form of key-value pair, wide columns and graphs.

MongoDB

- MongoDB is an open source, document-based database.
- It is developed and supported by a company named 10gen which is now known MongoDB Inc.
- The first ready version of MongoDB was released in March 2010.

Why MongoDB is needed?

- There are so many efficient RDBMS products available in the market, then why do we need MongoDB?
- Well, all the modern applications require big data, faster development and flexible deployment.
- This need is satisfied by the document-based database like MongoDB.

Features of MongoDB

1. It is a schema-less, document-based database system.

2. It provides high performance data persistence.

3. It supports multiple storage engines.

4. It has a rich query language support.

5. MongoDB provides high availability and redundancy with the help of replication. That means it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.

6. MongoDB provides horizontal scalability with the help of sharding. Sharding means to distribute data on multiple servers.

7. In MongoDB, every field in the document is indexed as primary or secondary. Due to which data can be searched very efficiently from the database.

Understanding Collections

- MongoDB groups data together through collections. A collection is simply a grouping of documents that have the same or a similar purpose.

- A collection acts similarly to a table in a traditional SQL database, with one major difference.
- In MongoDB, a collection is not enforced by a strict schema; instead, documents in a collection can have a slightly different structure from one another as needed.
- This reduces the need to break items in a document into several different tables, which is often done in SQL implementations.

Understanding Documents

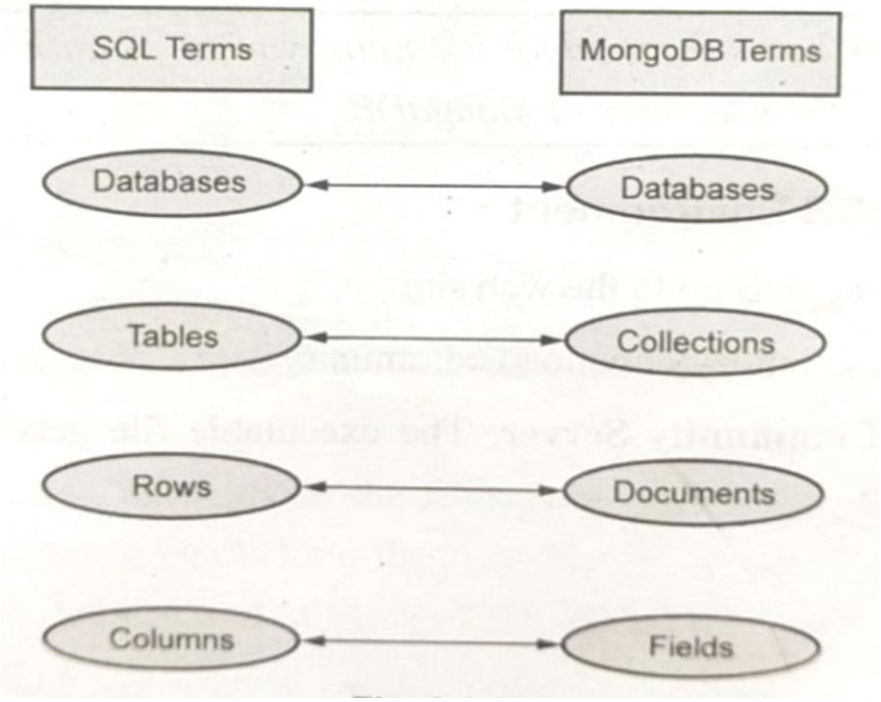
- A document is a representation of a single entity of data in the MongoDB database. A collection is made up of one or more related objects.
- A major difference between MongoDB and SQL is that documents are different from rows.
- Row data is flat, meaning there is one column for each value in the row. However, in MongoDB, documents can contain embedded subdocuments, thus providing a much closer inherent data model to your applications.
- The records in MongoDB that represent documents are stored as BSON, which is a lightweight binary form of JSON, with field: value pairs corresponding to JavaScript property: value pairs.
- These field: value pairs define the values stored in the document.
- That means little translation is necessary to convert MongoDB records back into the JavaScript object that you use in your Node.js applications.
- For example, a document in MongoDB may be structured similarly to the following with name, version, languages, admin, and paths fields:

```
{
name: "New Project",
version: 1,
languages: ["JavaScript", "HTML", "CSS"],
admin: {name: "Brad", password: "****"},
paths: {temp: "/tmp", project: "/opt/project", html:
"/opt/project/html"}
}
```

SQL Structure Vs. MongoDB

- Following Figure 5.5 shows the terms in SQL are treated differently in MongoDB. In MongoDB the data is not stored in tables, instead of that, there is a concept called collection which is analogous to the tables. In the same

manner the rows in RDBMS are called documents in MongoDB, likewise the columns of the record in RDBMS are called fields.



Figureure 5.5 SQL Structure Vs. MongoDB

Consider a student database as follows:

- To the left-hand side, we show the database in the form of table and to the right-hand side the database is shown in the form of collection,

Name	Course	AddressLine1	AddressLine2
Ankita	Computer Engineering	Lane1	Lane2
Prajkta	Mechanical Engineering	Street1	Street2
Kavita	Civil Engineering	Road1	Road2

Collection

```
{
  "_id":527f191e810c19349de860eg,
  "Name":"Ankita",
  "Course":"Computer Engineering",
  "Address":{
  "line1":"Lane1"
  "line2":"Lane2"
      }
},
{
  "_id":557f191e810c19729de860ea,
  "Name":"Prajkta",
  "Course":"Mechanical Engineering",
  "Address":{
  "line1":"Street1",
  "line2":"Street2"
      }
},
{
  "_id":507f141e510c19729dey60ed,
  "Name":"Kavita",
  "Course":"Civil Engineering",
  "Address":{
  "line1":"Road1",
  "line2":"Road2"
      }
}
```

Fields

Document 1

Document 2

Document 3

9. What are the data types used in MongoDB? Also, Write the MongoDB command to create and drop the database.

Data Types used in MongoDB

- Following are various types of data types supported by MongoDB.

  1. Integer: This data type is used for storing the numerical value.

  2. Boolean: This data type is used for implementing the Boolean values i.e. true or false.

  3. String: This is the most commonly used data type used for storing the string values

  4. Double: Double is used for storing floating point data.

  5. Min/Max keys: This data type is used to compare a value against the highest BSON element.

  6. Arrays: For storing an array or list of multiple values in one key, this data used.

  7. Object: The object is implemented for embedded documents.

  8. Symbol: This data type is similar to string data type. This data type is used specific symbol type.

  9. Null: For storing the null values this data type is used.

  10. Date: This data type is used to store current date or time. We can also own date or time object.

  11. Binary data: In order to store binary data, we need to use this data type.

  12. Regular expression: This data type is used to store regular expression.

Administering Databases.

- Write the MongoDB command to create and drop the database. operations.
- In this section we will discuss how to create and handle database in MongoDB using various commands.

(1) Create Database

For example,

- Open the command prompt and type the command mongos for starting the MongoDB shell.
- The test> prompt will appear. For creating a database, we need to “use” the command.
- **Syntax**

  Use Database_name

(2) Displaying all the databases present in the system

- We can see the list of databases present in the MongoDB using the command show dbs
- **For example,**
  mystudents> show dbs Admin 180.06 KiB
  configure 72.00 KiB
  Local 72.00 kiB mystudents>
- Note that in above listing we cannot see the mystudents database.
- This is because we have not inserted any document into it.
- To get the name of the database in the listing by means of show command, there should be some record present in the database.

(3) Drop Database

- The dropDatabase() command is used to delete the database. For example - Mystudents> db.dropDatabase()
  {ok: 1, dropped: 'mystudents'} mystudents>

10. Explain with suitable examples, CRUD operations in MongoDB. Managing Collections or Discuss in depth about MongoDB database and its CRUD operations with an example application. (APR/MAY 2023) OR Explain MongoDB data modelling in detail with real time example. (NOV/DEC 2023)

CRUD operations in MongoDB

- After creating some sample database, we must create some collection inside that database.
- We can perform various operations such as insert, delete and update on this collection.
- These operations are called CRUD operations. CRUD stands for Create, Read. Update and Delete operation.

1.Create Collection Syntax

- We can create collection explicitly using createCollection command. db.create Collection(name, options)

  where

  name is the name of collection

  options are an optional field. This field is used to specify some parameters such as Size, maximum number of documents and so on.
- Following is a list of such options.

Field	Type	Description
Capped	Boolean	Capped collection is a fixed size collection. It automatically overwrites the oldest entries when it reaches to maximum size. If it is set to true, enabled a capped collection. When you specify this value is true, you need to specify the size parameter.
autolndexID	Boolean	This field is required to create index id automatically. Its default value is false.
size	Number	This value indicates the maximum size in bytes for a clapped collection.
Max	Number	It specifies the maximum number of documents allowed in capped collection.

Table Options for create collection

- **For example** - Following command shows how to create collection in a database using explicit command

```
test> use myStudents Switched to db mystudents
myStudents> db.createCollection("Student_details")
{ok: 1} Mystudents>
```

2) Display Collection

- To check the created collection, use the command "show collections" at the command prompt.

```
myStudents> show collections Student_details
myStudents>
```

3) Drop Collection

- The drop collection command is actually used to remove the collection completely from the database. The drop command removes a collection completely from database.
- **Syntax**

  db.collection name.drop()
- **For Example,**

  my Students> db.Student_details.drop() true

  my Students>
- We can verify the deletion of the collection by using "show collections" in the database.

  myStudents> show collections myStudents>

4) Insert documents within the collection

- The document is inserted within the collection. The document is analogous to rows in database
- **Syntax**

  db.collection name.insert({key, value }
- **For example,**

  myStudents> show collections

  myStudents> db.create Collection("stedent details)

  {ok: 1}

  myStudents> db.Student_details.insert (name: AAA" age":2}) acknowledged : true,

  myStudents>

  insertedIds: {'0': 0bjectId("643e4b63436497399alala66")}

  myStudents>
- We can verify this insertion by issuing following command myStudents> db. Student.details.find()

  [{id: Object Id ("643e4b63436497399al a2 a66"), name: AAA, age: 22}] myStudents>

5) Inserting Multiple Documents

- It is possible to insert multiple documents at a time using a single command. Following program shows how to insert multiple documents in the existing collection.

  my Students> var all_students. = [

  {

"name": "BBB" "age":20
},
{
"name": "DDD" "age":21
},
];
acknowledged: true, end
details.find()
myStudents> db.Student detaits . Insert(allstudents); insertedIds: {'0': ObjectId ("643e518d436197399ala2a67") '1': "ObiectId"643e518d436497399aia2a68")
}
}
myStudents>

- To verify the existence of these documents in the collection you can use find command as follows -

myStudents> db.Student_details. find ()
{-id: ObjectId ("643e4b63436497399ala2a66"), name: 'AAA', age: 22}
{-íd: ObjectId ("643e518d436497399ala2a67 "), name: 'BBB', age: 21,}
{-íd: ObjectId ("643e518d436497399ala2a67 "), name: 'DDD', age: 20,}
}
myStudents>

6) Delete Documents

- For deleting the document, the remove command is used.
- This is the simplest command
- **Syntax**

db.collection_name.remove(delete_criteria)

- **For example,** first of all, we can find out the documents presents in the collection using find () command

myStudents> db.Student_details. find () [
{-id: ObjectId ("643e4b63436497399ala2a66"), name: 'AAA', age: 22}
{-íd: ObjectId ("643e518d436497399ala2a67 "), name: 'BBB', age: 21,}
{-íd: ObjectId ("643e518d436497399ala2a67 "), name: 'DDD', age: 20,}
]
myStudents>

- Now to delete a record with name "DDD" we can issue the command as follows-

myStudents> db.Student details.deleteOne({“name”: “DDD” });
{acknowledged: true, deletecount: 0}
myStudents>

- Now using find () command we can verify if the desired data is deleted or not.
  myStudents> db.Student_details. find ()
  {-id: ObjectId ("643e4b63436497399ala2a66"), name: ‘AAA’, age: 22}
  {-íd: ObjectId ("643e518d436497399ala2a67 "), name: ‘BBB’, age: 21,}
  }

7. Delete all documents

- For deleting all the documents from the collection,we can use.deleteMany()
- **For example,**
  myStudents> db.Student details.deleteMany({ });
  {acknowledged: true, deletecount:2} myStudents>
- We can verify it using db.Student_details.find() command,we can verify that records are deleted or not.
  myStudents> db.Student_details. find () myStudents>

8. Update Documents

- For updating the document, we have to provide some criteria based on which the document can be updated.
  db.collection_name.update(criteria, update_data)
- **For example -** Suppose the collection "Student_details” contain following documents
  myStudents> db.Student_details.find() [
  {_id: ObjectId ("643e4b63436497399ala2a66"), name: ‘AAA’, age: 22}
  {_íd: ObjectId ("643e518d436497399ala2a67 "), name: ‘BBB’, age: 21,}
  {_íd: ObjectId ("643e518d436497399ala2a67 "), name: ‘DDD’, age: 20,}
  {_íd: ObjectId ("643e518d436497399ala2a67 "), name: ‘CCC’, age: 23,}
- And we want to change the name CCC" to WWW", then the command can be issued as
  myStudents>db.Student_details.
  update({“name:”CCC”},{set:{“name”:”WWW”}}) DeprecationlWarning: Collection. update () is deprecated. Use updateOne, updateMany, or bulkwrite.
  {
  acknowledged: true, InsertedId: null, matchedCount: 1, modifiedCount:1, upsertedCount:0
  }

- It can be verified using db.Student details.find() command my Students> db.Student_details .find().

  [

  {-id: ObjectId ("643e4b63436497399ala2a66"), name: 'AAA', age: 22}

  {-id: ObjectId ("643e518d436497399ala2a67 "), name: 'BBB', age: 21,}

  {-id: ObjectId ("643e518d436497399ala2a67 "), name: 'DDD', age: 20,}

  {-id: ObjectId ("643e518d436497399ala2a67 "), name: 'WWW', age: 23,}

  ]

  myStudents>
- Thus, the document gets updated.
- By default, the update command updates a single document. But we can update multiple documents as well. For that purpose, we have to add db. Student details. update ({"age"21}, ($set: {"age":23}}, {multi: true})

11. Write about Column Oriented Database: HBase OR State and explain HBase model and CRUD operations with examples. (NOV/DEC 2022)

Concept of Column Oriented Database

- The column-oriented database is a database system that stores the data table by column rather than by rows. The advantage of column-oriented Database is that we can access the data more efficiently when only subset of columns is used for querying.
- **For example-**Consider the customer table

Archana	**email address**	**gender**	**age**
	archana123@gmail.com	Female	30

Archana	**email address**	**gender**
	ankit123@gmail.com	Male

Aswini	**email address**	**Country**
	aswini123@gmail.com	India

- We can see that a column family has several rows. Within each row, there can be several different columns, with different names, links, and even sizes.

Advantages of Column oriented database

- The most important benefit of column-oriented database is faster performance compared to row-oriented database
- The column-oriented database can store more data in smaller amount of memory.
- The column-oriented database is highly used for big data applications.
- Another benefit of a column-based DBMS is self-indexing, which uses less disk space than Relational database schema.
- We can use aggregation for vast amount of data.

12. Write about Introduction to HBase Data Model in detail. (APR/MAY 2024)

HBase Data Model

- HBase is a column oriented non-relational database management system. It runs on top of Hadoop Distributed File System (HDFS)
- HBase is an Open-source technology developed by Apache Software Foundation.
- The HBase can store massive amount of data from terabyte to petabytes

Features of HBase Data Model

1) HBase is horizontally scalable. That means we can add any number of columns anytime.

2) The data is stored in key value format.

3) It is a platform for storing and retrieving data with random access.

4) It does not enforce relationship within the data.

5) It is designed to run on cluster of computers.

Difference between RDBMS and HBase

RDBMS	HBase
It requires SQL.	There is no SQL. in HBase.
It has fixed schema.	It does not have a fixed schema
It is row-oriented.	It is column oriented.
It is static in nature	It is dynamic.
The data retrieval is slower.	Data can be retrieved fast in HBase
In RDBMS, the data is normalized.	The HBase is not normalized
It accommodates small tables.	It accommodates large tables

HBASE CRUD Operations

- The CRUD stands for Create, Read, Update and Delete Operations. In order to communicate with HBase, the HBase Shell is used.
- The CRUD operations are verified by issuing the Create, put, get alter, delete, list and many more such commands. Let us discuss some commonly used commands

1) Create Table:

- The create command is used to create a table.
  CREATE <table-table>','<column-family>
- **For example,**
  > CREATE customer ef

2) Read Operation:

- The 'get' command is used to read the contents of row or cell.
- The syntax is-
  get<table-name>', '<row>"
- **For example-**
  hbase(main): 015:0> get 'customer', '1'
  COLUMN CELL
  cf: ge timestamp = 1417621885277, value - 30
  ef:city timestamp =1417521848375, value-Pune

cf: name timestamp = 1417521785385, value- Supriya

- Reading a specific column

  hbase(main): 018:0> get customer', 'T', (COLUMN 'cfname")

  COLUMN CELL

  cf: name timestamp = 1418035791555, value Supriya

  1 row(s) in 0.0080 seconds

3) Insert Data:

- We can insert data to the table using the put command.
- Now, we can create a table by inserting some data into it in column-based manner using put command

  >put customer 1,'ct name", "Supriya

  >put 'customer',1,'cf: city', 'Pune'

  >put 'customer',1,'cfage', '30"

- We get the table as follows

Row Key	cf:name	cf:city	cf:age
1	Supriya	Pune	30

- We can check if table is created using **list** command.
- **For example**,

  > list 'customer

4) Update Operation

- We can modify some properties of existing table using alter command.
- We add more column families or we can modify the table attributes.
- The syntax is alter <tablename>, NAME> <column family name> VERSIONS> <new version no>
- To add the column family 'status' in table 'customer' and to keep a maximum of 5 cells

  VERSIONS

  >alter 'customer, NAME'status', VERSIONS =>5
- To delete some column family 'status' we use alter command as

  >alter customer, NAME->'status', METHOD=>delete

5) Deletion Operation

- We can delete a particular cell by using the delete command. The syntax is
- **For example-**We can delete a cell for the column city from the customer table as follows hbase(main) 0060 delete customer, T. of city,
  1417621948376
  0 row 0.0060 seconds
- **Delete all:** It deletes all the cells in a row
  desteel table name>, <row>
- **For example-**
  bum (main) 0070 deletesil customer. T
  (E) Display the Contents of Table
- For displaying the contents of the table '**scan**' commad is used.
- **For example,**
  hBase(main) 040:0> scan 'customer' NOW COLUMN+CELL
  1 column =cf:age, timestamp 1505200004953, value-30
  1 column =cf:city, timestamp-1505200291903, value-Pune
  1 culumn=ct:name, timestamp=1505200278058, value-Supriya

04/01/2023 - FN.

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 70006

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2022.

Third Semester

Artificial Intelligence and Data Science

AD 3391 — DATABASE DESIGN AND MANAGEMENT

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define the concept of specialization in ER diagram.
2. What is sequence diagram? and why do we use it?
3. Define the terms relation schema.
4. Why null values might be introduced into the database.
5. What is partial functional dependencies? Give example
6. What is the lossless join property of decomposition? Why is it important?
7. What is a transaction?
8. Why must lock and unlock be atomic operations?
9. What are the two kinds of new data types supported in object-database systems?
10. When to use SQL and NOSQL?

PART B — (5 × 13 = 65 marks)

11. (a) State and explain the database system development life cycle with an example.

Or

(b) Explain the following terms briefly in E-R model: attribute, domain, entity, relationship, entity set, relationship set, one-to-many relationship, many-to-many relationship, participation constraint, overlap constraint, covering constraint, weak entity set, aggregation, and role indicator.

12. (a) (i) Describe the four clauses in the syntax of a simple SQL retrieval query. Show what type of constructs can be specified in each of the clauses. Which are required and which are optional? (7)

(ii) State the need for triggers. When to use triggers and when not to use triggers? Explain with example. (6)

Or

(b) Discuss how NULLs are treated in comparison operators in SQL. How are NULLs treated when aggregate functions are applied in an SQL query? How are NULLs treated if they exist in grouping attributes?

13. (a) Discuss the correspondences between the ER model constructs and the relational model constructs. Show how each ER model construct can be mapped to the relational model and discuss any alternative mappings.

Or

(b) What is the need for normalization? Discuss the various normalization technique used with example.

14. (a) What is the function of locking protocol? State and explain the two-phase locking with example.

Or

(b) (i) Define these terms: atomicity, consistency, isolation, durability, schedule, blind write, dirty read, unrepeatable read, serializable schedule, recoverable schedule, avoids-cascading-aborts schedule. (7)

(ii) Compare binary locks to exclusive/shared locks. Why is the latter type of locks preferable? (6)

15. (a) (i) What are collection hierarchies? Give an example that illustrates how collection hierarchies facilitate querying. (7)

(ii) Discuss how a DBMS exploits encapsulation in implementing support for ADTs (6)

Or

(b) (i) What are indexes in MongoDB? State and explain different types of index with example. (7)

(ii) State and explain the Hbase data model and CRUD operations with example. (6)

PART C — (1 × 15 = 15 marks)

16. (a) Consider the following relations.

Suppliers (sid:integer sname:string address:string)

Parts(pid:integer pname:string color:string)

Catalog(sid:integer pid:integer cost:real)

Write SQL statement for the following queries (5 × 3 = 15)

(i) Construct the E-R Diagram for the schema given.

(ii) Find the names of suppliers who supply some red part.

(iii) Find the sids of suppliers who supply some red part or are at No: 1, Anna Salai.

(iv) Find the pids of parts supplied by at least two different suppliers.

(v) Find the pids of the most expensive parts supplied by suppliers named sam.

Or

(b) A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color. Special data are included for certain types of vehicles:

- Trucks: cargo capacity.
- Sports cars: horsepower, renter age requirement.
- Vans: number of passengers
- Off-road vehicles: ground clearance, drivetrain (four-or two-wheel drive).

Construct an SQL schema definition for this database. Use inheritance where appropriate.

———————

16/6/23 - FN.

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 30010

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2023.

Third Semester

Artificial Intelligence and Data Science

AD 3391 – DATABASE DESIGN AND MANAGEMENT

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is quaternary in UML? Give an example.
2. Compare between Centralized Approach and View Integration Approach in requirement collection.
3. What is a Referential Integrity? Give an example.
4. Why do we need view in SQL?
5. Compare between ER model and EER model.
6. Write a note on BCNF normal form.
7. Define granularity.
8. How serializability is guaranteed by Two – Phase Locking technique?
9. State the CAP theorem in NOSQL.
10. Compare between subtypes and supertypes.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Illustrate in detail about the three levels (external, conceptual, internal) of database architecture. (7)

(ii) Outline the flow of database system development lifecycle and explain. (6)

Or

(b) (i) Exemplify in detail about the diagrammatic representation of entity and relationship types in UML. (7)

(ii) Compare and contrast between specialization and generalization process in Enhanced ER modeling. (6)

12. (a) Describe in detail about the relationship between mathematical relations and relations in the relational data model.

Or

(b) (i) Explain in detail about SELECT statement in SQL with its general form and examples. (7)

(ii) Discuss in detail about SQL datatypes with examples. (6)

13. (a) Illustrate in detail about Functional dependencies with relevant examples.

Or

(b) Exemplify in detail about First, Second and Third Normalization with relevant examples.

14. (a) (i) Sketch the state transition diagram for a database transaction and explain the flow. (7)

(ii) Describe the ACID properties of transaction processing. (6)

Or

(b) Discuss in detail about Two Phase Locking and Timestamp based protocols in Concurrency Control.

15. (a) Explain in detail about Object Relational Data Model with diagram.

Or

(b) Discuss in depth about MongoDB database and its CRUD operations with an example application.

PART C — (1 × 15 = 15 marks)

16. (a) Create an Entity – Relationship diagram for a car dealership. The dealership sells both new and used cars, and it operates a service facility. Base your design on the following business rules:

- A salesperson may sell many cars, but each car is sold by only one salesperson.

- A customer may buy many cars, but each car is bought by only one customer.
- A salesperson writes a single invoice for each car he or she sells.
- A customer gets an invoice for each car he or she buys.
- A customer may come in just to have his or her car serviced; that is, a customer need not buy a car to be classified as a customer.
- When a customer takes one or more cars in for repair or service, one service ticket is written for each car.
- The car dealership maintains a service history for each of the cars serviced. The service records are referenced by the car's serial number.
- A car brought in for service can be worked on by many mechanics, and each mechanic may work on many cars.
- A car that is serviced may or may not need parts (e.g., adjusting a carburetor or cleaning a fuel injector nozzle does not require providing new parts)

Or

(b) Create the following tables using SQL

Employee (FName, Middle, LName, SSN, BDate, Address, Sex, Salary, SupervisorSSN, DeptNo)

Department (DName, DeptNo, Manager SSN, StattDate)

Write SQL queries to perform the following tasks:

(i) Find the sum of the salaries of all employees of the 'Accounts' department as well as the maximum salary, the minimum salary, and the average salary in this department.

(ii) Retrieve the name of each employee Controlled by department number 5

(iii) Retrieve the name of each Dept and number of employees working in each department which has at least 2 employees.

(iv) Retrieve the name of employees who born in the year 1990's.

(v) Retrieve the name of employees and their Dept name.

———————

Reg. No. : E N G G T R E E . C O M

Question Paper Code : 20013

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2023.

Third Semester

Artificial Intelligence and Data Science

AD 3391 – DATABASE DESIGN AND MANAGEMENT

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What are the disadvantages of file processing systems?
2. List the applications of DBMS.
3. What are the categories of SQL command?
4. What is a candidate key and primary key?
5. List the desirable properties of decomposition.
6. What is meant by normalization of data?
7. What are the ACID properties?
8. What are the different modes of lock?
9. What are the advantages of NoSQL?
10. Write the advantages of object oriented model.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Design a relational database for a university registrar's office. The office maintains data about each class, including the instructor, the number of students enrolled, the time and place of the class meetings. For each student-class pair, a grade is recorded. (6)

(ii) Draw an ER-diagram for the above specified relational database for a university registrar's office. (7)

Or

(b) (i) Explain the 3 schema architecture of DBMS. Why do we need mappings between different schema level? (6)

(ii) Explain the architecture of DBMS. (7)

12. (a) List and explain various DDL, DML and DCL commands in detail with examples. (13)

Or

(b) Explain in brief about Subqueries and Correlated queries. (13)

13. (a) Explain various normal forms in database management systems which are required for fulfilling normalization requirements of an organization. (13)

Or

(b) Explain in detail, the Closure of set of functional dependency and Closure of Attribute sets. (13)

14. (a) Explain in detail two-phase locking and how does it guarantee serializability. (13)

Or

(b) Discuss the concurrency control mechanism in detail using suitable example. (13)

15. (a) Explain mapping an EER Schema to an ODB Schema in detail. (13)

Or

(b) Explain MongoDB-Data Modelling in detail with a real time example. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Consider the following schema :

Suppliers (sid: integer, sname: string, address: string)

Parts (pid: integer, pname: string, color: string)

Catalog (sid: integer, pid: integer, cost: real)

Write appropriate SQL commands to solve the following queries :

(i) Find the names of suppliers who supply some red part.

(ii) Find the sids of suppliers who supply some red or green part.

(iii) Find the sids of suppliers who supply some red part or are at 221 Packer Street.

(iv) Find the sids of suppliers who supply some red part and some green part.

(v) Find the sids of suppliers who supply every part.

Or

(b) Consider the following Schema :

Flights (flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time)

Aircraft (aid: integer, aname: string, cruisingrange: integer)

Certified (eid: integer, aid: integer)

Employees (eid: integer, ename: string, salary: integer)

Write appropriate SQL commands to solve the following queries :

(i) Find the eids of pilots certified for some Boeing aircraft.

(ii) Find the names of pilots certified for some Boeing aircraft.

(iii) Find the aids of all aircraft that can be used on non-stop flights from Bonn to Madras.

(iv) Identify the flights that can be piloted by every pilot whose salary is more than $100,000.

(v) Find the names of pilots who can operate planes with a range greater than 3,000 miles but are not certified on any Boeing aircraft.

Question Paper Code : 50009

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2024.

Third/Fourth Semester

Artificial Intelligence and Data Science

AD 3391 — DATABASE DESIGN AND MANAGEMENT

(Common to Computer Science and Engineering (Artificial Intelligence and Machine Learning))

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. List any two advantages of database systems.

2. State the different types of integrity constraints used in designing a relational database.

3. Define trivial functional dependency.

4. An object is created without any reference to it, how can that object be deleted?

5. Write an efficient relational algebraic expression for the following query:

   SELECT B1.BANKNAME FROM BANK AS B1, B2 WHERE B1.ASSETS> B2.ASSETS AND B2.BANKLOCATION="TAMILNADU".

6. List the steps involved in query processing.

7. Define the term transaction. Give an example.

8. State the benefits of strict two-phase locking.

9. Distinguish total rollback from partial rollback.

10. State denormalization.

PART B — (5 × 13 = 65 marks)

11. (a) Classify Database system architecture and explain.

Or

(b) Construct an ER-diagram for hospital management system with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examination conducted.

(i) Draw the ER-diagram for Hospital Database. (7)

(ii) For each entity set and relationship used, indicate primary key, 1-1, many to one and one to many relationships. (6)

12. (a) Distinguish between procedural and non-procedural languages. Is relational algebra procedural or non-procedural. Explain the operations with example.

Or

(b) Consider the following relational database :

Employee (person_name, street, city)

Works (person_name, company_name, salary)

Company (company_name, city)

Manager (person_name, manager_name)

Give an SQL DDL, definition of this database. Identify referential integrity constraints that should hold, and include them in DDL definition.

13. (a) Consider the relation $R(A,B,C,D,E)$ with functional dependencies

$\{a \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

Identify super keys. Find F*.

Or

(b) Discuss the procedure used for loss-less decomposition with an example.

14. (a) Distinguish recoverable and non-recoverable schedules. Why is recoverability of schedules desirable? Are there any circumstances under which it would be desirable to allow non-recoverable schedules? Justify your answer.

Or

(b) What is the need for concurrency control mechanisms? Explain the working of lock-based protocols.

15. (a) Describe the features of object-oriented data model.

Or

(b) Explain the HBase data model with an example.

PART C — (1 × 15 = 15 marks)

16. (a) Describe normalization upto 3NF and BCNF with examples. State the desirable properties of decomposition.

Or

(b) Discuss query optimization with a diagram.