3. What is DBMS? What are the advantages and disadvantages of DBMS? (5 M)

**Ans.** Refer to Sections 1.1.4 and 1.5.2.

4. Write a note on functional components of DBMS. (5 M)

**Ans.** Refer to Section 1.4.

**October 2017**

1. What is a physical file? (1 M)

**Ans.** Refer to Page 1.2 Point (2).

2. State any two functions of DBA. (1 M)

**Ans.** Refer to Section 1.5.1.3.

3. State any two advantages of DBMS over file system. (1 M)

**Ans.** Refer to Section 1.5.2.

4. What are the types of Data Independence? (1 M)

**Ans.** Refer to Section 1.3.2.

5. Write short note on Data Abstraction. (5 M)

**Ans.** Refer to Section 1.3.1.

6. Explain overall DBMS structure with neat diagram. (5 M)

**Ans.** Refer to Section 1.4.

**April 2018**

1. List any two advantages of DBMS. (1 M)

**Ans.** Refer to Section 1.5.2.

2. State different types of users of DBMS. (1 M)

**Ans.** Refer to Sections 1.5.1 and 1.5.1.1.

3. Write a short note on data independence. (5 M)

**Ans.** Refer to Section 1.3.2.

❖❖❖

CHAPTER 2

# Conceptual Design

## 2.0 INTRODUCTION

- Database design is the process of producing/developing a detailed data model of a database. Database design is the process of constructing a stable database structure from user requirements.
- Database design enables us to represent the real-world entities in a form that can be processed by the computer.
- Database design requires understanding both the operational and business requirements of an organization as well as the ability to model and realize those requirements using a database.
- Conceptual modeling is an important in designing a successful database. Conceptual modeling is also called as semantic modeling.
- Conceptual modeling represents various pieces of data and their relationships at a very high-level of abstraction. Conceptual design mainly focuses on what data is required and how it should be organized rather than what operations are to be performed on the data.
- Database desgin involves a set of activities, which include requirements analysis, conceptual database design, selection of a database management system, mapping from conceptual model to physical model and the physical database design.

## 2.1 OVERVIEW OF DB DESIGN PROCESS

- The process of designing a database begins with an analysis of what information the database must hold and what are the relationships among components of that information.
- Design the logical and physical structure of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications.
- The database design is a process of designing the logical and physical structure of one or more databases. The reason behind it is to accommodate the information needs or queries of the users for a defined set of applications and support the overall operation and objectives of an enterprise/organization.
- The goals of database (DB) design are explained below:
  1. Provide a natural and easy-to-understand structuring of the information.
  2. Satisfy the information content requirements of the specified users and applications.
  3. Support processing requirements and any performance objectives, such as response time, processing time and storage space.
- Various phases in DB design in Fig. 2.1 are explained below:

**Phase 1: Requirements Collection and Analysis:**

- Before we can effectively design a database, we must know and analyze the expectations/ requirements of the users and the intended uses of the database in as much detail as possible. This process is called requirements collection and analysis.
- To specify the requirements, we must first identify the other parts of the information system that will interact with the database system.
- These include new and existing users and applications, whose requirements are then collected and analyzed.

**Phase 2: Conceptual Database Design:**

- The goal of this phase is to produce a conceptual schema for the database that is independent of a specific DBMS.
- We often use a high-level data model such as the E-R or EER model during this phase.
- Additionally, we specify as many of the known database applications or transactions as possible, using an notation that is independent of any specific DBMS.
- Often, the DBMS choice is already made for the organization; the intent of conceptual design is still to keep it as free as possible from implementation considerations.

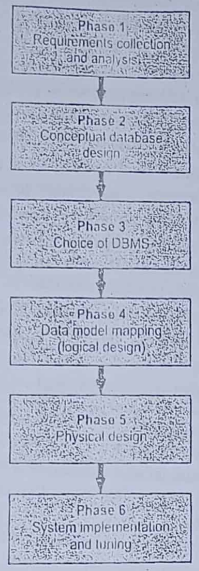



Fig. 2.1: Phases in DB Design

**Phase 3: Choice of DBMS:**

- The choice of a DBMS is governed by a number of factors.
  - (i) **The technical factors** are concerned with the suitability of the DBMS for the task at hand. Issues to consider are the type of DBMS (relational, object-relational, object, etc.), the storage structures and access paths that the DBMS supports, the user and programmer interfaces available, the types of high level query languages, the availability of development tools, the ability to interface with other DBMSs via standard interfaces, the architectural options related to client-server operations and so on.
  - (ii) **Non-technical factors** include the financial status and the support organization of the vendor.

**Phase 4: Data Model Mapping:**

- During this phase, which is also called logical database design, we map (or transform) the conceptual schema from the higher-level data model used in Phase 2 into the data model of the chosen DBMS.
- We can start this phase after choosing a specific type of DBMS (for example, if we decide to use some relational DBMS but have not yet decided on which particular one).
- We call the latter system-independent (but data model-dependent) logical design. In terms of the three-level DBMS architecture, the result of this phase is a conceptual schema in the chosen data model.
- In addition, the design of external schemas (views) for specific applications is often done during this phase.

**Phase 5: Physical Database Design:**

- During this phase, we design the specifications for the stored database in terms of physical storage structures, record placement and indexes.
- This corresponds to designing the internal schema in the terminology of the three-level DBMS architecture.

**Phase 6: Database System Implementation and Tuning:**

- During this phase, the database and application programs are implemented, tested and eventually deployed for service.



- Various transactions and applications are tested individually and then in conjunction with each other. This typically reveals opportunities for physical design changes, data indexing, reorganization and different placement of data. An activity referred to as database tuning.
- Tuning is an ongoing activity – a part of system maintenance that continues for the life cycle of a database as long as the database and applications keep evolving and performance problems are detected.

## 2.2 INTRODUCTION TO DATA MODELS (April 17)

- A data model is a collection of conceptual tool for describing data, its relationship, data semantics and data constraints.
- A data model consists of:
  1. A named logical unit i.e., record type and data item.
  2. Relationship among these logical units.
  3. Data item is logical unit of data and record type is collection of data items.
- We can say, a data model is description of a container in which data and its methods of storing and retrieving data are available. Actually, it provides abstraction of database application.

**Definition and Types of Data Models:**

A data model is a collection of concepts that can be used to describe the structure of a database. A database model is a specification describing how a database is structured and used.

- A model defines the method of storing and retrieving data. Data modeling helps in the understanding of the meaning of the data.
- A data model is a way of representing data and its inter-relationship. Data models play a major role in the database design process.
- The process of applying a data model theory to create a data model instance is known as data modeling. Data modeling is a way to structure and organize data so it can be used easily by databases.

**Definition of Data Model:**

- A database model is, "a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated". OR
- A data model is defined as, "an integrated collection of concepts for describing and manipulating data, relationships between data and constraints on the data in an organization".

**Types of Data Models:**

- Fig. 2.2 shows categories of data models and the models are described below: conceptual ✓

1. **Object Based Data Models:** These data models are used to describe data at the logical and view level of a database. It is used to describe logical data structure. There are two subtypes:
   - (i) **Entity Relationship Model:** It is high level data model based on the fact of real world. This model consists of collection of basic objects called as entities and the relationship among these objects.
   - (ii) **Object Oriented Model:** This model is based on collection of objects. Each object has its own methods. Actually the value stored for a object and methods are grouped into a logical unit called as class.
2. **Physical Data Models:** These data models describe how data is stored in the computer. It represents information like record structures, record ordering and access paths. For example, the Unifying model and the Frame memory model.
3. **Record Based Models:** These data models are used in describing data at the logical and view levels. These models are so named because the database is structured in fixed format records of several types. (April 17)

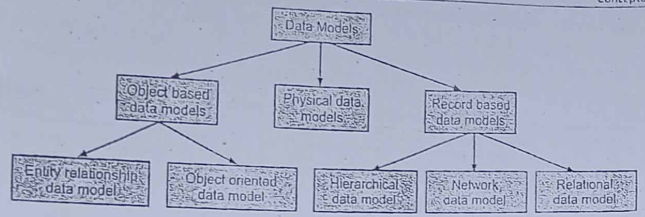



Fig. 2.2: Types of Data Models

## 2.2.1 Entity-Relationship Model

(April 16, 19)

- An Entity Relationship (E-R) model was introduced by Dr. Peter Chen in 1976. E-R model is the most popular high-level conceptual data model and used for the conceptual design of database applications.
- An E-R data model is defined as, "a conceptual data model that views the real world as entities and relationships".
- An E-R model is useful to a database designer in the following ways:
  1. An E-R data model maps well to the relational model i.e., the constructs used in the E-R data model can be easily transformed into relational tables.
  2. An E-R data model can be used by the database designer to communicate the design to the end user.
- An E-R data model is a systematic way of describing and defining a conceptual database design process.
- An E-R model is based on a perception of a real world which consists of a collection of basic objects called entities and relationship among these objects.
- It represents the overall logical structure of a database. An E-R model is normally expressed as an E-R diagram which is a graphical representation of a particular database.
- Elements as the E-R model includes Entities, Attributes, Entity sets, Relationships, Relationship sets etc. Fig. 2.3 shows basic concepts of E-R data model using enterprise College where the students are studied.

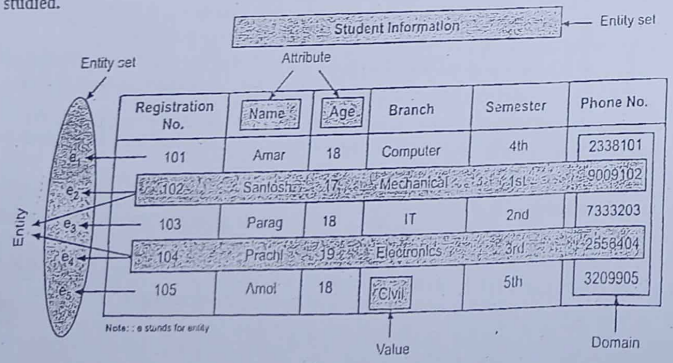



Fig. 2.3: Representation of E-R Model



- The basic concepts of E-R model are explained below:
  1. **Entity:** It refers to an "object" or "thing" in real world, such as students of colleges, loans in banks, employees in any organisation etc.
  2. **Attributes:** These are characteristics of any entity, like a student can be described by his name, age, address, height, class, department etc.
  3. **Value:** It is the information or data which is stored in attributes of any entity.
  4. **Entity Sets:** All the entities having same attributes make an entity set.
  5. **Domain (Value Set):** It is the set of all values or information about any attribute.
- The E-R model has several advantages as given below:
  1. E-R model captures the real-world data requirements in a simple, meaningful and logical way.
  2. It can be easily mapped onto the relational model. The basic constructs, that is, the entities and attributes of the E-R model can be easily transformed into relations (or tables) and columns (or fields) in a relational model.
  3. It can be used as a design plan and can be implemented in any database management software.
  4. It is simple and easy to understand and thus, can be used as an effective communication tool between database designers and end users.

## 2.2.2 Relational Model

- The relational model was formally introduced by Dr. E. F. Codd in 1970. This model shows collection of tables to represent both data and the relationship. Relational model is Record Based Logical Model.
- A relational model database is defined as, "a database that allows to group its data items into one or more independent tables that can be related to one another by using fields common to each related table".
- The relational model represents data in the form of two-dimension tables called as relations. Each table represents some real-world thing, (person, book, house etc.) or event about which information is collected.
- A relational table is a flat file composed of a columns (attributes) and rows (values for columns). The columns of the tables contain information about the table. The rows of the table represent occurrences of the "thing" represented by the table.
- A data value is stored in the intersection of a row and column. Each named column has a domain, which is the set of values that may appear in that column.
- **For example:** In a database, CUSTOMER table and ACCOUNT table are the two tables available. The relationship between these two tables have been shown by a third table, where one field or column name from both the table is taken together. So a new table called CUSTOMER-AMT is formed.

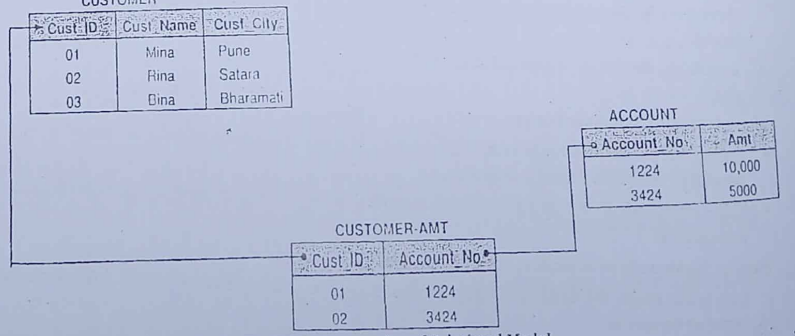



Fig. 2.4: Representation of Relational Model

**Advantages:**

1. **Easier to Design, Implementation, Maintenance and Usage:** Both data as well as structural independence are provided by the relational model which makes database design, maintenance, administration and usage much easier and simpler than the other models.
2. **Simple:** It is simpler model as it frees the designers from the actual physical data storage details. This allows them to concentrate on the logical view of the database.
3. **Structural Independence:** This model has structural independence, i.e. changes made in the database structure does not affect the DBMS's capability to access data.
4. **Better Query Capability:** This model provides very powerful, flexible and easy-to-use query facilities. It uses Fourth-Generation Languages (4GL) like SQL that makes ad_hoc queries a reality.
5. **No Anomalies:** This model does not suffer from insert, update, and delete anomalies. The retrieval operation is very simple, easy and systematic.

**Disadvantages:**

1. **Ease of Design can Result in Bad Design:** As the relational database is an easy-to-design and use system, it can result in the development and implementation of poorly designed database management systems.
2. **Performance Degradation:** As the size of the database increases, several problems may creep in this model like system slowdown, performance degradation and data corruption.
3. **Hardware Overheads:** The RDBMS needs comparatively powerful hardware as it hides the implementation complexities and the physical data storage details from the users.

## 2.2.3 Network Model

- In late 1960s, the DataBase Task Group (DBTG) formalized network model. Fig. 2.5 shows network model.
- In network model data is represented by collection of records and the relationship among the data is represented by links. The records are organized as a collection of arbitrary graphs.
- Network model consists of network diagram in which record type corresponds to the entity sets and set type corresponds to the relationship represented by links.
- **For example:** In CUSTOMER table extra field or column is considered, which consists of the address where ACCOUNT table information is stored. Each link information represents the related record address.

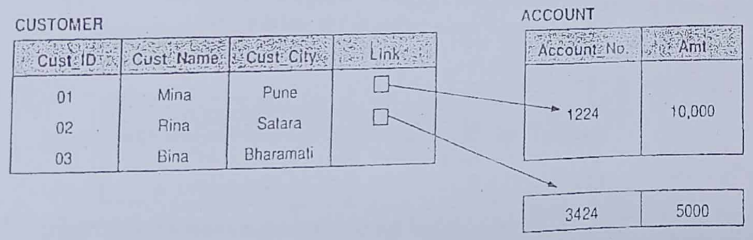



Fig. 2.5: Representation of Network Model

**Advantages:**

1. **Simplicity:** This model is simple and easy to design just as hierarchical data model.
2. **Data Integrity:** This model, no member can exist without an owner. A user must therefore first define the owner record and then the member record. This ensures the data integrity.
3. **Capable of Handling More Relationship Type:** This model handles 1 : 1, 1 : M and M : M relationships which helps in modeling the real world applications.



4. **Data Independence:** This model is better than the hierarchical model in isolating the programs from the complex physical storage details.
5. **Database Standards:** This model is based on the universal standards formulated by the DBTG.

**Disadvantages:**

1. **Lack of Structural Independence:** Making structural modifications to the database is very difficult in this model as the data access method is navigational.
2. **Not User-friendly:** This model is not a design for user-friendly system and is a highly skill-oriented system.
3. **Operational Anomalies:** Since, network model contains operational anomalio like update, insert, delete etc.
4. **System Complexity:** As all the records are maintained using pointers, so the whole database structure becomes very complex.

## 2.2.4 Hierarchical Model

Oct.-16

- This is the oldest type of data model, developed by IBM in 1968.
- In a hierarchical model, data is organized into a tree-like structure in which each child node segment (also called as dependents) can have only one parent node (segment). At the top of hierarchy there is a single node is called as root.
- Data in this model is similar to network model i.e., links or pointers used to show the relationships. Only change is that the records are organized as a collection of tree.
- In this model data is viewed as a collection of relations (segments) that form a hierarchical relation. A hierarchical relation is defined as, "a collection of relations that are connected together as a hierarchical tree by logical association (links)".

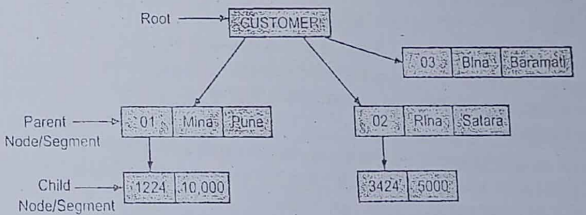



Fig. 2.6: Representation of Hierarchical Model

**Advantages:**

1. **Simplicity:** The database is based on the hierarchical structure, the relationship between the various layers is logically (conceptually) simple. Thus, the design of a hierarchical database is simple.
2. **Data Integrity:** The hierarchical model is based on the parent/child relationship, there is always a link between the parent segment and the child segments under it. The child segments are always automatically referenced to its parent, this model promotes data integrity because of parent/child relationship.
3. **Efficiency:** This model is a very efficient one when the database contains a large number of 1:N relationships (one-to-many relationships).
4. **Data Sharing:** Because all data are held in a common database, data sharing becomes practical in this model.

**Disadvantages:**

1. **Inflexibility:** A hierarchical database lacks flexibility. The changes in the new relations or segments often yield very complex system management task.
2. **No Standards:** The implementation of this model does not confirm to any specific standard.
3. **Implementation Complexity:** Although the hierarchical database model is quite complex to implement. The database designers should have very good knowledge of the physical data storage characteristics.
4. **Implementation Limitation:** Many of the common relationships do not conform to the 1:*N* format as required by the hierarchical model.
5. **Database Management Problems:** If any changes in the database structure of a hierarchical database, then it need to make the necessary changes in all the application programs that access the database. Thus, maintaining the database and the applications can become very difficult. It also takes time, so it becomes time consuming.
6. **Operational Anomalies:** This model suffers from the insert, update and deletion anomalies. The retrieval operation is complex and asymmetric.

**Comparison between Hierarchical Model, Network Model and Relational Model:**

| Sr. No. | Hierarchical Model | Network Model | Relational Model |
|---|---|---|---|
| 1. | This model is based on tree (hierarchy) structure. | This model is based on graph (tree of records) structure. | This model is based on mathematical concept of relations (tables). |
| 2. | Structural independence is missing. | Structural independence is missing. | It offers structural independence. |
| 3. | Hierarchical database consists of collection of records, connected to each other by links. | Network database consists of collection of records, connected to each other by links. | Relational database consists of tables (relations) and data is stored in tabular form. |
| 4. | Queries are easy to write than in network model but more complicated than relational model. | Queries are more complicated to write than hierarchical and relational model. | Queries are easy and simple to write than other models. |
| 5. | Data access is navigational. | Data access is navigational. | Data access is non-navigational. |
| 6. | It is easy to understand and more efficient than network data model. | It offers more flexibility than hierarchical model. | It offers simplicity in representing data than network and hierarchical models. |
| 7. | The hierarchical structure is asymmetric and is a major drawback that leads to unnecessary complications for the user. | The network structure is more symmetric than hierarchical structure. | The relational structure is more symmetric than network and hierarchical structure. |
| 8. | It is difficult to access values at lower level. | Data accessing is more easy than hierarchical model. | Data accessing is easy and simple than other models. |

*Contd...*



| | | | |
|---|---|---|---|
| 9. | Information is replicated in hierarchical model which leads to inconsistency and wastage of storage space. | It offers more data consistency than hierarchical model. | If offers more data consistency than other two models. |
| 10. | It uses DI/I though COBOL. | It uses COBOL based commands. | It uses SQL. |

## 2.3 CONCEPTUAL DESIGN USING ER DATA MODEL (April 16, 18)

### 2.3.1 Entities (April 16)

- An entity is a real world object. This object has its own properties. This object is distinguishable from other objects.
- For example, student, employee, doctor, politician are the entities. Student has its own properties like his/her own name, address etc.
- An entity is represented by a rectangle in E-R diagram. Rectangles are named with the entity set they represent.
- For example, Fig. 2.7 shows entity Employee.

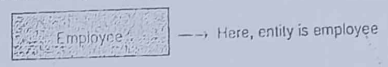



Fig. 2.7

- An entity must have some attribute through which it is uniquely identified. For example: Employee code is the identity of employee. Roll number is the identity of a student.

### 2.3.2 Attributes (April 16, 18, Oct. 16)

- An entity has set of properties called attributes which holds value in it. Once, the entity is identified, then it is described in real terms or through its attributes.
- An attribute is description that is used to identify, qualify, classify, or otherwise express the set of an entity occurrence or a relationship.
- For each attribute, set of permitted values are available called as domain. The attributes which uniquely defines the occurrence of an entity are called primary keys.
- If no attribute works as a primary key then a new attribute is defined for that purpose. Roll no and employee id is a primary key of above entities student and employee.
- For example,

DISEASE

| Disease Name | Causing Rrganism |
|---|---|
| Typhoid | Salmonella Typhi |
| Dysentry | Shigelladysentriae |

- No primary key is available. Then we can consider disease-no as extra field or attribute. An attribute must appear in only one place in the model. Either it may be required or optional.
- When the attribute is required, it holds a value. When it is optional, it may or may not have a value for it. For example, Entity plant. It's attributes are plant name, plant type, date of acquisition and pot size.
- Pot size is optional because some plants do not come in pots. They come in plastic bags. But other fields are required.

- In E-R diagram, an attribute is represented in two different ways:
  1. By ellipses attached to entities, (Refer Fig. 2.8).
  2. By listing the attribute as text, (Refer Fig. 2.9).

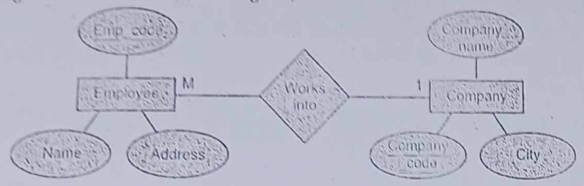



Fig. 2.8: Representation of attributes by Ellipses

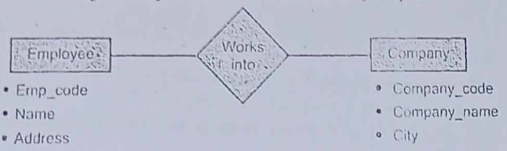



Fig. 2.9: Representation of Attribute by Text

- An attribute may be of following types:
  1. **Simple Attribute:** It represents a single property i.e. these are not divided into subparts. These are also called as atomic attributes. A simple attributes is defined as, "an attribute that is composed of a single component with an independent existence". For example, student's phone-number is an atomic value of 10 digits, (Refer Fig. 2.10).
  2. **Composite Attribute:** The attributes can be divided into subparts. This helps us to group together the related attributes. This may appear as a hierarchy. A composite attribute is defined as, "an attribute composed of multiple components each with an independent existence". For example, a student's complete name may have FirstName and LastName. Composite attributes are represented by ellipses that are connected with an ellipse, (Refer Fig. 2.11).

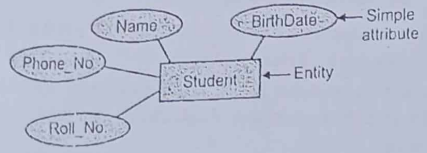



Fig. 2.10: Simple Attribute

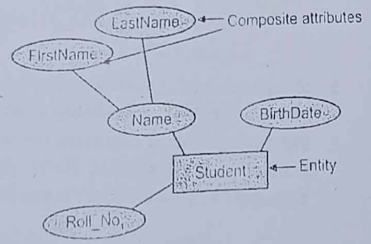



Fig. 2.11: Composite Attribute

  3. **Single valued Attribute:** The attributes which contains one value in their column is called single valued attributes. Single-valued attributes is defined as, "an attribute that holds a single value for a single entity". For examples: Roll_No, Name, Amount, Age has only one value. (April 15)
  4. **Multi-valued Attribute:** The attributes which contains more than one value for a specific entity is called multi-valued attributes. Multi-value attribute is defined as, "an attribute that holds a multiple values for a single entity". For example, a student can have more than one phone numbers, e-mail_addresses etc. Multi-valued attributes are depicted by double eclipse, (Refer Fig. 2.12). (April 15)



  5. **NULL Attribute:** The attribute without any value is called as null attribute. Null value is used when entity does not have a value for an attribute. There are two assumptions done for this null attribute. Sometimes when there is no value present we say "not applicable" or sometimes we may say "value is missing". Let's see the example. Suppose employee have no dependents and the name of dependent is null it means, this is not applicable. If say employee-code or social security number is null it means it is missing. The attribute phone-no of employee can have value NULL, if employee does not have telephone connection.
  6. **Derived Attribute:** The value of this attribute can be derived from values of other related attributes or entities. Derived attribute is defined as, "an attribute that represents a value that is derivable from the value of related attribute or a set of attributes, not essentially in the same entity". For example, age can be derived from BirthDate. Derived attributes are depicted by dashed ellipse, (Refer Fig. 2.13).

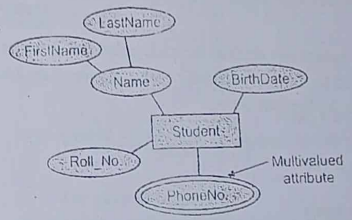



Fig. 2.12

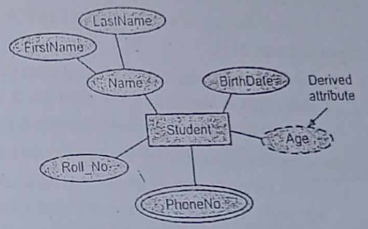
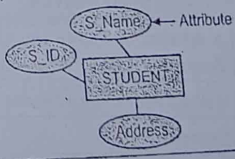



Fig. 2.13

Difference between Entity and Attribute:

| Sr. No. | Entity | Attribute |
|---|---|---|
| 1. | Entity represents a real world object or a thing that can stand on its own independently and can be identified uniquely. | Attributes are the properties of these entities. |
| 2. | Entities are represented as tables (each row representing individual instances) in relational databases. | In relational databases, where entities are realized as tables, each column represents the attributes of these entities. |
| 3. | Entity type is represented as rectangle enclosing the type name which is singular noun. | Attribute names are enclosed by ellipse and connected to their entities by single line. |
| 4. | Example:   | Example:   |

(April 18)

## 2.3.3 Entity Sets

- Entity is the basic object of E-R model. The set of all entities of the same type is called an entity set.
- An entity is an 'object' in the real world which has its own properties and it is distinguishable from other objects.
- An entity set is a set of entities of the same type that share the same properties or attributes.

- For example,
  1. The set of all persons who are employees at the college and defined as a entity set employee.
  2. The set of all persons who are students at the college and defined as a entity set student.
  3. If a bank has number of branches then branch is also a entity set. So in each branch entity-set may be described by the attributes branch_name, branch_city and assets.
- Entity sets do not need to be disjoint.

  For example, `employee of a college [employee]`

  `student of a college [student]`
- A person entity may be an employee entity or student entity or both or sometimes no one from this set. An entity may be concrete like a flower or a book or abstract like a concept or holiday.

**Types of Entity Sets:** (April 18)

- The attribute (or combination of attributes) whose values are distinct for each individual instance of an entity type is known as a key attribute. It is used to uniquely identify each instance of an entity type i.e., no two entities can have the same value for the key attribute.
- There are two types of entity sets as given below:
  1. **Weak Entity Set:** An entity set that does not have any key attribute of its own is called a weak entity set. (April 18)
  2. **Strong Entity Set:** An entity set that has a key attribute is called a strong entity set. (Oct.-17, April 18)

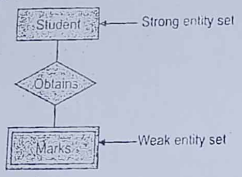


Fig. 2.14: Strong and Weak Entity Set

- The weak entity is also called a dependent entity as it depends on another entity for its identification.
- The strong entity is called an independent entity, as it does not rely on another entity for its identification.
- For example, entity set Marks which represents the marks obtained by a student. The existence of Marks entity depends upon the existence of Student entity.

**Difference between Strong Entity Set and Weak Entity Set:** (April 18, Oct. 17)

| Sr. No. | Strong Entity Set | Weak Entity Set |
|---|---|---|
| 1. | An entity set that has a key attribute is called as strong entity set. | The entity set which does not have any key attribute of its own is called as weak entity set. |
| 2. | A member of a strong entity set is called dominant entity. | A member of weak entity set is called as subordinate entity. |
| 3. | It is represented by a rectangle. | It is represented by a double rectangle. |
| 4. | Example: Student | Example: Marks |

## 2.3.4 Relations and Types of Relationship (Oct.-16)

- Relation means reference and relationship means connection. As it is related to a database, it shows the association between different entities.
- In short, the association among entities is called relationship.



- There are four types of relationships:
  1. **One-to-One (1 : 1) Relationship:** An entity in A is associated with at the most one entity in B and then entity in B is also associated with at the most one entity in A, (Refer Fig. 2.15).

     **For example:** One class has one Roster (Refer Fig. 2.16).

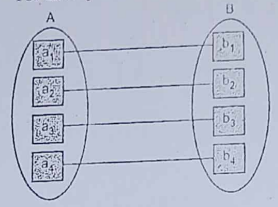


Fig. 2.15: One-to-One

Fig. 2.16: Representation of One-to-One Relationship

  2. **One-to-Many (1 : M) Relationship:** An entity in A is associated with any number of entities in B and entity in B is associated with at the most one entity in A. The 1 : M relationship is represented using 'parent-child' concept. Here 'one' side of relationship is called parent and 'many' side of relationship is called as child, (Refer Fig. 2.17)

     **For example:** One student takes many subjects (See Fig. 2.18 (a)). Similarly, one manager manages many employees, (Refer Fig. 2.18 (b)).

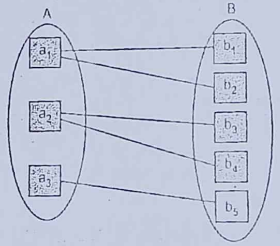


Fig. 2.17: One-to-Many

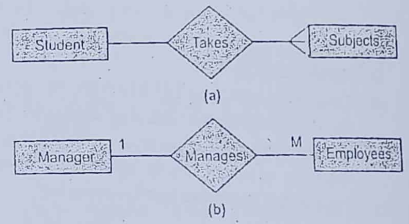


Fig. 2.18: Representation of One-to-Many Relationship

  3. **Many-to-One (M : 1) Relationship:** An entity in A is associated with at most one entity in B and an entity in B is associated with any number of entities in A, (Refer Fig. 2.19).

     **For example:** Many politicians are present in a party [Refer Fig. 2.20 (a)]. Many employees are working in a department, [Refer Fig. 2.20 (b)]. It means one department can have many employees and one employee in one department.

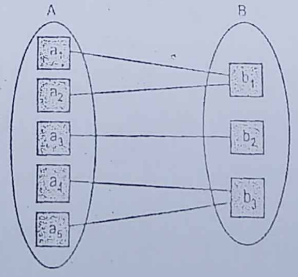


Fig. 2.19: Many-to-One

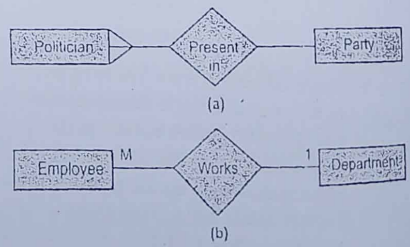


Fig. 2.20: Many-to-One Relationship

4. **Many-to-Many (M : M) Relationship:** Entities in A can be associated with any number of entities in B and vice versa also, (Refer Fig. 2.21).

   **For example:** Many teachers conduct many tests [Refer Fig. 2.22 (a)]. Similarly many doctors works in many Hospitals [see Fig. 2.22 (b)]. Means one doctor in many hospitals and one hospital has many doctors. Therefore, relationship is many-to-many.

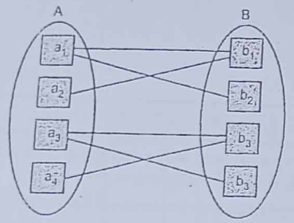



Fig. 2.21: Many-to-Many

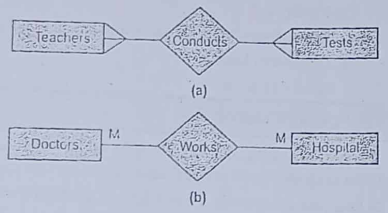



Fig. 2.22: Many-to-Many

- All above are the types of relationship depending upon cardinality. There are three more types of relationship available namely, Binary relationship, Unary relationship and Ternary relationship.
- Let us learn these types of relationships in detail.

1. **Binary Relationship:** It has involvement of two entities only. Therefore the degree of relationship is 2. One entity may be related with 1 : 1, 1 : M, M : M cardinalities with another entity. The number of participating entities in an relationship defines the degree of the relationship. Degree of a relationship is defined as, "the number of entities associated with the relationship".

   **For example:** Many products can be ordered by many customers. Here product and customer are two entities,[Refer Fig. 2.23 (a)]. Similarly, one customer can have many accounts so again two entities involved [Refer Fig. 2.23 (b)] and therefore referred to as binary relationship.

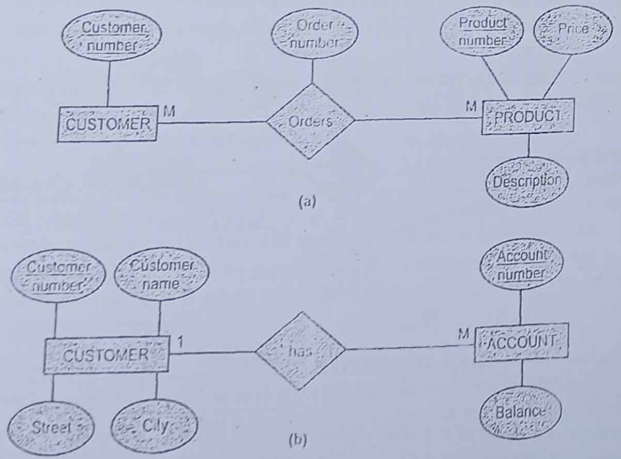



Fig. 2.23: Binary Relationship



In E-R diagram key is represented by the symbol shown in Fig. 2.24.

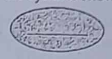


Fig. 2.24: Representation (Primary Key)

2. **Unary Relationships** This represents the association between the occurrence of single entity. Therefore, the degree is 1, (Refer Fig. 2.25).

   For examples: (i) Unary person 1 : 1 person (Marriage).

   (ii) Unary item M : M item (Bill of material).

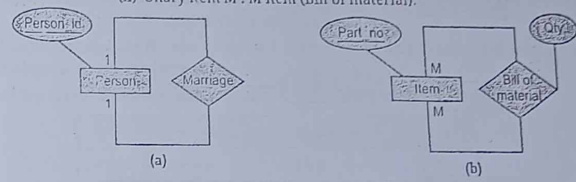



Fig. 2.25: Unary Relationship

3. **Ternary relationship:** It refers to the occurrence of three entities at the same time and degree is ~~six~~ 3 (April 15, Oct. 16)

   For examples:

   (i) An employee works into a branch and his job is say manager. If we write it as binary relationship, then employee_branch and employee job will create some problems for validity, (Refer Fig. 2.26).

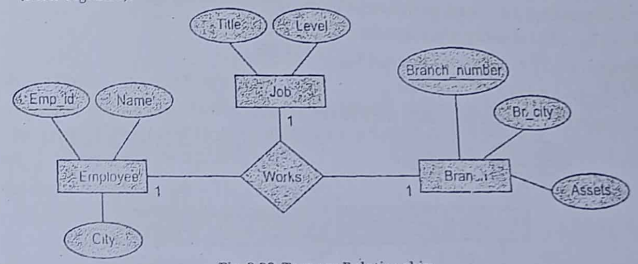



Fig. 2.26: Ternary Relationship

   (ii) Ternary relationship is available with item, vendor and warehouse entities (M : M : M).

   (iii) Ternary relationship is available with customer, branch and account as shown in the Fig. 2.27.

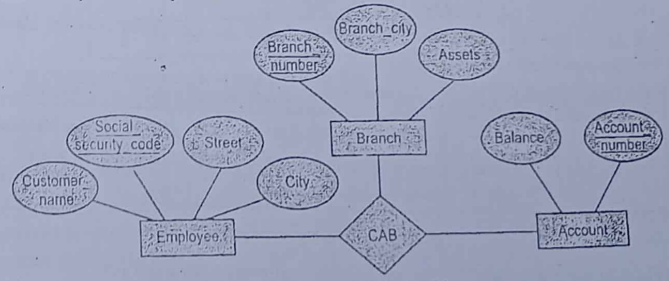



Fig. 2.27: Ternary Relationship

## 2.3.5 Relationship Sets (April 15)

- A relationship is association or linkage among several entities.
- **For example:** The relationship between Employee and Deptartment is shown in the Fig. 2.28 (a).
- The relationship between Customer and Sales order is shown in the Fig. 2.28 (b).
- Similarly the relationship between Customer and Account is shown in the Fig. 2.28 (c).

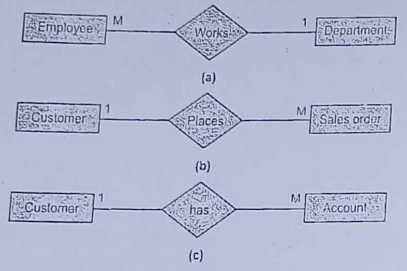



Fig. 2.28: Relationship

- The relationships are represented by diamond shapes. A Relationship Set is a set of relationships of the same type.
- Formally speaking, it is a mathematical relation on $n \geq 2$ possibly non-distinct sets.
- If $E_1, E_2, E_3 \ldots, E_n$ are entity sets, then a relationship set S is a subset of,

  $\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

  where $(e_1, e_2, \ldots, e_n)$ in a relationship.
- For example: Consider the two entity sets doctor and patients, (Refer Fig. 2.29).

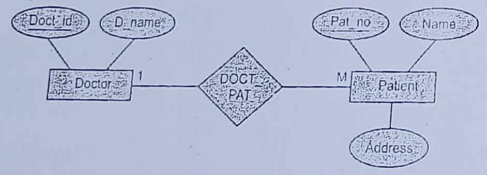



Fig. 2.29: Entity Doctor and Patient

- We can define the relationship set DOCT_PAT to denote the association between above named entities. This represents real-world entities.

| DOCT_ID | D_NAME |
|---|---|
| 01 | Sharma |
| 02 | Sakhare |
| 03 | Kale |

| PAT_NO | NAME | ADDRESS |
|---|---|---|
| 101 | Bina | Pune Wadgaon |
| 102 | Tina | Warje |
| 103 | Rina | |

Fig. 2.30: Relationship DOCT PAT

- From Fig. 2.30, we can say DOCTOR Sharma handles 2 patients i.e., Bina and Tina.



- There are four types of relationship occurs amongst the entities, depending upon the cardinality. Cardinality expresses or shows number of entities in the entity set. Connectivity or Cardinality of a relationship describes the mapping of associated entity instances in the relationship. (April 16)
- Mapping is the process of choosing a logical model and then moving to a physical database file system from a conceptual model (E-R model).
- **For example:** If entity set is Customer – Loan – Bank

  Cardinality is 3.
- Even though generally we consider binary relationships only.

  i.e. Customer - Loan

  Loan - Bank etc.

## 2.4 CONSTRAINTS

- The entities, entity sets, relationship, attributes etc., are very important while designing E-R model. Other than this there are various other important constraints which are required in E-R model.
- In database constraints are restrictions on the contents of the database or on database operations.

### 2.4.1 Overview of Keys Constraints

- Keys are fundamental components of any relational database. Relational database will not be useful without keys.
- A key is an attribute or set of attributes that is used to identify data in entity sets. The attributes which are used as key are known as key attributes and rest of all are known as non-key attributes.
- Sometimes, an additional constraint exists for a given relationship set, that any entity from one of the associated sets appears in at most one such relationship.
- For example, consider a relationship set "Manages" which associates departments with employees. If a department cannot have more than one manager, this is an example of a one-to-many relationship set (it may be that an individual manages multiple departments). This type of constraint is called a key constraint.
- Key constraint is represented in the E-R diagrams by drawing an arrow from an entity set E to a relationship set R when each entity in an instance of E appears in at most one relationship in (a corresponding instance of) R.

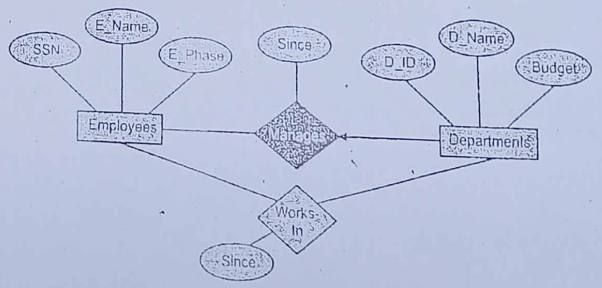



Fig. 2.31: Example of Key Constraint

**Concept of Key:**

- Key is an attribute or a group of attributes whose values can be used to uniquely identify individual entity in an entity set. For example, for the following entity:

  STUDENT : {RollNo, Name, Address} RollNo attribute is a key.

- Other possible key might be combination of {Name, Address}.

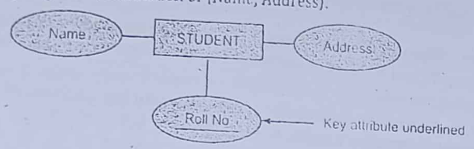



Fig. 2.32: Key Notation in RDBMS

- For example:

Employee

| Reg. No. | ID | Name | Salary | Dept_ID |
|---|---|---|---|---|
| A01 | 1 | Amar | 15,000 | 2 |
| B02 | 2 | Atharva | 20,000 | 4 |
| C03 | 3 | Vikas | 30,000 | 2 |
| D04 | 4 | Madhu | 10,000 | 1 |
| E05 | 5 | Deepa | 50,000 | 4 |

Department

| Dept_ID | Dept_Name |
|---|---|
| 1 | Sales |
| 2 | Marketing |
| 4 | Development |

Fig. 2.33: Entity Sets Employee and Department

**Primary Key:**

- An attribute which identifies data uniquely is known as primary key.
- Any entity set can have more than one candidate key but only one primary key.
- For example, in entity set Employee, either Reg. No. is primary key or ID is primary key.

**Foreign Key:**

- A foreign key is an attribute in any entity set which is also a primary key in any other entity set.
- For example, Dept_ID is an attribute in entity set Employee and also a primary key in entity set Department. Thus, it is a foreign key in Employee.

## 2.4.2 Integrity Constraints

(Oct. 16, 17, April 17)

- Integrity is related with a data into a database. A database constraint is a condition specified on a database schema that restricts the data to be inserted in an instance of the database.
- The term data integrity refers to the correctness and completeness of the data. In any relational database we have collection of related tables. So RDBMS tries to preserve the integrity of data by providing integrity constraints.

### 2.4.2.1 Key Constraints

(April 15)

- Constraints are rules that are used to control the invalid data entry in a column.
- A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.
- There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called key for that relation.



- Key constraints forces that:
  1. in a relation with a key attribute, no two tuples can have identical value for key attributes.
  2. key attribute cannot have NULL values.
- Key constrains are also referred to as entity integrity constraints. By definition all entities must be distinguishable that is they must have unique identification property. A primary key performs the unique identification function in a relational database.
- Keys are attributes or sets of attributes that uniquely identify an entity within its entity set. An entity set E can have multiple keys out of which one key will be designated as the primary key. Primary key must have unique and not null values in the relational table.
- In a subclass hierarchy, only the root entity set has a key or primary key and that primary key must serve as the key for all entities in the hierarchy.
- Example of key constraints in a simple relational table is shown in Fig. 2.34.

| SID | Name | Class (semester) | Age |
|---|---|---|---|
| 8001 | Ankit | 1st | 19 |
| 8002 | Srishti | 1st | 18 |
| 8003 | Somvir | 4th | 22 |
| 8004 | Sourabh | 6th | 45 |
| 8002 | Tony | 5th | 23 |

Not allowed as Primary key values must be unique

Fig. 2.34

1. **Integrity Rule 1 (Entity Integrity Rule or Constraint):** (April 16)

- This rule states that no attribute of primary key will contain a null value. If a relation have a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained.
- Consider the example shown in Fig. 2.35.

| SID | Name | Class (semester) | Age |
|---|---|---|---|
| 8001 | Ankit | 1st | 19 |
| 8002 | Srishti | 2nd | 18 |
| 8003 | Somvir | 4th | 22 |
| | Sourabh | 6th | 19 |

Not allowed as Primary key cannot contain a NULL value

Fig. 2.35

2. **Integrity Rule 2 (Referential Integrity Rule or Constraint):** (April 17)

- Referential integrity defines that a foreign key must have a matching primary key.
- This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables.
- It means the reference from a row in one table to other table must be valid.

- For example, in Fig. 2.36, if a foreign key in Table 1 refers to the primary key of Table 2, then every value of the foreign key in Table 1 must be available in Table 2.

Table 1

| ENO | Name | Age | D_No |
|---|---|---|---|
| 1 | Ankit | 19 | 10 |
| 2 | Srishti | 18 | 11 |
| 3 | Somvir | 22 | 14 |
| 4 | Sourabh | 19 | 10 |

Foreign Key

Not allowed as D_No 14 is not defined as a primary key of Table 2, and in Table 1, D_No is a foreign key defined

Relationships

Table 2

| Primary Key → D_No | D_Location |
|---|---|
| 10 | Rohtak |
| 11 | Bhiwani |
| 12 | Hansi |

Fig. 2.36

- Let the table in which the foreign key is defined is foreign table or details table i.e. Table 1 in above example and the table that defines the primary key and is referenced by the foreign key is master table or primary table i.e., Table 2 in above example.

## 2.4.2.2 Primary Key Constraint

- Primary key constraint uniquely identifies each row record in a database. A primary key must contain unique value and it must not contain NULL value.
- Usually primary key is used to index the data inside the table. To define the primary key, you must use the PRIMARY KEY constraint to specify which column or columns will serve as the table's primary key.
- If you want to add a PRIMARY KEY constraint to a column definition. Use the following syntax:

```
<column name> { <data type> | <domain> } PRIMARY KEY
```

- If you want to add a PRIMARY KEY constraint as a table constraint, you must add it as a table element in the table definition, as shown in the following syntax:

```
[CONSTRAINT <constraint name> ]
PRIMARY KEY (<column name> [{,<column name>}....])
```

- Example, using PRIMARY KEY constraint at Table Level:

```
CREATE table Student (s_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);
```

- The above command will creates a PRIMARY KEY on the s_id.
- Example, using PRIMARY KEY constraint at Column Level:

```
ALTER table Student add PRIMARY KEY (s_id);
```

- The above command will creates a PRIMARY KEY on the s_id.

## 2.4.3 Referential Integrity (Oct. 18, April 15, 18)

- Referential integrity is specified between two relations and is used to maintain the consistency among tuples of the two relations.
- The referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.



- To define referential integrity, we must first define the concept of a foreign key, which we have already defined.
- The conditions for a foreign key, given below, specify a referential integrity constraint between the two relation schemas $R_1$ and $R_2$. A set of attributes FK in relation schema $R_1$ is a foreign key of $R_1$ if it satisfies the following two rules.
  1. The attributes in FK have the same domain as the primary key attributes PK of another relation schema $R_2$; the attributes FK are said to reference or refer to the relation $R_2$.
  2. A value of FK in tuple $t_1$ of $R_1$ either occurs as a value of PK for same tuple $t_2$ [PK], (if not null), and we say that the tuple $t_1$ references or refer to the tuple $t_2$.

**Example:** Consider the example of Employee and Manager relation. Managers are also employees. Emp_No is a primary key of employee. The manager_no attribute represents the employee number of the manager. Manager_no is a foreign key; note that it is referring to the primary key of the same relation (See Fig. 2.37 (a)).

Employee

| Emp_No | Name | Manager_No |
|---|---|---|

Manager

| Manager_no | name |
|---|---|

⇒

| Emp_No | Name | Manager_No |
|---|---|---|
| 101 | Jones | @ |
| 103 | Smith | 110 |
| 104 | Lalonde | 107 |
| 107 | Evan | 110 |
| 110 | Drew | 112 |
| 112 | Smith | 112 |

Fig. 2.37 (a)

**Example:**

EMPLOYEE

| EMP_NO | EMP_NAME | ADDR | SALARY | D_NO |
|---|---|---|---|---|

DEPARTMENT

| D_NAME | D_NUMBER | NO_PROJECT |
|---|---|---|

DEPT_LOCATIONS

| D_NUMBER | D_LOCATION |
|---|---|

Fig. 2.37 (b)

- In Fig. 2.37 (b) primary keys are underlined. In the EMPLOYEE relation, the attribute D_NO refers to the department for which an employee works, hence D_NO to be a foreign key of EMPLOYEE, referring to the DEPARTMENT relation.
- This means that a value of D_NO in any tuple $t_1$ of EMPLOYEE relation must match a value of the primary key of DEPARTMENT in some tuple $t_2$ of the DEPARTMENT relation, or the value of D_NO can be null if the employee does not belongs to a department.

## 2.4.4 Unique Constraint

- The unique constraint uniquely identifies each record in a database table. The unique and primary key constraints both assure for uniqueness of a column or set of columns.
- A primary key constraint automatically has a unique constraint defined on it. This constraint can be applied at column level or table level.

- Unique constraint ensures that a field or column will only have unique values. A Unique constraint field will not have duplicate data. Unique constraint can be applied at column level or table level.
- To create a column constraint, add it as part of the column definition, as shown in the following syntax:

```
<column name> { <data type> | <domain> } UNIQUE
```

- Example, using UNIQUE constraint when creating a Table (Table Level):

```
CREATE table Student(s_id int NOT NULL UNIQUE, Name varchar(60), Age int);
```

- The above query will declare that the s_id field of Student table will only have unique values and wont take NULL value.
- If you want to add a unique constraint as a table constraint, you must add it as a table element in the table definition, as shown in the following syntax:

```
[CONSTRAINT <constraint name> ]
UNIQUE(<column name> [{,<column name>}....])
```

- Example, using UNIQUE constraint after Table is created (Column Level):

```
ALTER table Student add UNIQUE(s_id);
```

- The above query specifies that s_id field of Student table will only have unique value.
- Difference between Primary Key and Unique Constraints:
  1. A table can have only one primary key, but it can have many unique constraints.
  2. When a primary key is defined, the column that compose the primary key are automatically mandatory. But with the unique constraint you must specify the column as not null.

## 2.4.5 NULL/NOT NULL Constraint

- NULL means unknown value. A NULL value in a relational database is a special code that can be placed in a column that indicates that the value for that column in that row is unknown.
- Constraints on a column may also be in the form of NULL constraints. A NULL constraint just indicates that the column may or may not be NULL. It is defined when the column is declared as part of the table.

**NOT NULL Constraint:**

- NOT NULL constraint restricts a column from having a null value.
- When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By applying Null constraint, we can be sure that a particular column(s) cannot have NULL values. This constraint can be applied at column level only, not for table level.
- NOT NULL constraint restricts a column from having a NULL value.
- Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value. One important point to note about NOT NULL constraint is that it cannot be defined at table level.
- It is a very straightforward process. Simply use the following syntax when creating a column definition:

```
<column name> { <data type> | <domain> } NOT NULL
```

- Example, using NOT NULL constraint:

```
CREATE table Student(s_id int NOT NULL, Name varchar(60), Age int);
```

- The above query will declare that the s_id field of Student table will not take NULL value.

## 2.4.6 Domain Constraint

- A table in DBMS is a set of rows and columns that contain data. Columns in table have a unique name, often referred as attributes in DBMS. A domain is a unique set of values permitted for an attribute in a table.



- For example, a domain of week days can accept Monday, Tuesday .... Sunday as possible values, a domain of integers can accept whole numbers that are negative, positive and zero. It specifies that the value taken by the attribute must be the atomic value from its domain.
- Domain restricts the values of attributes in the relation. Domain constraints specifies that what set of values an attribute can take.
- Value of each attribute X must be an atomic value from the domain of X. The data type associated with domains include integer, character, string, date, time, currency etc.
- An attribute value must be available in the corresponding domain. Consider the example in Fig. 2.38.

| SID | Name | Class (semester) | Age |
|---|---|---|---|
| 8001 | Ankit | 1st | 19 |
| 8002 | Srishti | 1st | 18 |
| 8003 | Somvir | 4th | 22 |
| 8004 | Sourabh | 6th | A ← Not allowed, because Age is integer attribute |

Fig. 2.38

- Domain integrity constraint are classified into two types i.e. Not Null Constraint and Check Constraint, explained in Sections 2.4.5 and 2.4.7.

## 2.4.7 Check Constraint

- Check constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database.
- It's like condition checking before saving data into a column. This constraint can be apply on a single column, or multiple columns of the table.
- Check constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database.
- It's like condition checking before saving data into a column.
- To create a column CHECK constraint, use the following syntax in a column definition:

```
<column name> { <data type> | <domain> } CHECK (<search condition>)
```

- To create a table CHECK constraint, use the following syntax in a table definition:

```
[CONSTRAINT <constraint name> ] CHECK (<search condition>)
```

- Example, using CHECK constraint at Table Level:-

```
CREATE TABLE Student(s_id int NOT NULL CHECK(s_id > 0),
Name varchar(60) NOT NULL,
Age int);
```

- The above query will restrict the s_id value to be greater than zero.
- Example, using CHECK constraint at Column Level:

```
ALTER TABLE Student add CHECK(s_id > 0);
```

## 2.4.8 Mapping Constraints (April 16)

- There are two types of mapping constraints namely, Mapping cardinalities and Participation constraints.

1. Mapping Cardinalities (Cardinality Ratios):

- It specifies the number of entities of an entity set that are associated with entities of another entity set through a relationship set.

- *Mapping cardinalities means expressing the number of entities to which another entity is associated with. They are most useful while describing binary relationship sets. We have already seen them in Section 2.3.4.*
- *In this section we will see the examples only.*

**(a) One-to-One:** In this cardinality mapping is done between at most one entity.

- For example:
  - (i) College – Principal
  - (ii) Country – President
  - (iii) Husband – Wife.

**(b) One-to-Many:** Here one entity is associated with any number of entities:

For example:

(i) One principal – Handle many employees.

(ii) One doctor – Treats many patients.

**(c) Many-to-One:** Here many entities are associated with a single entity.

For example:

(i) Many colleges – Affiliated to one university

(ii) Many books – Placed in one shelf.

**(d) Many-to-Many:** Here many entities (zero or more) are associated with many entities.

For example:

(i) Many teachers – Teachers – Many students.

**2. Participation Constraint:**

- Participation constraints tell the participation of entity set in relationship sets. The participation constraint can be total or partial.
- There are two types of participations as explained below:

**(i) Total Participation:** Each entity is involved in the relationship. Total participation is represented by double lines, (Refer Fig. 2.39).

**(ii) Partial Participation:** Not all entities are involved in the relationship. Partial participation is represented by single lines, (Refer Fig. 2.39).

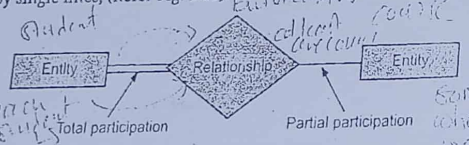



Fig. 2.39

## 2.5 EXTENDED FEATURES OF E-R MODEL (April 16, 17)

- *The basic E-R concept can model most of the database features.But same aspect of database can be expressed by adding some extra features called extended features.*
- *The basic E-R modeling concepts were not able to represent the requirements of newer and more complex applications. So, the need to search for and develop more powerful semantic modeling concepts was felt by the database designers.*
- The E-R model that is supported with the additional semantic concepts is called the Enhanced Entity-Relationship (EER) Model. The additional concepts are specialization, generalization and aggregation.



### 2.5.1 Specialization (↓)

- It is the abstracting process of introducing new characteristics of objects to create one or more new classes of objects.
- In simple words specialization is a result of taking a subset of higher level entity set to form a lower level entity set, (Refer Fig. 2.40).
- Specialization is defined as, "the process of identifying subsets of an entity (super-class or super-type) that share some distinguishing characteristic".
- Entities are expanded in specialization. In the Fig. 2.40 line implies doctor can also be anything else apart from permanent doctor and consulting doctor.

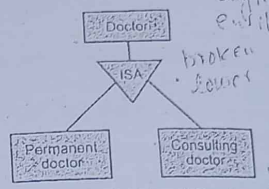



Fig. 2.40: Specialization

### 2.5.2 Generalization (↑) (April 15)

- It is abstracting process of viewing set of objects as a single general class.
- Generalization is defined as "the process of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities processing these common features".
- It concentrates on the general characteristics of consequent set while suppressing or ignoring their differences. It means generalization is the result of taking the union of two or more entity sets to produce a higher level entity set, (Refer Fig. 2.41).
- In the Fig. 2.41 the line implies that a doctor has to be either a Permanent doctor or a Consulting doctor, nothing else. So the entities are limited in generalization.

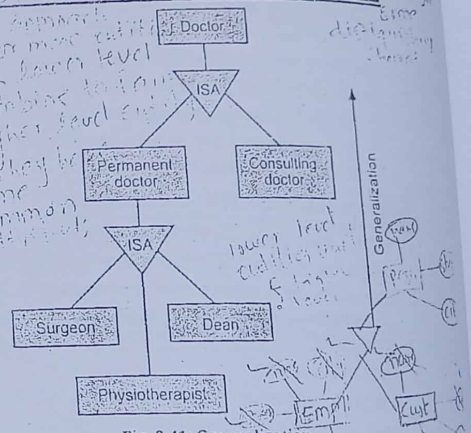



Fig. 2.41: Generalization

**Difference between specialization and generalization:** (April 17, Oct. 17)

| Sr. No. | Generalization | Specialization |
|---|---|---|
| 1. | It is bottom-top design process. | It is top-down design process. |
| 2. | It is union of two or more lower level entity sets to produce higher level entity set. | It is subset of higher level entity set to form lower level entity set. |
| 3. | In this, lower level entity sets are also described by attributes and relationship of higher level entity sets. | Lower level entity sets may have different attributes and may participate in relationship that do not apply higher level entity set. |
| 4. | It is used to emphasize similarities among lower level entity types. | It is used to emphasize distinction between higher level and lower level entity set. |
| 5. | In this, every higher level entity must also be a lower level entity. | It allows for the possibility of more low level entity sets also. |

### 2.5.3 Aggregation

April-16, 17, Oct.-17

- It is the process of combining information on a object so that the higher level objects can be abstracted. Aggregation is defined as, "the process of compiling information on an object, thereby abstracting a higher-level object".
- In any entity relationship diagram, we cannot form a relationship among the entities and their relationship. Aggregation is the process which allows to do so, (Refer Fig. 2.42).
- In Fig. 2.42 many Doctors Gives visits to many Rooms and everything is done into the same Hospital.

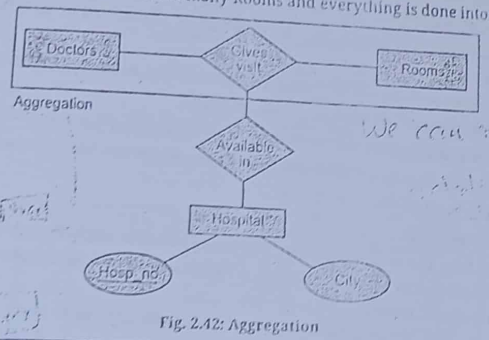



Fig. 2.42: Aggregation

## 2.6 PICTORIAL REPRESENTATION OF E-R (SYMBOLS)

Oct.-16

- The pictorial/graphical representation of E-R model is Entity-Relationship Diagram (ERD). An ERD is a pictorial representation of the entities and the relationship between them. From this, user can use the information structure of the application at a glance.
- Afterwards these ERD's are used to design tables and databases. An ERD serves several purposes a listed below:
  1. ERD is used to communicate the logical structure of the database to the end users.
  2. ERD helps the database designer in understanding the information to be contained in the database.
  3. ERD serves as a documentation tool.
- Symbols used in E-R diagrams are shown in Table 2.1.

**Table 2.1: Symbols used in E-R Diagram**

| Sr. No. | Name of Symbol | Symbol | Meaning |
|---|---|---|---|
| 1. | Rectangle | | Entity / Entity Set (Strong) |
| 2. | Double Rectangle | | Entity Set (Weak) |
| 3. | Ellipse | | Attribute |
| 4. | Diamond | | Relationship / Relationship Set |

*Contd...*



| | | | |
|---|---|---|---|
| 5. | Double Diamond | | Identifying Relationship Type |
| 6. | Double Ellipses | | Multi-valued Attributes |
| 7. | Dashed Ellipses | | Derived Attributes |
| 8. | Ellipse with line inside it | | Key Attributes |
| 9. | Ellipse joined with other ellipses | | Composite Attributes |
| 10. | Double lines | $E_1$ R $E_2$ | Total Participation |
| 11. | Single line | $E_1$ R $E_2$ | Partial Participation |
| 12. | Triangle | ISA | Specialization or Generalization |

- For example: Consider the example of a database that contains information on the residents of a city. The E-R diagram shown in the Fig. 2.43. It contains two entities person and city. There is a single "Lives In" relationship. Each person lives in only one city, but each city can has many persons.
- E-R modeling is a data modeling technique used in software engineering to produce a conceptual data model of a information system.

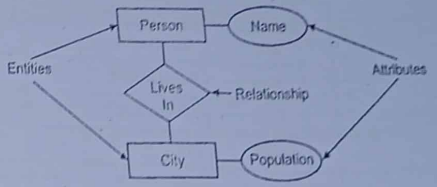



Fig. 2.43

**Advantages of ERD:**

1. ERD is easy to understand and simple to create.
2. ERD provide visual. (graphical) representation of database design.
3. Using ERD reduced data redundancy.

**Disadvantages of ERD:**

1. ERD has no universal standards.
2. ERD has limited constraints and relationship representations.
3. ERD has no representation of data manipulation.

## 2.7 STRUCTURE OF RELATIONAL DATABASE

April-19

- The relational model is an abstract theory of data. It is based on mathematical theory. It was suggested by Dr. E. F. Codd.
- The relational database management system is based on relational model. This model tells us how to represent and manipulate data.

- A relational database is a collection of relations (or two-dimensional tables) having distinct names. A relational database is created using the relational model. The software used in a relational database is called a Relational Database Management System (RDBMS).
- A RDBMS is a software package that stores data into the database based on the relational data model and supports relational database languages for retrieving and manipulating data in the database.
- RDBMS is the basis for SQL and for all model database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The Relational Data Model (RDM) stores the data in the form of a table. Each table consists of multiple columns and each column is recognized by a unique name.
- In RDBMS relational data model is the primary data model, which is used widely for data storage and processing.
- Relational model is simple and have all the properties and capabilities required to process data with storage efficiency.
- Fig. 2.44 shows the major components of the relational database structure.

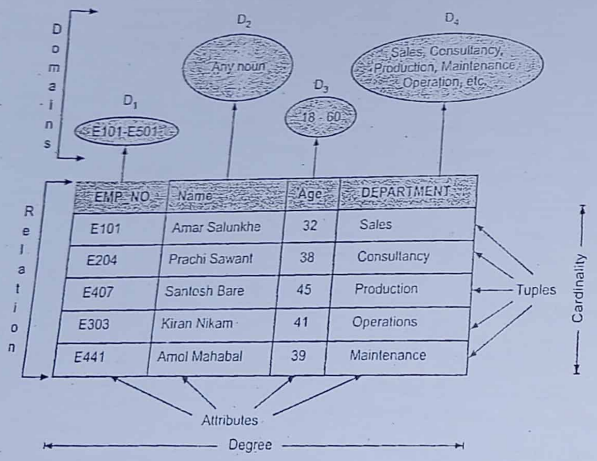



Fig. 2.44

- Fig. 2.44 shows following components or elements of relational database structure:

1. **Tables:** The data in RDBMS is stored in database objects called tables. The table is a collection of related data entries and it consists of columns and rows. In RDBMS table rows represent records and table column represents the attributes.
2. **Relation:** A relation is defined as, "a table with column and rows". According to E. F. Codd, data can be stored in form of two-dimensional table.
3. **Tuple:** A single row of a table, which contains a single record for that relation is called a tuple. A record also called a row of data, is each individual entry that exists in a table.
4. **Column:** A column is a vertical entity in a table that contains all information associated with a specific field in a table.
5. **Attribute:** It is a named column of a relation.



6. **Domain:** It is a set of allowed values for one or more attributes. Every attribute in a relation is defined on a domain. (April 16)
7. **Extension (or State) of a Relation:** The extension of a relation is the set of tuples or records that appear in that relation at any given instant of time.
8. **Degree:** The degree of a relation is the number of attributes it contains.
9. **Cardinality:** The cardinality of a Relation is the number of tuples it contains. So, it changes as we add new tuples or delete them.
10. **Relational Database:** It is defined as, "the collection of normalized or structured relations with distinct relation names".
11. **Relation Instance:** A finite set of tuples in the relational database system represents relation instant. Relation instances do not have duplicate tuples.
12. **Relation Schema:** It describes the relation name (table name), attributes and their names.
13. **Relation Key:** Each row has one or more attributes which can identify the row in the relation (table) uniquely, is called the relation key.

    Relational keys includes:

    (i) **Primary Key:** The attribute (column) or combination of attributes which uniquely identifies each tuple (row) of a relation (table) is called a primary key. Primary key cannot contain duplicate or null values. (April 18)

    (ii) **Composite Key:** When primary key is a combination of two or more attributes, it is known as composite key. Composite key uniquely identifies each tuple of a relation. (April 18)

    (iii) **Super Key:** An attribute or combination of attributes that uniquely identify each record in a table is called a super key. Super key is a super set of primary keys. (April 18, Oct.-19)

    (iv) **Candidate Key:** A super key without redundancies or a minimal super key, is called a candidate key. The candidate key uniquely identifies each record of a table. In a table, there may be more than one candidate keys. After identifying the candidate keys, any one of the appropriate candidate key is selected as a primary key. The candidate keys have unique values. (April 17)

    (v) **Alternate Key:** The candidate keys, which are not a primary key, are called alternate key.

    (vi) **Unique Key:** The key which contains unique values id known as unique key. Unique key accepts only unique values, but it accepts null values also.

    (vii) **Foreign Key:** An attribute or combination of attributes, in one table, whose value must either match the primary key in another table or be null is known as foreign key. The field or combination of fields can become foreign key, if and only if, it is a primary key of another table.

    (viii) **Secondary Key:** The attribute or combination of attributes which are used for retrieval purpose is known as secondary key.

    (ix) **Surrogate Key:** The artificial primary key, is called as surrogate key.

14. **Attribute Domain:** Every attribute has some pre-defined value scope, known as attribute domain.

## 2.7.1 Concept of Table

- The data in relational database is stored in database objects called tables. The table is a collection of related data entries and it consists of columns and rows.
- In RDBMS table rows represent records and table column represents the attributes. A table is a collection of rows and columns.

- The collection of all records is called a table. A record has set of attributes (fields) and a possible value is defined for each attribute. Records of each type forms a table or relation, (Refer Fig. 2.45).
- The degree of a relation is the number of attributes in the table.
- A one-attribute table is a unary relation; a two-attribute table is a binary relation and a n-attribute table is a ternary relation, (Refer Fig. 2.45).

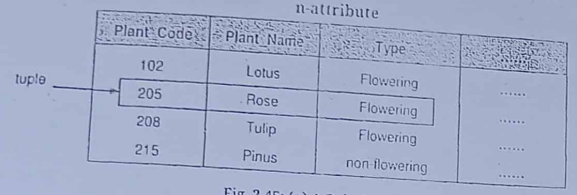



Fig. 2.45: (a) A Relation (Table)

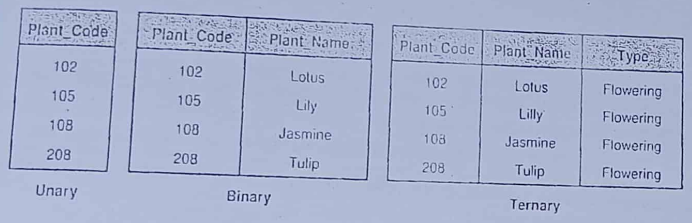



Fig. 2.45: (b) Degree of a Relation

## 2.7.2 Rows (Tuples) and Domains

(April 17, Oct. 17)

- This is a normal practice that when we arrange data in a tabular format then everyone understands it easily. This means that the data is arranged in two-dimensional format.
- So the table should have following characteristics:
  1. Repeating rows are not there.
  2. Column has same data items with same type.
  3. Each column has distinct name by using which we identify the table.
- A row in a table is called tuple or a record for us. A relation is a set of tuples. The number of rows in a table is called cardinality of the table and the number of columns in a table is called as its degree, (Refer Fig. 2.46).
- A domain is a set of all possible values for a column.

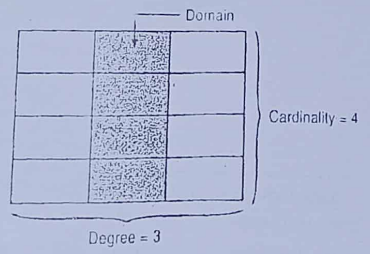



Fig. 2.46: Degree and Cardinality of a Table

## 2.7.3 Relations

(Oct. 16)

- The rows are unordered in a relation and cannot be identified by its position in a relation.
- Every table must have some columns or combination of columns which uniquely identify each row in the table. For that, we require a key. A key is simply a field used to identify a record.



- In other words, a key is a relation of subset of attributes with following properties:
  1. The value of key is unique for each tuple.
  2. No data redundancy.
- As we know, a key is an identifying property of a database, to remove confusion in modeling, the type of key should always be stated according to purpose.
- In a database every entity and the relationship between them is distinct. And this difference is expressed in terms of their attributes. The concept of key allows to make such distinction between two or more entities.

## 2.8 DBMS VERSUS RDBMS

- Following table compares DBMS and RDBMS:

| Sr. No. | DBMS | RDBMS |
|---|---|---|
| 1. | DBMS is a software that is used to define, create and maintain a database and provides controlled access to the data. | It is an advanced version of a DBMS. RDBMS is based on the relational model of data, as proposed by E. F. Codd in 1970. |
| 2. | Data is stored in the navigational or hierarchical model (in files). | Data is stored in the relational model (in tables). |
| 3. | Data access require more time. | Data access is faster as compared to DBMS. |
| 4. | It supports single user. | It supports multiple users. |
| 5. | It deals with small quantity of data. | It deals with large amount of data. |
| 6. | It supports 3 rules of E.F. Codd out off 12 rules. | It supports minimum 6 rules of E.F. Codd. |
| 7. | Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| 8. | Distributed Databases are Not supported. | Distributed Databases are supported. |
| 9. | In DBMS it is difficult to recover the database in case of loss of data. | RDBMS helps in recovery of the database in case of loss of data. |
| 10. | The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| 11. | Keys and indexes are Not used in DBMS. | To establish relationship keys and indexes are used in RDBMS. |
| 12. | Examples: Dbase, Microsoft Access, FoxPro. | Examples: SQL Server, Oracle, MySQL, PostgreSQL. |

## 2.9 CASE STUDIES ON E-R MODEL

(April 14, 15, 16, 17, Oct. 16, 17)

- Solved case studies are included to show how E-R model is used for conceptual design.

**Case Study 1:** Consider a trucking company which is responsible for picking up shipments for warehouses of a retail chain and deliver the shipments to the individual store location. A truck may carry several shipments in a single trip and deliver it to multiple stores. Draw an E-R diagram for truck-shipment system.

- The collection of all records is called a table. A record has set of attributes (fields) and a possible value is defined for each attribute. Records of each type forms a table or relation, (Refer Fig. 2.45).
- The degree of a relation is the number of attributes in the table.
- A one-attribute table is a unary relation; a two-attribute table is a binary relation and a n-attribute table is a ternary relation, (Refer Fig. 2.45).

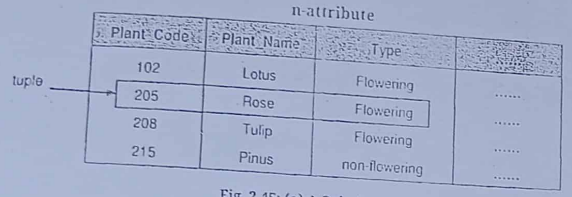



Fig. 2.45: (a) A Relation (Table)

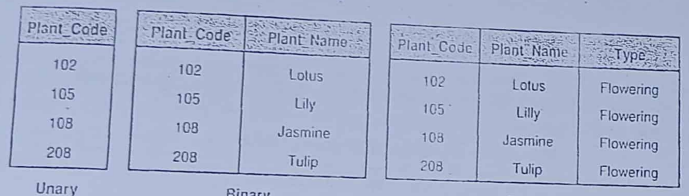



Fig. 2.45: (b) Degree of a Relation

## 2.7.2 Rows (Tuples) and Domains

(April 17, Oct. 17)

- This is a normal practice that when we arrange data in a tabular format then everyone understands it easily. This means that the data is arranged in two-dimensional format.
- So the table should have following characteristics:
  1. Repeating rows are not there.
  2. Column has same data items with same type.
  3. Each column has distinct name by using which we identify the table.
- A row in a table is called tuple or a record for us. A relation is a set of tuples. The number of rows in a table is called cardinality of the table and the number of columns in a table is called as its degree, (Refer Fig. 2.46).
- A domain is a set of all possible values for a column.

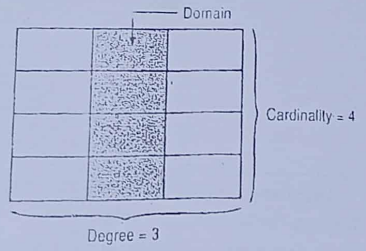



Fig. 2.46: Degree and Cardinality of a Table

## 2.7.3 Relations

(Oct. 10)

- The rows are unordered in a relation and cannot be identified by its position in a relation.
- Every table must have some columns or combination of columns which uniquely identify each row in the table. For that, we require a key. A key is simply a field used to identify a record.



- In other words, a key is a relation of subset of attributes with following properties:
  1. The value of key is unique for each tuple.
  2. No data redundancy.
- As we know, a key is an identifying property of a database, to remove confusion in modeling, the type of key should always be stated according to purpose.
- In a database every entity and the relationship between them is distinct. And this difference is expressed in terms of their attributes. The concept of key allows to make such distinction between two or more entities.

## 2.8 DBMS VERSUS RDBMS

- Following table compares DBMS and RDBMS:

| Sr. No. | DBMS | RDBMS |
|---|---|---|
| 1. | DBMS is a software that is used to define, create and maintain a database and provides controlled access to the data. | It is an advanced version of a DBMS. RDBMS is based on the relational model of data, as proposed by E. F. Codd in 1970. |
| 2. | Data is stored in the navigational or hierarchical model (in files). | Data is stored in the relational model (in tables). |
| 3. | Data access require more time. | Data access is faster as compered to DBMS. |
| 4. | It supports single user. | It supports multiple users. |
| 5. | It deals with small quantity of data. | It deals with large amount of data. |
| 6. | It supports 3 rules of E.F. Codd out off 12 rules. | It supports minimum 6 rules of E.F. Codd. |
| 7. | Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| 8. | Distributed Databases are Not supported. | Distributed Databases are supported. |
| 9. | In DBMS it is difficult to recover the database in case of loss of data. | RDBMS helps in recovery of the database in case of loss of data. |
| 10. | The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| 11. | Keys and indexes are Not used in DBMS. | To establish relationship keys and indexes are used in RDBMS. |
| 12. | Examples: Dbase, Microsoft Access, FoxPro. | Examples: SQL Server, Oracle, MySQL, PostgreSQL. |

## 2.9 CASE STUDIES ON E-R MODEL

(April 14, 15, 16, 17, Oct. 16, 17)

- Solved case studies are included to show how E-R model is used for conceptual design.

**Case Study 1:** Consider a trucking company which is responsible for picking up shipments for warehouses of a retail chain and deliver the shipments to the individual store location. A truck may carry several shipments in a single trip and deliver it to multiple stores. Draw an E-R diagram for truck-shipment system.

**Solution:** Fig. 2.47 shows E-R diagram (model) fro Truck Shipment System.

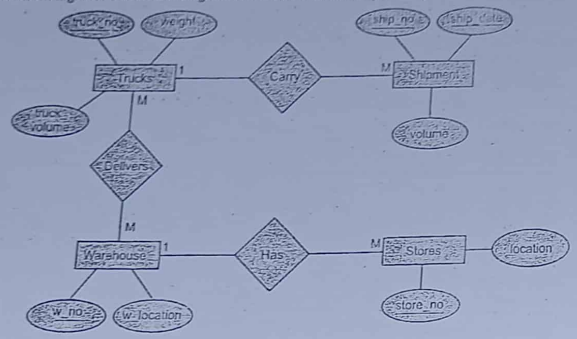



Fig. 2.47: Truck Shipment System

**Case Study 2:** Draw an E-R diagram for airlines reservation system. Here a passenger can book ticket from personnel for a flight on some date.

**Solution:** Fig. 2.48 shows E-R diagram (model) for Airline Reservation System.

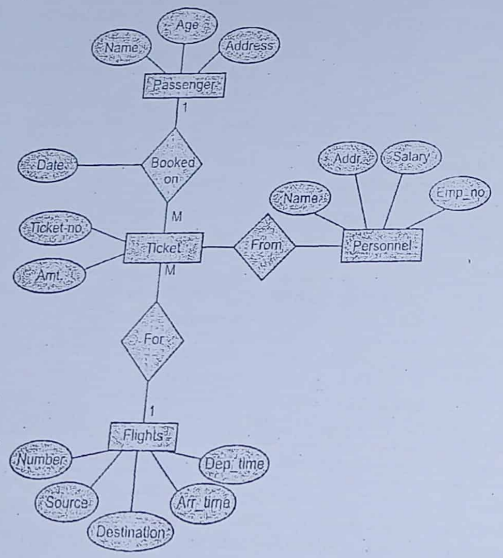



Fig. 2.48: Airlines Reservation System



**Case Study 3:** Construct an E-R diagram for a car insurance company that has a set of customers. Each customer owns one or more cars. Each are associated with more cars. Each can be associated with zero to any number of recorded accidents.

**Solution:** Fig. 2.49 shows E-R diagram for Car Insurance System.

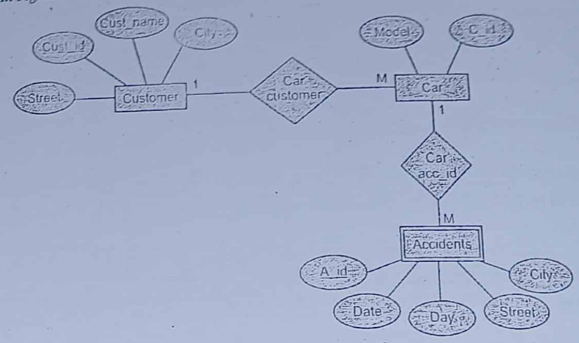



Fig. 2.49: Car Insurance System

**Case Study 4:** Draw an E-R diagram for order processing system where a person can give order for many items by specifying its quantity.

**Solution:** Fig. 2.50 shows E-R diagram for Order Processing System.

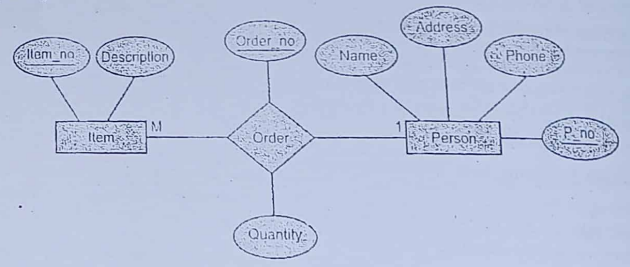



Fig. 2.50: Order Processing System

**Case Study 5:** A movie studio wishes to institute a database to manage their files of movies actors and directors. The following facts are relevant.

(i) Each actor has appeared in many movies.

(ii) Each director has directed many movies.

(iii) Each movie has one director and one or more actors.

(iv) Each actor and director may have several addresses.

Draw E-R diagram.

**Solution:** Fig. 2.51 shows E-R diagram for Movie Studio Institute.

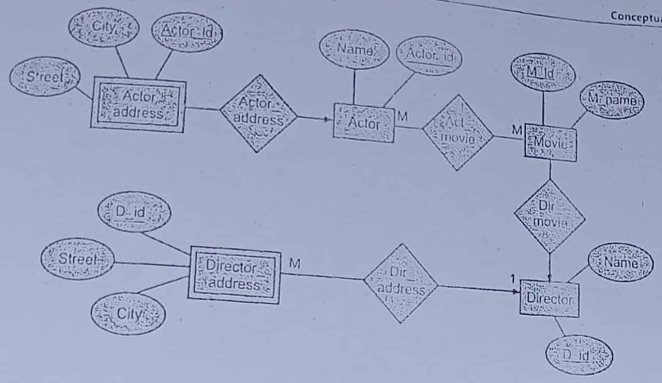



Fig. 2.51

**Case Study 6:** Assume you are to compose database requirements of a wholesale dealer for audio, video consumer equipment from different manufacturers. Customers are the various retail outlets (Retailers). Wholesaler extends credit to old customers (retailers). Draw E-R model.

**Solution:** Fig. 2.52 shows E-R digram for Wholesale Dealer.

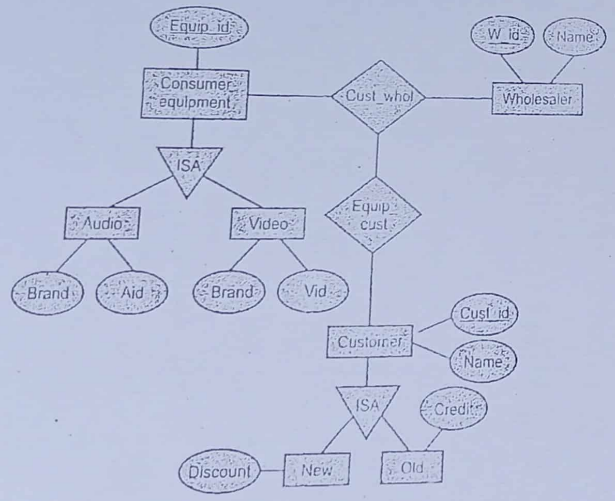



Fig. 2.52: Wholesale Dealer

**Case Study 7:** Draw E-R model for Hospital Management System.

**Solution:** Fig. 2.53 shows E-R model for Hospital Management System.



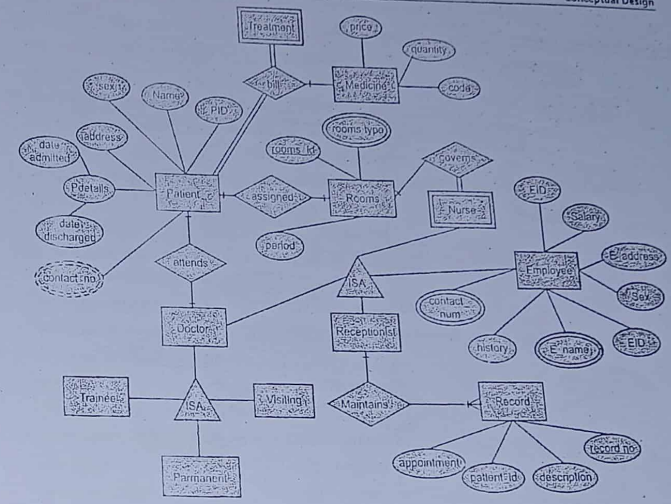



Fig. 2.53

**Case Study 8:** Draw E-R diagram for Library Management System.

**Solution:** Fig. 2.54 shows E-R model (diagram) for Library Management System.

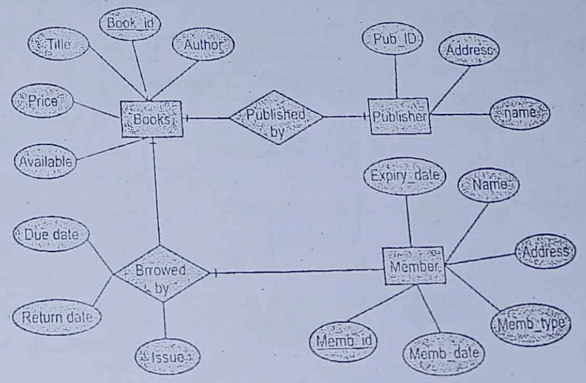



Fig. 2.54

**Case Study 9:** Draw E-R diagram for College Management System.

**Solution:** Fig. 2.55 shows E-R model for College Management System.

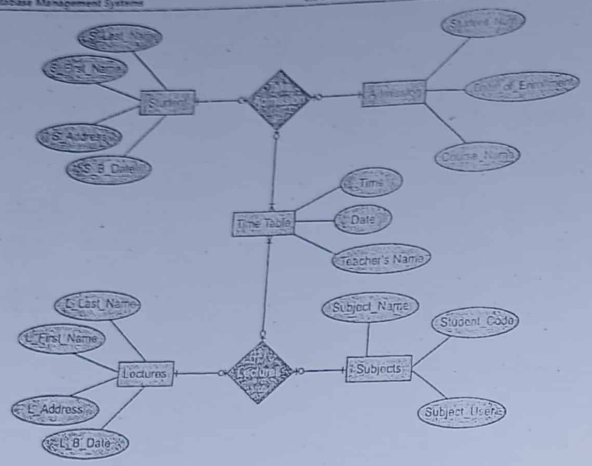



Fig. 2.55

**Case Study 10:** Draw E-R diagram for Railway Reservation System.

**Solution:** Fig. 2.56 shows E-R model (diagram) for Railway Reservation System.

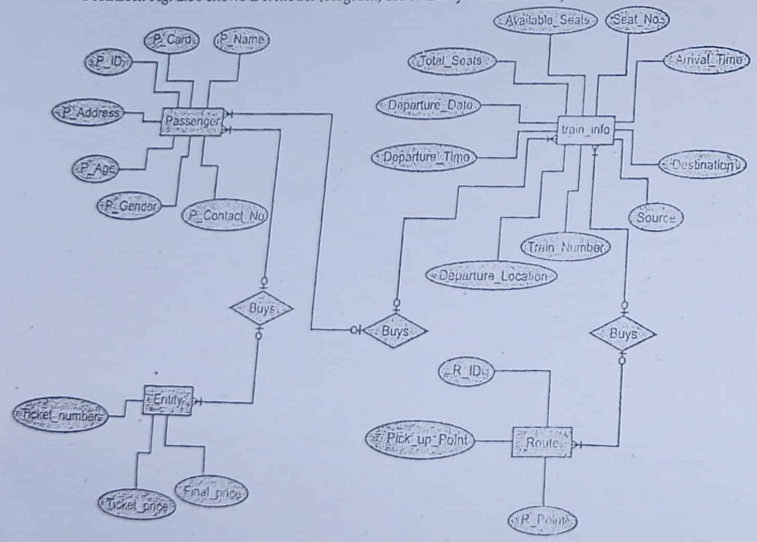



Fig. 2.56



**Case Study 11:** Draw E-R diagram for On-line Movie Ticket Booking System.

**Solution:** Fig. 2.57 shows E-R model for On-line Movie Ticket Booking System.

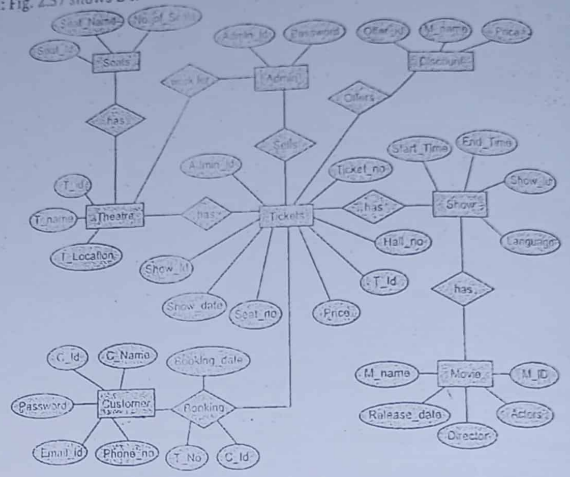



Fig. 2.57

**Case Study 12:** Draw E-R diagram for On-line Book Shopping.

**Solution:** Fig. 2.58 shows E-R model (diagram) for On-line Book Shopping.

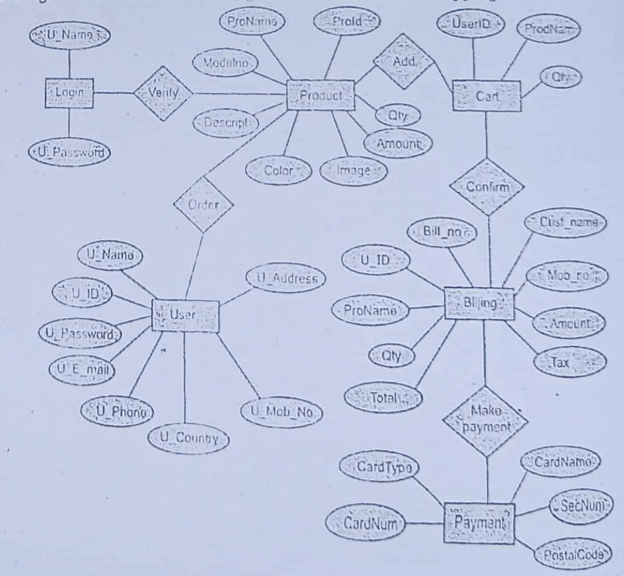



Fig. 2.58

## PRACTICE QUESTIONS

**Q. I: Multiple Choice Questions:**

1. A data model is a collection of conceptual tool for describing data, its _____.
   (a) relationship (b) semantics
   (c) constraints (d) All of these.
2. A way to structure and organize data so it can be used easily by databases called as _____.
   (a) Data modeling (b) Design modeling
   (c) Analysis modeling (d) None of these
3. Which model consists of collection of basic objects called as entities and the relationship among these objects?
   (a) Network model (b) Hierarchical model
   (c) E-R model (d) All of these.
4. Which model was formally introduced by Dr. E. F. Codd in 1970.
   (a) Relational model (b) Hierarchical model
   (c) E-R model (d) All of these.
5. In which model data is organized into a tree-like structure?
   (a) Network model (b) Hierarchical model
   (c) E-R model (d) All of these.
6. Types of attributes includes _____.
   (a) Simple attribute (b) Single valued attribute
   (c) NULL attribute (d) All of these.
7. The set of all entities of the same type is called as _____.
   (a) entity set (b) relationship set
   (c) strong entity set (d) None of these.
8. The association among entities is called as _____.
   (a) relationship (b) entity
   (c) attribute (d) All of these
9. Which is a set of relationships of the same type?
   (a) entity set (b) relationship set
   (c) strong entity set (d) None of these.
10. Which constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple?
    (a) Key (b) Domain
    (c) Referential Integrity (d) Unique
11. Which constraint uniquely identifies each record in a database table?
    (a) Key (b) Domain
    (c) Referential Integrity (d) Unique
12. Which constraint is used to restrict the value of a column between a ranges?
    (a) Key (b) Domain
    (c) Check (d) Unique



13. Which additional concepts contais an E-R model?
    (a) specialization (b) generalization
    (c) aggregation (d) All of these
14. The pictorial/graphical representation of E-R model is _____.
    (a) Entity-Relationship Diagram (ERD) (b) Object diagram
    (c) Relational diagram (d) All of these
15. Which database is a collection of relations (or two-dimensional tables) having distinct names?
    (a) object (b) relational
    (c) network (d) All of these
16. Concepts of data models that are only useful to computer specialists rather than end users of programs are classified as _____.
    (a) triggered data models (b) logical data models
    (c) conceptual data models (d) physical data models
17. The row in the table is called as _____.
    (a) tuple (b) dimension
    (c) cardinality (d) degree
18. The number of rows in the table of DBMS is called as _____.
    (a) tuple (b) relation
    (c) cardinality (d) degree
19. _____ is a set of columns that uniquely identifies every row in a table.
    (a) key (b) candidate key
    (c) foreign key (d) super key
20. A key consisting of two or more columns is called _____.
    (a) composite key (b) candidate key
    (c) key (d) database key
21. A _____ by identifies every record in a table uniquely.
    (a) candidate key (b) key
    (c) primary key (d) all
22. In E-R diagram double ellipse represents _____.
    (a) multivalued attributes (b) weak entity
    (c) derived attributes (d) primary key
23. E-R model is _____ type of model.
    (a) relational (b) object-based
    (c) record based (d) network

**Answers**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. (d) | 2. (a) | 3. (c) | 4. (a) | 5. (b) | 6. (d) | 7. (a) | 8. (a) | 9. (b) | 10. (a) |
| 11. (d) | 12. (c) | 13. (d) | 14. (a) | 15. (b) | 16. (d) | 17. (a) | 18. (c) | 19. (d) | 20. (a) |
| 21. (c) | 22. (a) | 23. (a) | | | | | | | |

**Q.II: Fill in the Blanks:**

1. A _____ is a collection of concepts that can be used to describe the structure of a database.
2. _____ represents the overall logical structure of a database.
3. The _____ represents data in the form of two-dimension tables called as relations.
4. In _____ data is represented by collection of records and the relationship among the data is represented by links.
5. An _____ is a real world object.
6. An entity set that does not have any key attribute of its own is called a _____.
7. _____ means reference and relationship means connection.
8. A database _____ is a condition specified on a database schema that restricts the data to be inserted in an instance of the database.
9. _____ are attributes or sets of attributes that uniquely identify an entity within its entity set.
10. A _____ is a unique set of values permitted for an attribute in a table.
11. _____ constraints tell the participation of entity set in relationship sets.
12. _____ is the process of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities processing these common features.
13. A single row of a table, which contains a single record for that relation is called a _____.

**Answers**

| | | | |
|---|---|---|---|
| 1. data model | 2. E-R model | 3. relational model | 4. network model |
| 5. entity | 6. Weak entity set | 7. Relation | 8. constraint |
| 9. Keys | 10. Domain | 11. Participation | 12. Generalization |
| 13. Tuple | | | |

**Q. III: State True or False:**

1. The process of applying a data model theory to create a data model instance is known as data designing.
2. An E-R data model is a conceptual data model that views the real world as entities and relationships.
3. Attribute refers to an "object" or "thing" in real world.
4. Data in hierarchical model is similar to network model i.e., links or pointers used to show the relationships.
5. An entity has set of properties called attributes which holds value in it.
6. Simple attribute is an attribute that represents a value that is derivable from the value of related attribute or a set of attributes, not essentially in the same entity.
7. An entity set that has a key attribute is called a strong entity set.
8. The number of participating entities in an relationship defines the degree of the relationship.
9. An attribute which identifies data uniquely is known as candidate key.
10. Referential integrity defines that a foreign key must have a matching primary key or it must be null.
11. Domain constraint uniquely identifies each row record in a database.



12. NOT NULL constraint restricts a column from having a null value.
13. Aggregation is the process of identifying subsets of an entity (super-class or super-type) that share some distinguishing characteristic.
14. An ERD is a pictorial representation of the entities and the relationship between them.
15. The data in RDBMS is stored in database objects called tables.
16. In RDBMS table rows represent attributes and table column represents the attributes records.
17. The weak entity in the E-R diagram is represent by double Ellipse.
18. Entity College is example of abstract entity.
19. A foreign key is a set of columns in one table, which is a key in another table.
20. A table in RDBMS is also called a columns.
21. A key is a minimal set of columns.

**Answers**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. (F) | 2. (T) | 3. (T) | 4. (F) | 5. (T) | 6. (F) | 7. (T) | 8. (T) | 9. (T) | 10. (T) |
| 11. (F) | 12. (T) | 13. (F) | 14. (T) | 15. (T) | 16. (F) | 17. (F) | 18. (T) | 19. (T) | 20. (F) |
| 21. (T) | | | | | | | | | |

**Q. IV: Answer the following Questions:**

**(A) Short Answer Questions:**

1. What is meant by database design?
2. What is E-R model?
3. Define data model.
4. What is meant by data modeling?
5. Define tuple and domain.
6. What is EER?
7. Define ERD.
8. What is meant by constraints?
9. What is key?
10. Enlist record based models.
11. Draw E-R symbols for attribute and entity with example.
12. Define relation.
13. What is meant by mapping cardinality?
14. Define relational database.
15. What are the additional features of E-R model?
16. What is RDBMS?
17. Enlist example of RDBMS.
18. What is meant by NULL?
19. What is unique constraint?
20. Define relationship set.
21. Define strong and weak entity sets.

9. (F) 10. (T)
19. (T) 20. (F)



**(B) Long Answer Questions:**

1. Explain database design process diagrammatically.
2. What is meant by conceptual design?
3. With the help of diagram describe components of E-R model.
4. What are advantages of RDBMS over DBMS?
5. Explain following terms with example:
   (i) Integrity Constraints (ii) Referential Integrity
   (iii) Domain Constraints (iv) Unique Constraints.
6. Describe entity and entity sets with example.
7. Explain key constraint with suitable examples.
8. Explain relationships and relationship sets with example.
9. What are different types of attributes?
10. Describe network model with example. Also states its advantages and disadvantages.
11. With the help of diagram describe structure of relational database.
12. Describe relational model with example. Also states its advantages and disadvantages.
13. Describe hierarchical model with example. Also states its advantages and disadvantages.
14. What is a composite attribute? Explain with example.
15. What is unary relationship? Explain with example?
16. What is binary relationship? Explain with example?
17. What do you mean by cardinality of a relationship?
18. Explain the concept of E-R modeling.
19. Describe mapping constraint with example.
20. What are candidate key and primary key?
21. Explain the difference between primary key and candidate key.
22. Explain with suitable examples mapping cardinalities.
23. Write a short note on: keys.
24. What is
25. Explain following terms :
    (i) Aggregation, (ii) Relationship, (iii) Attribute, (iv) Primary key,
    (v) Super key, (vi) Candidate key, (vii) Weak entity set.
26. What should be qualities of attribute set selected as keys ?
27. Write a short note on E-R diagrams.
28. What is generalization? Explain with examples.
29. What is aggregation? Explain with example.
30. Are weak entities necessary? What is the distinction between weak entity set and strong entity set?
31. What are strong and weak entities? Explain with example.
32. Compare network, relational and hierarchical data models.
33. Draw an E-R diagram for following requirements:
    (i) Each school is composed of several departments e.g. accounting dept., computer dept.



    (ii) The smallest number of departments operated by a school is one and the largest number of departments is indeterminate.
    (iii) Each department belongs to only a single school.
34. What are the candidate keys in the following relationship?
    (i) Employee_code, employee_name, license_number, passport_number, addr, age.
    (ii) Part_id, part_name, unit-price, colour, make.
35. What are the primary keys in the following relationship?
    (i) Movie_id, name, actor_name, actress_name, Director_name.
    (ii) Player_name, run, Match_id, ground_name, Date.
36. Draw an E-R diagram for relationship between students who learn many subjects taught by many teachers, we need to maintain information such as the students names, address and divisions and the teachers name and addresses.
37. Compare DBMS and RDBMS.

## UNIVERSITY QUESTIONS AND ANSWERS

### April 2015

1. Define Super Key. (1 M)

Ans. Refer to Page 2.29 Point (iii).

2. What is the referential integrity constraint? (1 M)

Ans. Refer to Section 2.4.3.

3. What is generalization? Explain with an example. (5 M)

Ans. Refer to Section 2.5.2.

4. What are key constraints? (5 M)

Ans. Refer to Section 2.4.2.1.

5. Explain single valued and multivalued attributes with examples. (5 M)

Ans. Refer to Pages 2.10 and 2.11 Points (3) and (4).

6. State and explain different types of relationships that can exist in an entity set in an E-R Model. (5 M)

Ans. Refer to Section 2.3.5.

7. Explain Ternary Relationship with examples. (3 M)

Ans. Refer to Page 2.15 Point (3).

### April 2016

1. Define a entity. (1 M)

Ans. Refer to Section 2.3.1.

2. State the Entity Integrity Constraint. (1 M)

Ans. Refer to Page 2.19 Point (1).

3. What do you mean by domain of an attribute? (1 M)

Ans. Refer to Section 2.7.2.

4. Justify true or false – 'Primary key cannot be null'. (1 M)

Ans. Refer to Page 2.29 Point (i).

5. Explain aggregation with example. (5 M)

**Ans.** Refer to Section 2.5.3.

6. What are mapping cardinalities? Explain any two in detail. (5 M)

**Ans.** Refer to Page 2.17.

7. What is an attribute? State different types of attribute and explain any two in detail. (5 M)

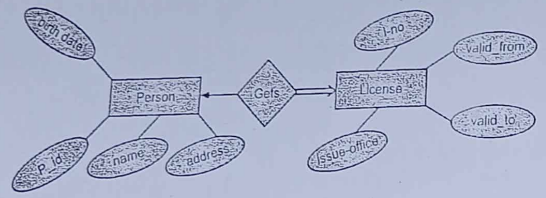
**Ans.** Refer to Section 2.3.2.

8. State rules of conversion of an Entity Relationship diagram into rational tables. (5 M)

**Ans.** Refer to Section 2.7.

9. Explain with example the difference between weak entity set and strong entity set. (3 M)

Consider the following ER diagram:




Convert the above E-R diagram into relational model. (3 M)

**Ans.** Refer to Section 2.7.

## October 2016

1. Define super key. (1 M)

**Ans.** Refer to Page 2.29 Point (iii).

2. What is ternary relationship? (1 M)

**Ans.** Refer to Page 2.15 Point (3).

3. What is referential integrity? (1 M)

**Ans.** Refer to Section 2.4.3.

4. What is attribute? Explain different types of attributes. (5 M)

**Ans.** Refer to Section 2.3.2.

5. Write a short note on hierarchical data model. (5 M)

**Ans.** Refer to Section 2.2.4.

6. What is integrity constraint? What are its different types? (5 M)

**Ans.** Refer to Section 2.4.2.

7. The 'Flyhigh' airline wants to organize their information in DBMS with the following facts :
   (i) Every airplane has a registration number and each airplane is of a specific model.
   (ii) The airport accommodates a number of airplane model that is identified by a model number and has a capacity and weight.
   (iii) All airport employees are part of a union. Their union membership along with address, phone and salary needs to get stored.



   (iv) The employees which are technician conduct different tests of airplanes to ensure it is still airworthy.
   (v) The airport needs to keep track of all such tests and the employees involved and result of the test.
   (a) Design a suitable information system for all the above.
   (b) Draw a suitable E-R diagram for the above system. (5 M)

**Ans.** Refer to Section 2.6.

## April 2017

1. List the record based logical models. (1 M)

**Ans.** Refer to Page 2.3 Point (3).

2. Define candidate key. (1 M)

**Ans.** Refer to Page 2.29 Point (iv).

3. What is referential integrity constraint? Explain in brief. (5 M)

**Ans.** Refer to Page 2.19 Point (2).

4. What is aggregation? Explain with example. (5 M)

**Ans.** Refer to Section 2.5.3.

5. Write a short note on data model. (5 M)

**Ans.** Refer to Section 2.2.

6. Differentiate between specialization and generalization. (3 M)

**Ans.** Refer to Page 2.25.

7. Suyog Hotel has many rooms. Rooms are classified as AC and Non-AC. Hotel provides the discount to customers who visit the hotel more than once. Hotel also has food section and laundry section. This facility gives to customer as per customer's requirement. Every room of hotel has TV and telephone facility, which charge with the total cost of room.
   (i) Design an E-R diagram. (5 M)

**Ans.** Refer to Section 2.6.

## October 2017

1. Explain the term tuple with example. (1 M)

**Ans.** Refer to Section 2.7.2.

2. Define strong entity. (1 M)

**Ans.** Refer to Page 2.12 Point (2).

3. What is aggregation? (1 M)

**Ans.** Refer to Section 2.5.3.

4. Differentiate between specialization and generalization. (5 M)

**Ans.** Refer to Page 2.25.

5. What are integrity constraints? How are they classified? (5 M)

**Ans.** Refer to Section 2.4.2.

6. Differentiate between strong entity and weak entity with suitable example. (3 M)

**Ans.** Refer to Page 2.12.

April 2018

1. Define attributes with example.

Ans. Refer to Section 2.3.2. (1 M)

2. What is referential integrity?

Ans. Refer to Section 2.4.3. (1 M)

3. Define entity set. Explain strong and weak entity.

Ans. Refer to Section 2.3.3. (5 M)

4. Compare primary key and super key.

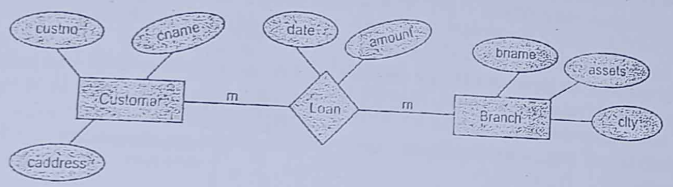
Ans. Refer to Page 2.29 Points (i) and (ii). (5 M)

5. What is a foreign key constraint? Why are such constraints important? What is referential integrity ?

Ans. Refer to Section 2. (5 M)

6. What is existence dependency of entity set ? Explain it with suitable example. (5 M)

Consider the following E-R diagram :




(i) Convert the above E-R diagram into relational model. (3 M)

Ans. Refer to Section 2.6.

7. Savitribai Phule Pune University offers choice based credit system to all P.G. Course. For P.G. science courses students have to complete 100 credits to get their certificate. These courses have semesters two, three ..... Each semester has set of subjects. The subject may be core (compulsory) or elective (optional).

Design an E-R diagram for above scenario, assume, attributes if necessary. (5 M)

Ans. Refer to Section 2.6

CHAPTER 3

# SQL

## 3.0 INTRODUCTION

- SQL is a standard language for accessing and manipulating databases. SQL stands for Structured Query Language. SQL lets you to access and manipulate databases.
- SQL is the standard language for Relational Database System. SQL is a database computer language designed for the retrieval and management of data in a relational database.
- All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
- In this chapter we study SQL with its basic concepts.

## 3.1 INTRODUCTION TO QUERY LANGUAGE

- A query language for relational model is based on relational algebra. It provides a notion for representing queries.
- The commercial database system requires a more user-friendly query language. SQL i.e. Structured Query Language (SQL), it uses a combination of relational algebra and relational calculus constructs.
- SQL contains other capabilities besides querying a database.
- These capabilities includes features for defining the structure of the data, features for modifying data in the databases and features for specifying security constraints.
- There are numerous versions of SQL. The original version was developed at IBM's San Jose Research Laboratory by Dr. E. F. Codd in early 1970. This was based on the relational database model.
- After that IBM's research division and Donald Chamberlin developed a prototype language called sequel i.e. Structured English Query Language.
- For some legal reason, IBM eventually changed the name sequel/2 to SQL (Structured Query Language). In 1986, ANSI and ISO had declared SQL as a standard language. Therefore sometimes, it is referred as Standard Query Language. From 1986, SQL has become universally adopted language.

Features of SQL:

1. SQL is very flexible.
2. SQL based applications are portable i.e., they can be run on any system.
3. SQL is a high level language.
4. SQL is a free-format syntax which gives users the ability to structure SQL statements on any of the screen.
5. SQL can be embedded also in front end application code like of PHP, JAVA and so on.

Elements of SQL:

- SQL contains following elements:
  1. Queries retrieve data based on particular or specific creation.
  2. Statements may control program flow, transactions etc.
  3. Expressions can produce either scalar values or tables which consisting of rows and columns of data.
  4. Clauses are in some cases optional, constituent components of queries and statements.