**1. Explain the three-schema architecture with a neat diagram .Justify the need of mappings between schema levels. (*imp*)**

**External schemas:(client)**

* at the external level
* It describes the various user views.
* It Uses a representational data model.

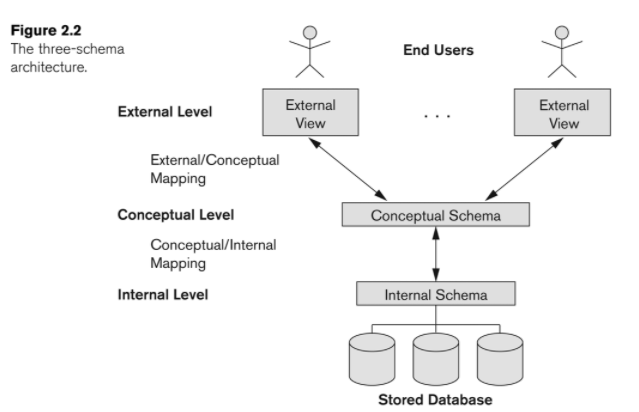
**Conceptual schema:(server)**

* at the conceptual level
* It describes the structure and constraints for the whole database for a community of users.
* It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
* It uses a representational data model.

**Internal schema:(database)**

* at the internal level
* It describes physical storage structures and access paths of the database (e.g indexes).
* It uses a physical data model.

**Goal of three-schema**: to separate user application from physical database.



* Mappings among schema levels are needed to transform requests and data.
* Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
* Data extracted from the internal DBMS level is reformatted to match the user’s external view (e.g. formatting the results of an SQL query for display in a Web page)

**2. What is DBMS. Explain the advantages of Database approach.**

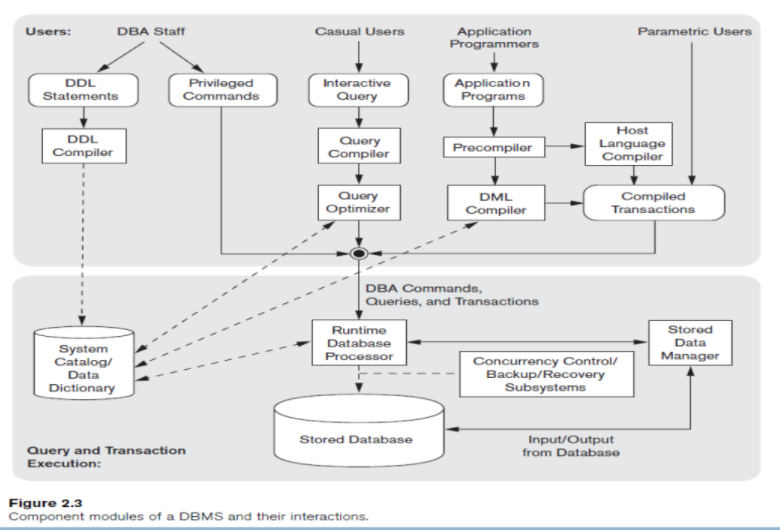
A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

**Advantages**

1. Controlling Redundancy
   1. This redundancy in storing the same data multiple times leads to several problems.
   2. First, entering data on a new student-multiple times: once for each file where student data is recorded. This leads to duplication of effort.
   3. Second, storage space is wasted when the same data is stored repeatedly, and this problem may be serious for large databases.
   4. Third, files that represent the same data may become inconsistent. It is sometimes necessary to use controlled redundancy for improving the performance of queries
2. Restricting Unauthorized Access
   1. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions.
   2. The DBMS should then enforce these restrictions automatically.
   3. only the DBA's staff may be allowed to use certain privileged software, such as the software for creating new accounts.
3. Providing Persistent Storage for Program Objects
   1. Programming languages typically have complex data structures,such as class definitions in c++ or Java.
   2. The values of program variables are discarded or explicitly stored into a format suitable for file storage.
   3. To read this data once more, the programmer must convert from the file format to the program variable structure.
   4. Object-oriented database systems are compatible with programming languages such as c++ and Java, and the DBMS software automatically performs any necessary conversions.
   5. A complex object in c++ can be stored permanently in an OODB. Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another c++ program.
4. Providing Storage Structures for Efficient Query Processing
   1. DBMS must provide specialized data structures to speed up disk search for the desired records.
   2. Auxiliary files called indexes are used for this purpose.
   3. In order to process the database records needed by a particular query, those records must be copied from disk to memory. DBMS has a buffering module that maintains parts of the database in main memory buffers or use operating system for the same.
   4. The query processing and optimization module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures.
5. Providing Backup and Recovery
   1. A DBMS must provide facilities for recovering from hardware or software failures.
   2. The backup and recovery subsystem of the DBMS is responsible for recovery.
6. Providing Multiple User Interfaces
   1. A DBMS should provide a variety of user interfaces.
   2. These include query languages for casual users
   3. programming language interfaces for application programmers
   4. Graphical user interfaces (GUIs) -forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for stand-alone users.
   5. Web GUl interfaces to a database
7. Representing Complex Relationships among Data
   1. A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.
8. Enforcing Integrity Constraints
   1. Most database applications have certain integrity constraints that must hold for the data.
   2. A DBMS should provide capabilities for defining and enforcing these constraints.
   3. The simplest type of integrity constraint involves specifying a data type for each data item.
   4. Other constraints may have to be checked by update programs or at the time of data entry.
9. Permitting Inferencing and Actions Using Rules
   1. Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts.
   2. Such systems are called deductive database systems.
   3. For example, there may be complex rules in the miniworld application for determining when a student is on probation.

( Pick any 6 advantages )

**3. Illustrate the different components of DBMS with a neat diagram.**



The following figure illustrates, in a simplified form, the typical DBMS components. The figure is

divided into two parts. The top part of the figure refers to the various users of the database

environment and their interfaces. The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions.

The **database** and the **DBMS catalog** are usually stored on disk. Access to the disk is controlled primarily by the operating system (OS) or the DBMSs themselves(Performance).

A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

Let us consider the top part of Figure first.

The **DDL compiler** processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta- data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints.

Casual users interact using the interactive query interface. These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a **query compiler** that compiles them into an internal form.

The **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution and makes calls on the runtime processor.

Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a precompiler.

**The precompiler** extracts DML commands from an application program written in a host programming language.

These commands are sent to the **DML compiler** for compilation into object code for database access. The rest of the program is sent to the **host language compiler**. The object codes for the DML commands and the rest of the program are linked, forming a canned

transaction whose executable code includes calls to the **runtime database processor**.

In the lower part of Figure

the runtime database processor executes

(1) the privileged commands,

(2) the executable query plans,

(3) the canned transactions with runtime parameters.

It also works with the stored data manager,which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory. The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory. Some DBMSs have their own buffer management module while others depend on the OS for buffer management.

**3. Define the following terminologies:**

**i) Entity and Attribute**

**ii) Total and partial participation**

**iii) Multivalued and derived attribute**

**iv) Simple and composite attributes**

**v) Schemas and Instances.**

i)

An **Entity** may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course

**Attributes** are the properties which define the entity type. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which define entity type Student.

ii)

**Total Participation** – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by a double line in the ER diagram.

**Partial Participation** – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

iii)

**Multivalued Attribute** –

An attribute consisting of more than one value for a given entity. For example, Phone\_No (can be more than one for a given student). In the ER diagram, the multivalued attribute is represented by a double oval.

**Derived Attribute** –

An attribute which can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In the ER diagram, the derived attribute is represented by a dashed oval.

iv)

**Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

**Composite Attribute** –

An attribute composed of many other simple attributes is called a composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In the ER diagram, the composite attribute is represented by an oval consisting of ovals.

v)

**Schema**

Includes descriptions of the database structure, data types, and the constraints on the database.

**Instance**

Database State:

The actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database. Also called **database instance** (or occurrence or snapshot).

**4. Discuss the various criteria based on which a database management system is classified / Classification of DBMS.**

Criterias to classify a database

1. **Data model** on which the DBMS is based. We can categorize DBMSs based on the data model: relational, object, object-relational, hierarchical, network, and other.
2. **Number of users** supported by the system. Single- user systems support only one user at a time and are mostly used with PCs. Multiuser systems, which include the majority of DBMSs, support concurrent multiple users.
3. **Number of sites** over which the database is distributed. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network.
4. **Cost**.. The giant systems are being sold in modular form with components to handle distribution, replication, parallel processing, mobile capability, and so on, and with a large number of parameters that must be defined for the configuration. Furthermore, they are sold in the form of licenses—site licenses allow unlimited use of the database system with any number of copies running at the customer site. Another type of license limits the number of concurrent users or the number of user seats at a location.
5. **Types of access path** options for storing files. One well-known family of DBMSs is based on inverted file structures.
6. **Purpose**, a DBMS can be general purpose or special purpose. When performance is a primary consideration, a special-purpose DBMS can be designed and built for a specific application

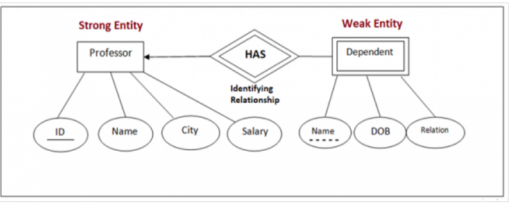
**5. Explain the following concepts with example:**

**i) Identifying relationship ii) Role Names. iii) Cardinality ratio.**

i) **Identifying relationship**

the relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type is called identifying relationship .A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.

Eg:Suppose one intends to keep a record of chapters included in a book. We know that chapters will only exist when a book exists. Thus the relationship between a book and its chapters is an identifying relationship.



ii) **Role Names**

Each entity type that participates in a relationship type plays a particular role in the relationship. The role name signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means.

Eg: In the WORKS\_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.

iii) **Cardinality Ratio**

Cardinality ratio is a concept that describes a binary relationship set (a relationship that connects two entity sets) and its types. It is about the maximum number of entities of one entity set that are associated with the maximum number of entities of the other entity set.

Eg - if a Department can offer many courses and a course can only be offered by at most one department, then the relationship between department and courses is a one-to-many relationship from department to courses.

The following are the types of relationship sets [cardinality ratios];

* [**One-to-one relationship**](http://www.exploredatabase.com/2017/03/explain-one-to-one-relationship-with-example-in-dbms.html)
* [**One-to-many relationship**](http://www.exploredatabase.com/2017/03/explain-one-to-many-relationship-with-example-in-dbms.html)
* **Many-to-one relationship**
* **Many-to-many relationship**

**6. Explain the different types of end users with an example.**

**1. Casual End Users –**

These are the users who occasionally access the database but they require different information each time. They use a sophisticated database query language basically to specify their request and are typically middle or level managers or other occasional browsers. These users learn very few facilities that they may use repeatedly from the multiple facilities provided by DBMS to access it.

**2. Naive or parametric end users –**

These are the users who basically make up a sizeable portion of database end-users. The main job function revolves basically around constantly querying and updating the database for this we basically use a standard type of query known as the canned transaction that has been programmed and tested. These users need to learn very little about the facilities provided by the DBMS they basically have to understand the users’ interfaces of the standard transaction designed and implemented for their use. The following tasks are basically performed by Naive end-users:

The person who is working in the bank will basically tell us the account balance and post-withdrawal and deposits.

Reservation clerks for airlines, railways, hotels and car rental companies basically check availability for a given request and make the reservation.

Clerks who are working at receiving end for shipping companies enter the package identifies via barcodes and descriptive information through buttons to update a central database of received and in-transit packages.

**3. Sophisticated end users –**

These users basically include engineers, scientists, business analytics, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their application to meet their complex requirements. These users try to learn most of the DBMS facilities in order to achieve their complex requirements.

**4. Standalone users –**

These are those users whose job is basically to maintain personal databases by using a ready-made program package that provides easy-to-use menu-based or graphics-based interfaces, An example is the user of a tax package that basically stores a variety of personal financial data for tax purposes. These users become very proficient in using a specific software package.

**7. Differentiate between:**

**i) Derived V/S non-derived attribute**

**ii) Candidate V/S super key.**

**iii) Single v/s multiple valued attribute**

**i) Derived V/S non-derived attribute**

1. **Stored Attribute/ Non Derived Attribute** :

Stored attribute is an attribute which are physically stored in the database.

Assume a table called as student. There are attributes such as student\_id, name, roll\_no, course\_Id. We cannot derive value of these attribute using other attributes. So, these attributes are called as stored attribute.

2. **Derived Attribute** :

A derived attribute is an attribute whose values are calculated from other attributes. In a student table if we have an attribute called as date\_of\_birth and age. We can derive value of age with the help of date\_of\_birth attribute.

**ii)**

|  | **Super Key** | **Candidate Key** |
| --- | --- | --- |
| 1. | Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation. | Candidate Key is a subset of a super key. |
| 2. | All super keys can’t be candidate keys. | But all candidate keys are super keys. |
| 3. | In a relation, number of super keys are more than number of candidate keys. | While in a relation, number of candidate keys are less than number of super keys. |

**iii)Single vs Multivalued Attributes**

**8. What are the responsibilities of a DBA?**

In any organization where many people use the same resources, there is a need for a chief

administrator to oversee and manage these resources. In a database environment, the primary

resource is the database itself, and the secondary resource is the DBMS and related software.

Administering these resources is the responsibility of the database administrator (DBA).

The DBA is responsible for

1. authorizing access to the database,
2. coordinating and monitoring its use, acquiring software and hardware resources as needed.
3. The DBA is accountable for problems such as security breaches and poor system response time.

In large organizations, the DBA is assisted by a staff that carries out these functions.

**9. What do you mean by participation constraint? Explain its types.**

The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the minimum cardinality constraint.

There are two types of participation constraints—total and partial.

1. **Total Participation** - For eg, If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS\_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS\_FOR is called **total participation**, meaning that every entity in the total set of employee entities must be related to a department entity via WORKS\_FOR. Total participation is also called **existence dependency**.
2. **Partial Participation -** For eg, We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial**, meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.

We will refer to the cardinality ratio and participation constraints, taken together, as the **structural constraints** of a relationship type. In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.

**10. What is data independence? Explain the concept of data independence using three schema architecture.**

**Data independence** can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

We can define two types of data independence:

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). In the last case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before. Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.

2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema. Generally, physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding placement, compression, splitting, merging of records, and so on are hidden from the user.

Applications remain unaware of these details. On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement. Whenever we have a multiple-level DBMS, its catalogue must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalogue.

Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed. The

three-schema architecture can make it easier to achieve true data independence, both physical and logical. However, the two levels of mappings create an overhead during compilation or execution of a query or program, leading to inefficiencies in the DBMS. Because of this, few DBMSs have implemented the full three-schema architecture.

**11. What is the referential integrity constraint? Explain with an example, the establishing of referential integrity constraints for recursive relationships.**

**12.Why would you choose a database system instead of simply storing data in operating system files? *(any advantages of dbms)***

**When would it make sense not to use a database system?**

In spite of the advantages of using a DBMS, there are a few situations in which a

DBMS may involve unnecessary overhead costs that would not be incurred in tradi-

tional file processing. The overhead costs of using a DBMS are due to the following:

■ High initial investment in hardware, software, and training

■ The generality that a DBMS provides for defining and processing data

■ Overhead for providing security, concurrency control, recovery, and

integrity functions

Therefore, it may be more desirable to use regular files under the following circum-

stances:

■ Simple, well-defined database applications that are not expected to change at

all

■ Stringent, real-time requirements for some application programs that may

not be met because of DBMS overhead

■ Embedded systems with limited storage capacity, where a general-purpose

DBMS would not fit

■ No multiple-user access to data

Certain industries and applications have elected not to use general-purpose DBMSs. For example, many computer-aided design (CAD) tools used by mechanical and civil engineers have proprietary file and data management software that is geared for the internal manipulations of drawings and 3D objects. Similarly, communication and switching systems designed by companies like AT&T were early manifestations of database software that was made to run very fast with hierarchically organized data for quick access and routing of calls. Similarly, GIS implementations often implement their own data organization schemes for efficiently implementing functions related to processing maps, physical contours, lines, poly-

gons, and so on. General-purpose DBMSs are inadequate for their purpose.

**13.Explain about the following component modules of a DBMS:**

1. **runtime database processor**
2. **Precompiler**
3. **backup and recovery system.**

**14.List and explain the characteristics of database systems**

In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications. The main characteristics of the database approach versus the file-processing approach are the following:

1. **Self-describing nature of a database system**
   1. A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
   2. This definition is stored in the DBMS catalogue, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
   3. The information stored in the catalogue is called meta-data, and it describes the structure of the primary database.
2. **Insulation between programs and data, and data abstraction**
   1. In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalogue separately from the access programs. We call this property **program-data independence**.
   2. In some types of database systems, such as object-oriented, user application programs can operate on the data by invoking some methods, without worrying about how the methods are implemented. This may be termed **program-operation independence**.
   3. The characteristic that allows program-data independence and program-operation independence is called **data abstraction**.
3. **Support of multiple views of the data**
   1. A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.
   2. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.
4. **Sharing of data and multi user transaction processing**
   1. A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include **concurrency control** software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.
   2. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications.
   3. The concept of a transaction has become central to many database applications. A transaction is an executing program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions.
   4. The DBMS must enforce several transaction properties. The **isolation property** ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently. The **atomicity property** ensures that either all the database operations in a transaction are executed or none are.

**15.Define:**

**i) DML.**

**ii) DDL**

**iii) DCL**

**I. DML**

A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data.

The functional capability of DML is organized in manipulation commands like SELECT, UPDATE, INSERT INTO and DELETE FROM

II. **DDL**

A data definition language (DDL) is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc.

Commonly used DDL in SQL querying are CREATE, ALTER, DROP, and TRUNCATE.

III. **DCL**

DCL (Data Control language) includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

**16.Explain three schema architecture with a neat diagram also Compare the logical independence and physical independence.**

Refer ans 1 and ans 10

**17.Describe the following terms:**

**1. Entity Type,**

**2. Degree of a relationship type,**

**3. Recursive relationship,**

**4. Weak Entity type**

**5. Entity set**

**6.Domains of Attributes.**

**18.Explain database system environment with the help of a neat diagram**

**19.How do you define, Structure of a database? Describe different Categories of Data Model.**

A data model is a collection of concepts that can be used to describe the structure of a database that provides the necessary means to achieve data abstraction.

**Categories of Data Models**

Many data models have been proposed, which we can categorize according to the types of concepts they use to describe the database structure. High-level or conceptual data models provide concepts that are close to the way many users perceive data, whereas low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks. Conceptual data models use concepts such as entities, attributes, and relationships. Concepts provided by low-level data models are generally meant for computer specialists, not for end users. Between these two extremes is a class of representational (or implementation) data models which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.

**20.Classify any three Actors on the scene. Briefly explain each with their responsibilities.**

The people whose jobs involve the day-to-day use of a large

database can be called as the actors on the scene.

**i. Database Administrators**

In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources. In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as security breaches and poor system response time. In large organizations, the DBA is assisted by a staff that carries out these functions.

**ii. Database Designers**

Database designers are responsible for identifying the data to be stored in the database and for hoosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements. In many cases, the designers are on the staff of the DBA and may be assigned other staff responsibilities after the database design is completed. Database designers typically interact with each potential group of users and develop views of the database that meet the data and processing requirements of these groups. Each view is then analyzed and integrated with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups.

**iii. End Users**

End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

■ **Casual end users**

■ **Naive or parametric end users**

■ **Sophisticated end users**

■ **Standalone users**

**21. Interfaces of DBS**

**ER DIAGRAM QUESTIONS REFER HERE**

[**https://docs.google.com/document/d/1e9QNPk6UPyTDEt791h1gCqkwUzhk4Cc5/edit**](https://docs.google.com/document/d/1e9QNPk6UPyTDEt791h1gCqkwUzhk4Cc5/edit)