Subject Notes [DBMS CS 502]

INTRODUCTION

Data:-

Data is a plural of datum, which is originally a Latin noun meaning “something given.” Today, data is used in English both as a plural noun meaning “facts or pieces of information” and as a singular mass noun meaning “information”.

Data can be defined as a representation of facts, concepts or instructions in a formalized manner which should be suitable for communication, interpretation, or processing by human or electronic machine.

Data is represented with the help of characters like alphabets (A-Z,a-z), digits (0-9) or special characters(+,-, /,*, <,>, = etc.).

Types of Data: Text, Numbers & Multimedia

Information: -

Information is organized or classified data which has some meaningful values for the receiver.
Information is the processed data on which decisions and actions are based.
For the decision to be meaningful, the processed data must qualify for the following characteristics:

- ✓ **Timely -** Information should be available when required.
- ✓ **Accuracy -** Information should be accurate.
- ✓ **Completeness -** Information should be complete.

File System

Before DBMS invented, information was stored in file processing system. A file processing system is a collection of files and programs that access/modify these files. Typically, new files and programs are added over time (by different programmers) as new information needs to be stored and new ways to access information are needed.
Problems with file processing systems:

File System Verses Database
Most explicit and major disadvantages of file system when compared to database management system are as follows:

1. **Data Redundancy**- The files are formed in the file system as and when required by an enterprise over its progress path. So, in that case the repetition of information about an entity cannot be avoided. E.g. The addresses of customers will be present in the file keeping information about customers holding savings account and also the address of the customers will be present in file maintaining the current account. Even when same customers have a saving account and current account his address will be present at two places.
2. **Data Inconsistency**: Data redundancy leads to bigger problem than just wasting the storage i.e. it may lead to inconsistent data. Same data which has been frequented at several places may not match after it has been updated at some places.
   For example: Suppose the customer requests to change the address for his account in the Bank and the Program is executed to update the saving bank account file only but his current bank account file is not updated. Afterwards the addresses of the same customer present in saving bank account file and current

bankaccount file will not match.
Moreover, there will be no way to find out which address is latest out of these two.

3. **Difficulty in Accessing Data**: For producing ad hoc reports the programs will not already be present and only options present will to write a new program to generate requested report or to work manually. This is going totake impractical time and will be more expensive.

   For example: Suppose all of sudden the administrator gets a request to generate a list of all the customers holding the saving banks account who lives in particular locality of the city. Administrator will not have any program already written to generate that list but say he has a program which can generate a list of all the customers holding the savings account. Then he can either provide the information by going thru the list manually to select the customers living in the particular locality or he can write a new program to generate the new list. Both of these ways will take large time which would generally be impractical.

4. **Data Isolation**: Since the data files are formed at different times and supposedly by different person, the structures of different files generally will not match. The data will be scattered in different files for a particularentity. So, it will be difficult to get appropriate data.

   For example: Suppose the Address in Saving Account file have fields: Add line1, add line2, City, State, Pin while the fields in address of Current account are: House No., Street No., Locality, City, State, and Pin. Administratoris asked to provide the list of customers living in a particular locality. Providing consolidated list of all the customers will require looking in both files. But they both have different way of storing the address.

   Writing a program to generate such a list will be difficult.

5. **Integrity Problems**: All the consistency constraints have to be applied to database through appropriate drafts in the coded programs. This is very difficult when number such constraint is very large.

   For example: An account should not have balance less than Rs. 500. To enforce this constraint appropriate check should be added in the program which add a record and the program which withdraw from an account.Suppose later on this amount limit is increased then all those checks should be updated to avoid inconsistency. These time to time changes in the programs will be great headache for the administrator.

   Security and access control: Database should be protected from unauthorized users. Every user should not be allowed to access every data. Since application programs are added to the system. For example: The Payroll Personnel in a bank should not be allowed to access accounts information of the customers.

6. **Concurrency Problems**: When more than one user is permitted to process the database.If in that environmenttwo or more users try to update a shared data element at about the same time then it may result into inconsistent data. For example, Suppose Balance of an account is Rs. 500. And User A and B try to withdraw Rs. 100 and Rs. 50 respectively at almost the same time using the Update process.

   Update:

   1. Read the balance amount.
   2. Subtract the withdrawn amount from balance.
   3. Write updated Balance value.

   Suppose A performs Step 1 and 2 on the balance amount i.e. it reads 500 and subtracts 100 from it. But at thesame time B withdraws Rs 50 and he performs the Update process and he also reads the balance as 500 subtract 50 and writes back 450. User A will also write his updated Balance amount as 400. They may update the Balance value in any order depending on various reasons concerning to system being used by both of the users. So finally, the balance will be either equal to 400 or 450. Both of these values are wrong for the updatedbalance and so now the balance amount is having inconsistent value forever.

Advantages of database systems

- **Data Independence**: Application programs should not, if possible, be showing to details of data representation and storage, The DBMS provides an abstract view of the data that hides such details.
- **Efficient Data Access:** A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

- **Data Integrity and Security:** If data is always retrieved through the DBMS, the DBMS can implement integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce access controls that govern what data is visible to different classes of users.
- **Data Administration:** When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals, who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.
- **Concurrent Access and Crash Recovery:** A DBMS schedules simultaneous accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced Application Development Time:** Clearly, the DBMS supports important functions that are common to many applications accessing data in the DBMS. This, in conjunction with the high-levelinterface to the data, facilitates quick application development. DBMS applications are also likely to be more robust than similar stand-alone applications because many important tasks are handled by the DBMS(and do not have to be debugged and tested in the application).

What Is Database:

A database consists of an organized collection of interrelated data for one or more uses, typically in digital form.Examples of databases could be: Database for Educational Institute or a Bank, Library, Railway Reservation system etc.

- Consists of two things- a Database and a set of programs.
- Database is a very large, integrated collection of data.
- The set of programs are used to Access and Process the database.

So, DBMS can be defined as the software package designed to store and manage or process the database.

Management of data involves:

- Definition of structures for the storage of information
- Methods to manipulate information
- Safety of the information stored despite system crashes.

Database model's real-world enterprise by entities and relationships.

Entities (e.g., students, courses, class, subject)

CHARACTERISTICS OF DATABASE APPROACH

A number of characteristics distinguish the database method from the much older approach of programming with files. In traditional file processing, each user defines and implements the files needed for a precise software application as part of programming the application.

For example, one user, the grade reporting office, may keep files on students and their grades. Programs to print a student's transcript and to enter new grades are implemented as part of the application.

A second user, the accounting office, may keep track of students' fees and their payments. Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files. This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common up-to-date data. In the database approach, a single repository maintains data that is defined once and then accessedby 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:

- Self-describing nature of a database system

- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing.

Data models

A data model—a collection of concepts that can be used to describe the structure of a database—provides the essential means to achieve the abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data. Most data models also contain a set of basic operations for specifying retrievals and updates on the database.
In addition to the basic operations provided by the data model, it is becoming more common to include concepts in the data model to specify the dynamic aspect or behavior of a database application. This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects.
Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

Types of Database Users:

Users are differentiated by the way they expect to interact with the system:

1. **Application programmers** - interact with system through DML calls.
2. **Sophisticated users** - form requests in a database query language.
3. **Specialized users** - write specialized database applications that do not fit into the traditional data processing framework.
4. **Naive users** - invoke one of the permanent application programs that have been written previously.

Three Level Architecture of DBMS-

An early proposal for a standard terminology and general architecture database a system was produced in 1971 by the DBTG (Data Base Task Group) appointed by the Conference on data Systems and Languages. The DBTG recognized the need for a two-level approach with a system view called the schema and user view calledsubschema. The American National Standard Institute terminology and architecture in 1975.ANSI-SPARC recognized the need for a three-level approach with a system catalog.

There are following three levels or layers of DBMS architecture:

1. External Level
2. Conceptual Level
3. Internal Level

1. External Level: - External Level is described by a schema i.e. it consists of definition of logical records and relationship in the external view. It also contains the method of deriving the objects in the external view from the objects in the conceptual view.

2. Conceptual Level: - Conceptual Level represents the entire database. Conceptual schema describes the records and relationship included in the Conceptual view. It also contains the method of deriving the objects in the conceptual view from the objects in the internal view.

3. Internal Level: - Internal level indicates hoe the data will be stored and described the data structures and access method to be used by the database. It contains the definition of stored record and method of representing the data fields and access aid used.

A mapping between external and conceptual views gives the correspondence among the records and relationship of the conceptual and external view. The external view is the abstraction of conceptual view which in turns is the abstraction of internal view. It describes the contents of the database as perceived by theuser or application program of that view.

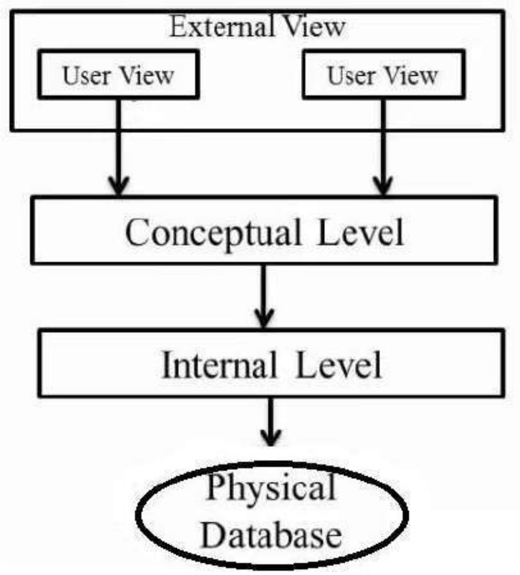


Fig-1.1

Difference between Two Tier and Three Tier Architecture

Sr. No	Two-tier Architecture	Three -tier Architecture
1	Client -Server Architecture	Web -based application
2	Client will hit request directly toserver and client will get response directly from server	Here in between client and server middle ware will be there, if client hits a request it will go to the middle ware and middle ware will send to server and vice versa.
3	2-tier means 1) Design layer 2) Data layer	3-tier means 1) Design layer 2) Business layer or Logic layer 3) Data layer

Three-Tier Architecture:

Three-tier architecture typically comprises a presentation tier, a business or data access tier, and a data tier. Three layers in the three-tier architecture are as follows:

1) Client layer
2) Business layer
3) Data layer

1) Client layer:

It is also called as Presentation *layer* which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user.
Example-Designing registration form which contains text box, label, button etc.

2) Business layer:

In this layer, all business logic likes validation of data, calculations, data insertion etc. This is an interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

3) Data layer:

In this layer, actual database is coming in the picture. Data Access Layer contains methods to join withdatabase and to perform insert, update, delete, get data from database based on our input data.

Advantages

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better Re-use
6. Improve Data Integrity
7. Improved Security – Client is not direct access to database.
8. Easy to maintain and modification is bit easy, won't affect other modules
9. In three tier architecture application performance is good.

Disadvantages

1. Increase Complexity/Effort

RDBMS-

It stands for Relational Database Management System. It organizes data into related rows andcolumns.

Features:

It stores data in tables.
Tables have rows and column. These
tables are created using SQL.

Difference between DBMS and RDBMS

No.	DBMS	RDBMS
1)	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.
4)	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomicity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be **no relation between the tables**.	In RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.

6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8)	DBMS is meant to be for small organization and **deal with small data**. It supports **single user**.	RDBMS is designed to **handle large amount of data**. It supports **multiple users**.

Types of data models are:

- Entity relationship model
- Relational model
- Hierarchical model
- Network model
- Object oriented model
- Object relational model

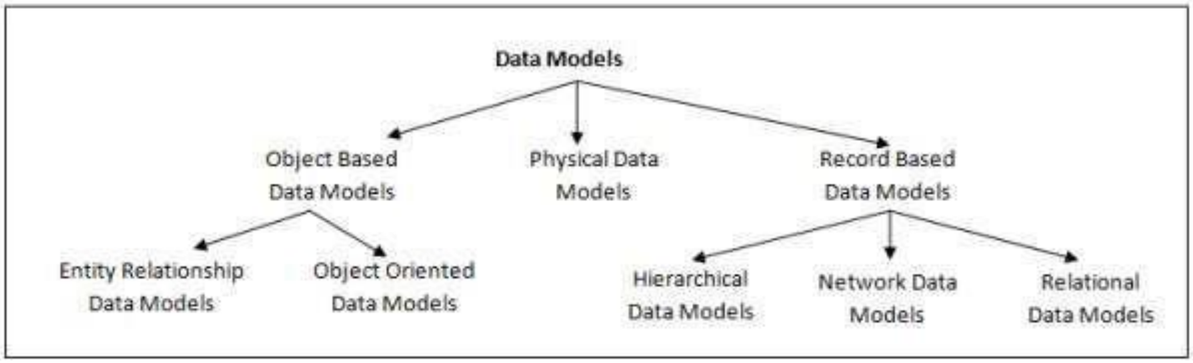


Fig. 1.2

Entity relationship model

1. Entity

An entity can be real world object, either animate or inanimate, that can be simply recognizable. For example, in a school database, students, teachers, classes and courses offered can be considered as entities. Entities are represented by means of rectangles.

2. Relationship

A relationship is an association among several entities. For example, an employee works at a department, a student enrolls in a course. Here, **works at** and **enrolls** are called relationship. Relationships are represented by diamond-shaped box.

3. Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes. Attributes are represented by means ofellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

- **Strong Entity**: Entities having its particular attribute as primary keys are called strong entity. For example, STUDENT has STUDENT_ID as primary key. Hence it is a strong entity.
- **Weak Entity**: Entities which cannot form their own attribute as primary key are known weak entities. These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.
- **Simple Attribute**

These kinds of attributes have values which cannot be divided further. For example, STUDENT_ID attribute which cannot be divided further. Passport Number is unique value and it cannot be divided.

- **Composite Attribute**

This kind of attribute can be divided further to more than one simple attribute. For example, address of a person. Here address can be further divided as Door#, street, city, state and pin which are simple attributes.

- **Derived Attribute**

Derived attributes are the one whose value can be obtained from other attributes of entities in thedatabase. For example, Age of a person can be obtained from date of birth and current date. Average salary,annual salary, total marks of a student etc. are few examples of derived attribute.

- **Stored Attribute**

The attribute which gives the value to get the derived attribute are called Stored Attribute. In example above, age is derived using Date of Birth. Hence Date of Birth is a stored attribute.

- **Single Valued Attribute**

These attributes will have only one value. For example, EMPLOYEE_ID, passport#, driving license#, SSN etc have only single value for a person.

- **Multi-Valued Attribute**

These attributes can have more than one value at any point of time. Manager can have more than one employee working for him, a person can have more than one email address, and more than one house etc is the examples.

- **Simple Single Valued Attribute**

This is the combination of above four types of attributes. An attribute can have single value at any point of time, which cannot be divided further. For example, EMPLOYEE_ID – it is single value as well as it cannot be divided further.

- **Simple Multi-Valued Attribute**

Phone number of a person, which is simple as well as he can have multiple phone numbers is an example of this attribute.

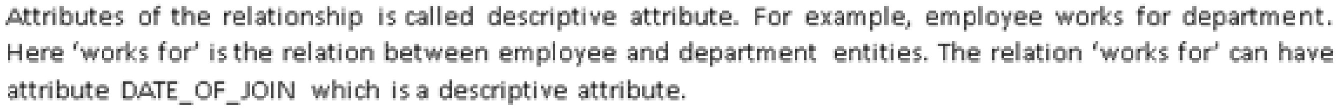
- **Composite Single Valued Attribute**

Date of Birth can be a composite single valued attribute. Any person can have only one DOB and it can be further divided into date, month and year attributes.

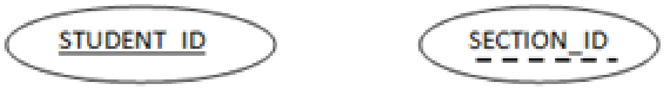
- **Composite Multi-Valued Attribute**

Shop address which is located two different locations can be considered as example of this attribute.

- **Descriptive Attribute**

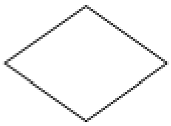


Attributes of the relationship is called descriptive attribute. For example, employee works for department. Here 'works for' is the relation between employee and department entities. The relation 'works for' can have attribute DATE_OF_JOIN which is a descriptive attribute.

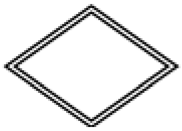


Relationship:

A diamond shape is used to show the relationship between the entities. A mapping with weak entity is shown using double diamond. Relationship name will be written inside them.



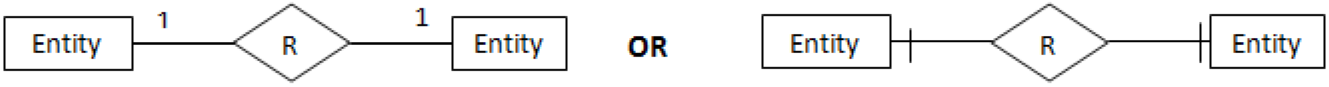
Normal Relations



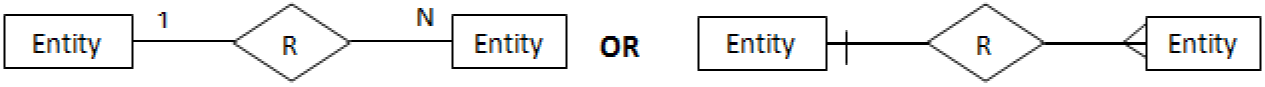
Identifying or weak Relations

Cardinality of Relationship: Different developers use different notation to represent the cardinality of the relationship. Not only for cardinality, but for other objects in ER diagram will have slightly different notations. But main difference is noticed in the cardinality. For not to get confused with many, let us see two types of notations for each.

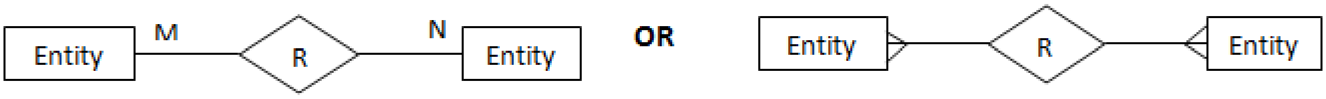
One-to-one relation:- A one-to-one relationship is represented by adding '1' near the entities on the line joining the relation. In another type of notation one dash is added to the relationship line at both ends.



One-to-Many relation: A one-to-many relationship is represented by adding '1' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have dash at LHS of relation and three arrow kinds of lines at the RHS of relation as shown below.



Many-to-Many relation: A one-to-many relationship is represented by adding 'M' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have three arrow kinds of lines at both sides of relation as shown below.



Participation Constraints: Total participation constraints are shown by double lines and partial participations are shown as single line.

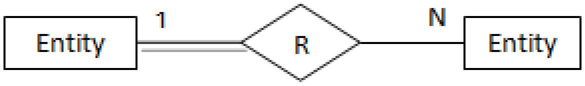
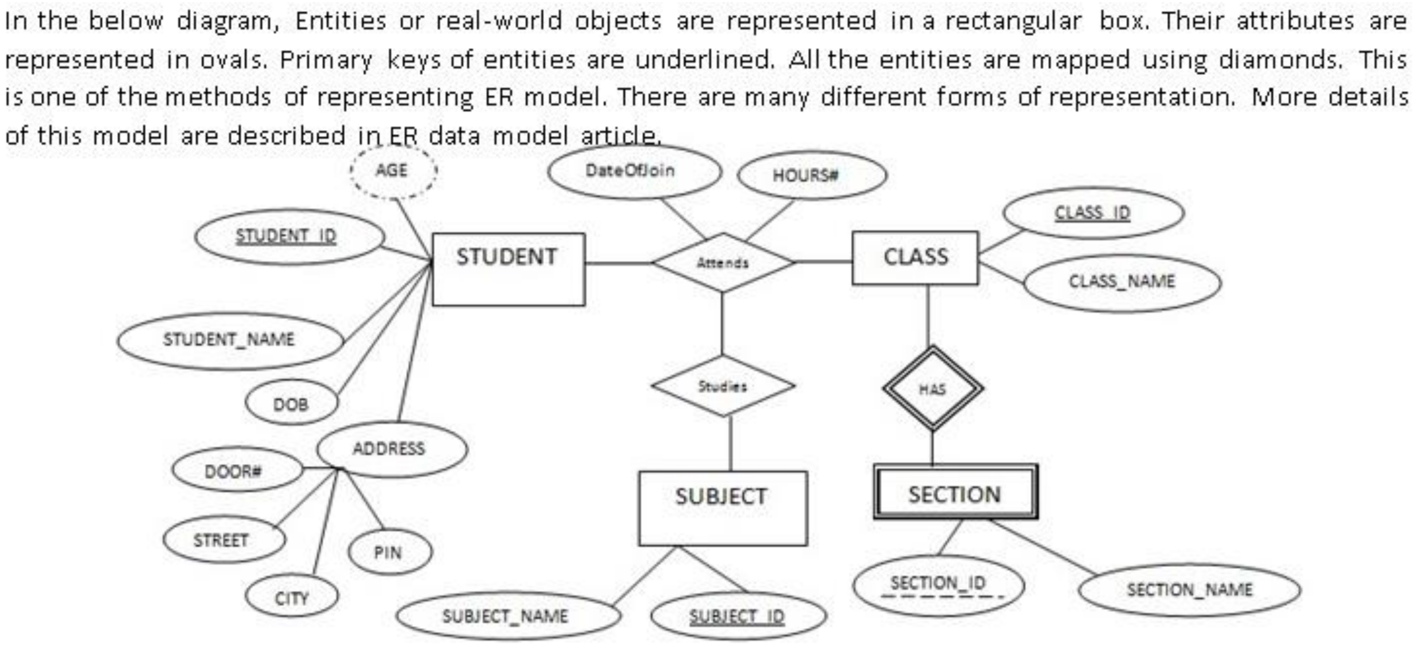


Fig 1.3

Data Models:

1. **ER Data Model:**



In the below diagram, Entities or real-world objects are represented in a rectangular box. Their attributes are represented in ovals. Primary keys of entities are underlined. All the entities are mapped using diamonds. This is one of the methods of representing ER model. There are many different forms of representation. More details of this model are described in ER data model article.

Basically, ER model is a graphical representation of real world objects with their attributes and relationship. It makes the system easily understandable. This model is considered as a top down approach of designing a requirement.

Advantages:

- It makes the requirement simple and easily reasonable by representing simple diagrams.
- One can covert ER diagrams into record based data model easily.
- Easy to understand ER diagrams

Disadvantages:

- No standard notations are available for ER diagram. There is great flexibility in the notation. It's all depends upon the designer, how he draws it.
- It is meant for high level designs. We cannot simplify for low level design like coding.

2. Object Oriented Data Model

This data model is another method of representing real world objects. It considers each object in the world asobjects and isolates it from each other. It groups its related functionalities together and allows inheriting i ts functionality to other related sub-groups.

Let us consider an Employee database to understand this model better. In this, we have different types of employees – Engineer, Accountant, Manager, Clark. But all these employees belong to Person group. Person can have different attributes like name, address, age and phone. What do we do if we want to get a person's address and phone number? We write two separate procedure sp_getAddress and sp_getPhone.

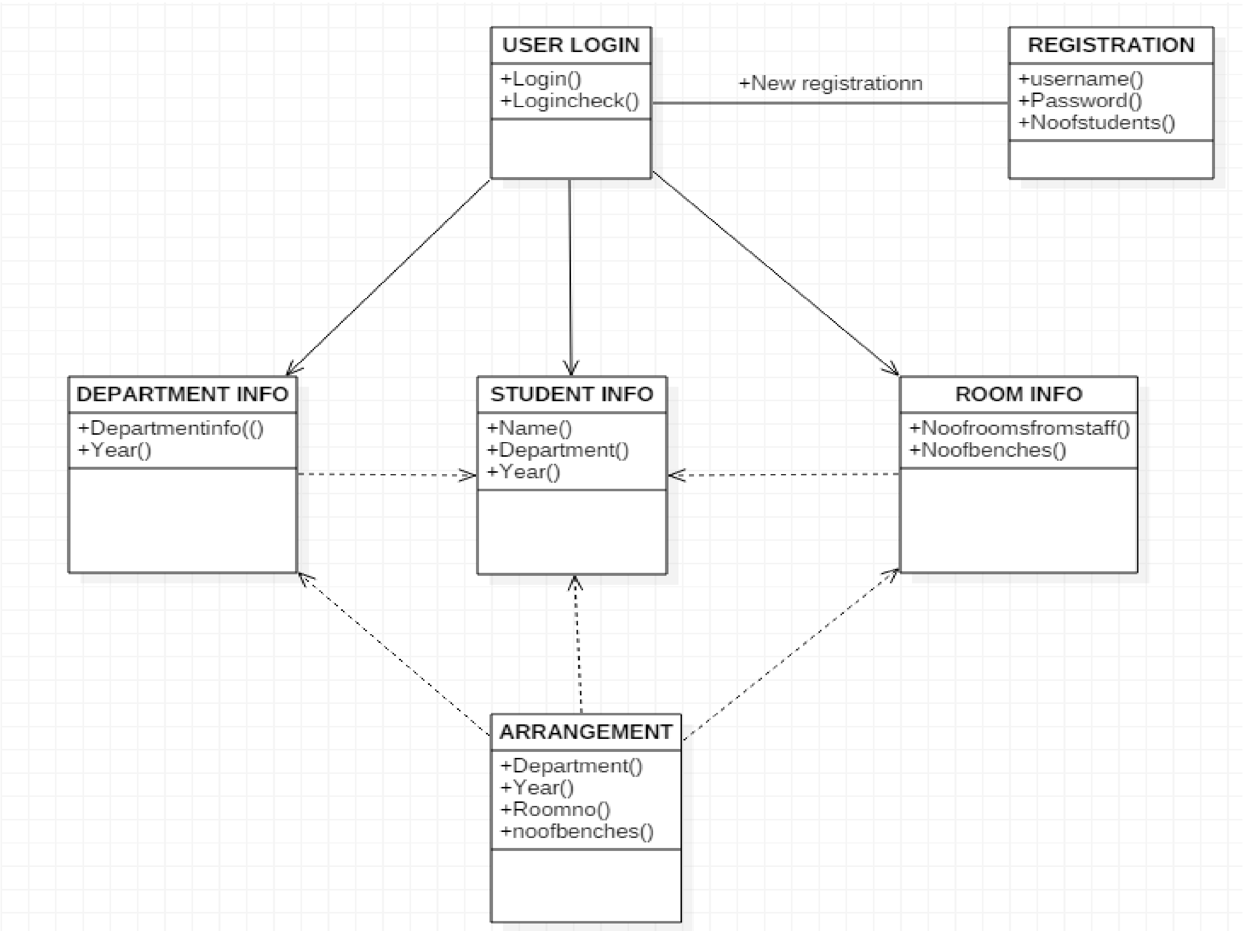


Fig 1.4

Advantages:

- ✓ We can re-use the attributes and functionalities. It reduces the cost of maintaining the same data at multipletimes. Also, these information's are encapsulated and, there is no fear being changed by other objects. If we need any new feature we can easily add new class inherited from parent class and adds new features. Hence itreduces the overhead and maintenance costs.
- ✓ Because of the above feature, it becomes more flexible in the case of any changes.
- ✓ Codes are re-used because of inheritance.
- ✓ Since each class binds its attributes and its functionality, it is same as the real-world object. We can see eachobject as a real entity. Hence it is more understandable.

Disadvantages:

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by theusers.
- It is a method for solving the requirement. It is not a technology. Hence it fails to put it in the databasemanagement systems.

3.Hierarchical Data Model:

Imagine we have to create a database for a company. What are the entities involved in it? Company, its department, its supplier, its employees, different projects of the company etc are the different entities we need to take care of. If we observe each of the entity they have parent –child relationship. We can design them like we do ancestral hierarchy. In our case, Company is the parent and rests of them are its children. Department has employees and project as its children and so on. This type of data modeling is called hierarchical data model.

- In this data model, the entities are represented in a hierarchical fashion. Here we identify a parent entity, andits child entity. Again, we drill down to identify next level of child entity and so on. This model can be imagined as folders inside a folder!

In our example above, it is diagrammatically represented as below:

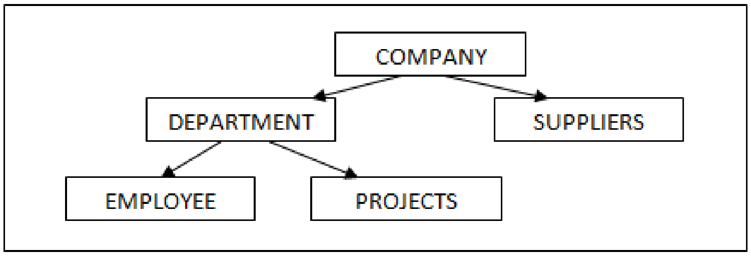


Fig 1.5

It can also be imagined as root like structure. This model will have only one main root. It then branches into sub-roots, each of which will branch again. This type of relationship is best defined for 1 :N type of relationships. E.g.; One company has multiple departments (1:N), one company has multiple suppliers (1:N),one department has multiple employees (1:N), each department has multiple projects(1:N) .

If we have M:N relationships, then we have to duplicate the entities and show it in the diagram.
For example, if a projectin the company involves multiple departments, then our hierarchical representation changes as below:

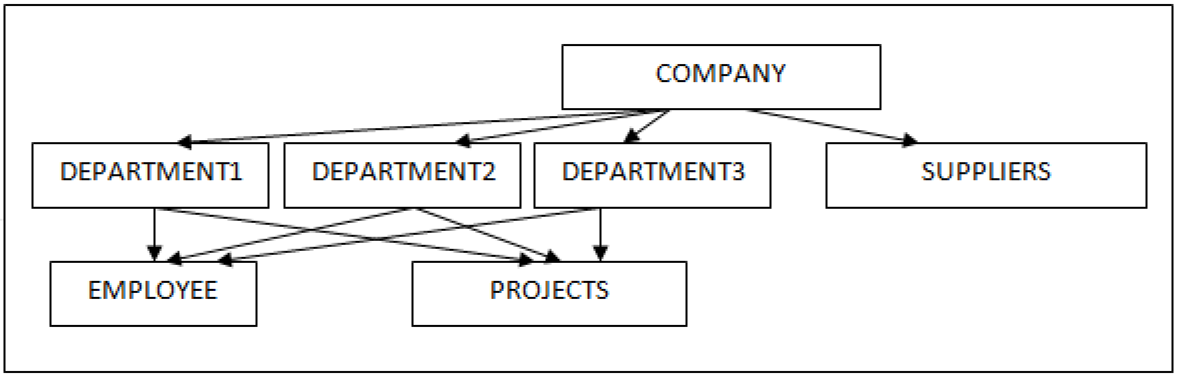


Fig 1.6

Advantages

It helps to address the issues of flat file data storage. In flat files, data will be scattered and there will not be ina proper structure. This model groups the related data into tables and defines the relationship between the tables, which is not addressed in flat files.

Disadvantages

- Redundancy: - When data is stored in a flat file, there is chance of repetition of same data multiple times and any changes required for the data will need to change in all the places in the flat file. Missing to update atany one place will cause incorrect data. This kind redundancy is solved by hierarchical model to some extent. Since records are grouped under related table, it solves the flat file redundancy issue. But look at the many to many relationship examples given above. In such case, we have to store same project information for more than one department. This is duplication of data and hence a redundancy. So, this model does not reduce the redundancy issue to a significant level.
- As we have seen above, it fails to handle many to many relationships competently. Results in redundant andhaving confusion. It can handle only parent-child kind of relationship.

- If we need to fetch any data in this model, we have to start from the root of the model and traverse through its child till we get the result. In order to perform the traversing, either we should know well in advance the layout of model or we should be very good programmer. Hence fetching through this model becomes bit difficult.
- Imagine company has got some new project details, but it did not assign it to any department yet. In this case, we cannot store project information in the PROJECT table, till company assigns it to some department. That means, in order to enter any child information, its parent information should be already known / entered.

4. Network Data Models

This is the improved version of hierarchical data model. It is designed to address the drawbacks of the hierarchical model. It helps to address M: N relationship. This data model is also represented as hierarchical, but this model will not have single parent relationship. Any child in the tree can have multiple parents here.

Revisit our company example: A company has different projects and departments in the company own those projects. Even suppliers of the company give input for the project. Here Project has multiple parents and each department and supplier have multiple projects. This is represented as shown below. Basically, it forms a network like structure between the entities, hence the name.

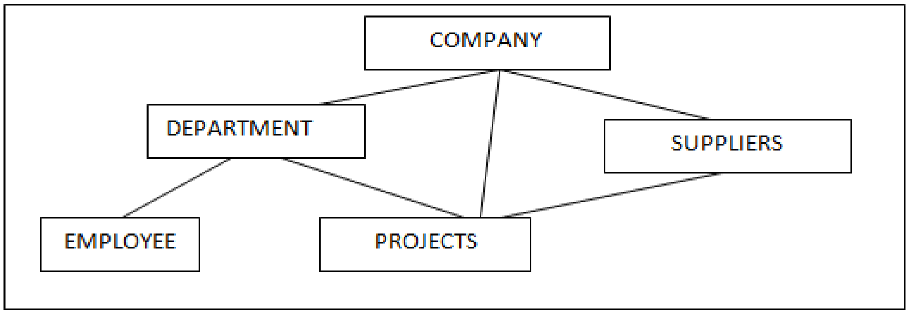


Fig 1.7

One more example:

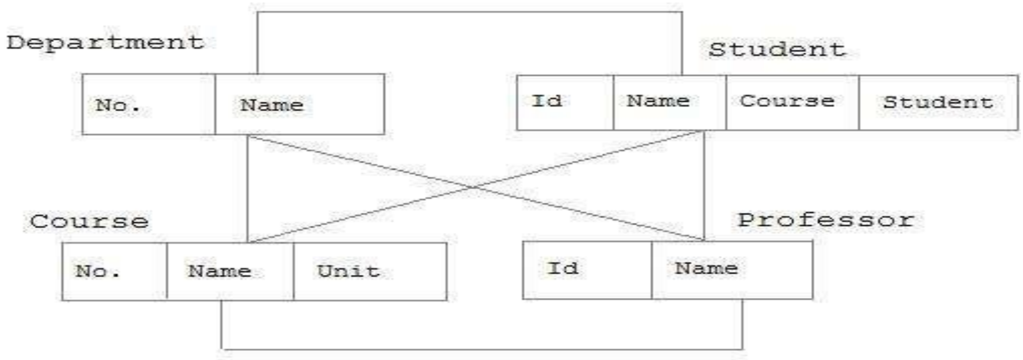


Fig 1.8

Advantages

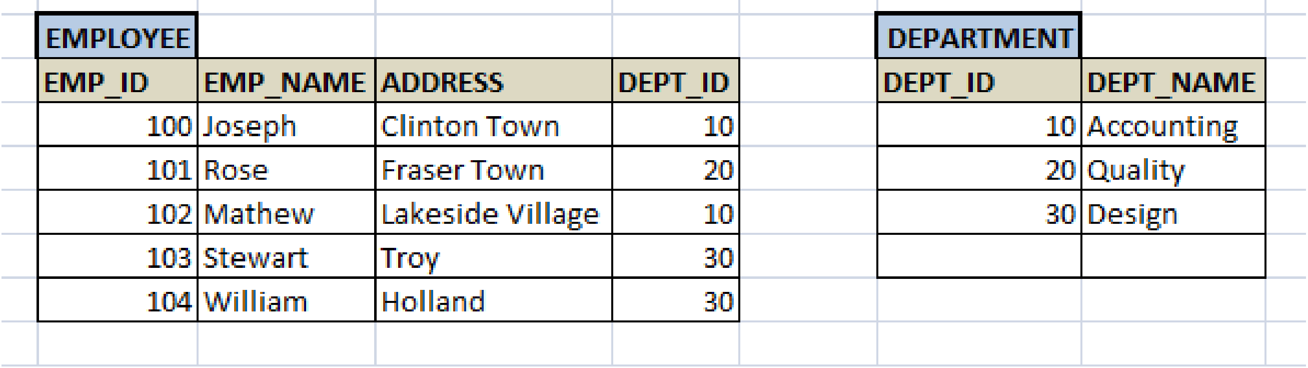
- ✓ Accessing the records in the tables is easy since it addresses many to many relationships. Because of this kind of relationship, any records can be easily pulled by using any tables. For example, if we want to know thedepartment of project X and if we know SUPPLIER table, we can pull this information. i.e.; SUPPLIER has the information about project X which includes the departments involved in the projects too. Hence makes the accessibility to any data easier, and even any complex data can be retrieved easily and quickly.
- ✓ Because of the same feature as per first point, one can easily navigate among the tables and get any data.
- ✓ It is designed based on database standards – ANSI/SP ARC.

Disadvantages

- If there is any requirement for the changes to the entities, it requires entire changes to the database. There is no independence between any objects. Hence any changes to the any of the object will need changes to thewhole model. Hence difficult to manage.
- It would be little difficult to design the relationship between the entities, since all the entities are related in some way. It requires thorough practice and knowledge about the designing.

5. Relational Data Models

This model is designed to overcome the drawbacks of hierarchical and network models. It is designed completely different from those two models. Those models define how they are structured in the database physically and how they are inter-related. But in the relational model, we are least concerned about how they are structured. It purely based on how **the records in each table are related**. It purely isolates physical structure from the logical structure. Logical structure is defining records are grouped and distributed.
Let us try to understand it by an example. Let us consider department and employee from our previous examples above. In this model, we look at employee with its data. When we say an employee what all comes into our mind? His employee id, name, address, age, salary, department that he is working etc. are attributes of employee. That means these details about the employee forms columns in employee table and value set of each employee for these attribute forms a row/record for an employee. Similarly, department has its id, name. Now, in the employee table, we have column which uniquely identifies each employee – that is employee Id column. This column has unique value and we are able to differentiate each employee from each other by using this column. Such column is called as primary key of the table. Similarly, department table has DEPT_ID as primary key. In the employee table, instead of storing whole information about his department, we have DEPT_ID from department table stored. i.e.; by using the data from the department table, we have established the relation between employee and department tables.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Fig 1.9

A relational data model revolves around 5 important rules:

1. Order of rows / records in the table is not important. For example, displaying the records for Joseph is independent of displaying the records for Rose or Mathew in Employee table. It does not change the meaningor level of them. Each record in the table is independent of other. Similarly, order of columns in the table is not important. That means, the value in each column for a record is independent of other. For example, representing DEPT_ID at the end or at the beginning in the employee table does not have any affect.

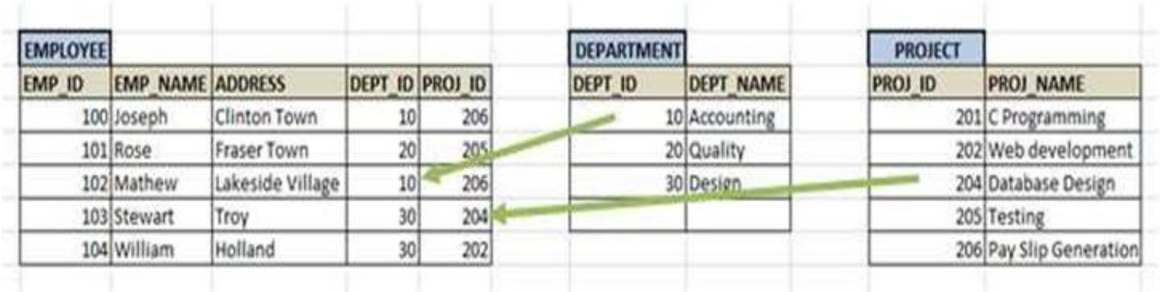
2. Each record in the table is unique. That is there is no duplicate record exists in the table. This is achieved bythe use of primary key or unique constraint.

3. Each column/attribute will have single value in a row. For example, in Department table, DEPT_NAME column cannot have 'Accounting' and 'Quality' together in a single cell. Both has to be in two different rows asshown above.

4. All attributes should be from same domain. That means each column should have meaningful value. For example, Age column cannot have dates in it. It should contain only valid numbers to represent individual's age. Similarly, name columns should have valid names, Date columns should have proper dates.

5. Table names in the database should be unique. In the database, same schema cannot contain two or more tables with same name. But two tables with different names can have same column names. But same columnname is not allowed in the same table.

Examine below table structure for Employee, Department and Project and see if it satisfies relational data model rules.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID	PROJ_ID
100	Joseph	Clinton Town	10	206
101	Rose	Fraser Town	20	205
102	Mathew	Lakeside Village	10	206
103	Stewart	Troy	30	204
104	William	Holland	30	202

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

PROJECT

PROJ_ID	PROJ_NAME
201	C Programming
202	Web development
204	Database Design
205	Testing
206	Pay Slip Generation

Fig 1.10

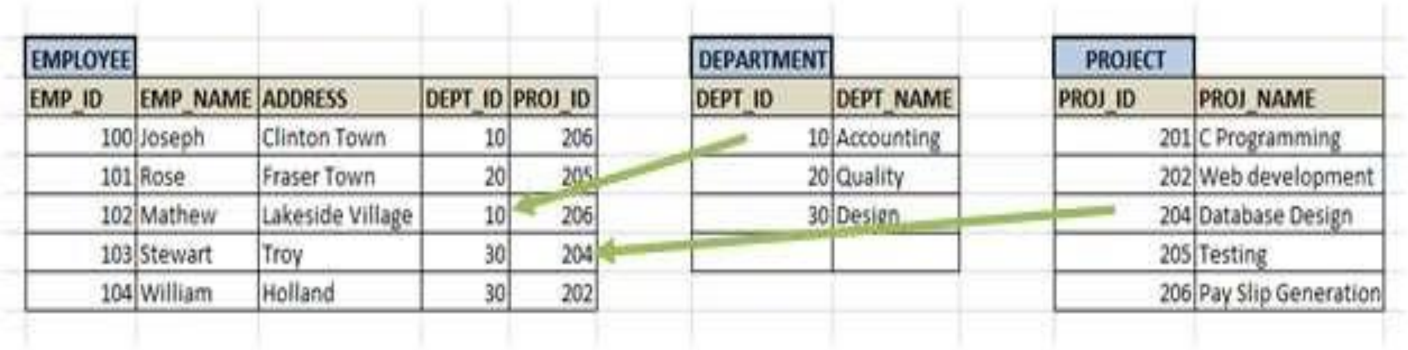
Advantages:

- ✓ Structural independence: - Any changes to the database structure, does not the way we are accessing the data. For example, Age is added to Employee table. But it does not change the relationship between the othertables nor changes the existing data. Hence it provides the total independence from its structure.
- ✓ Simplicity: - This model is designed based on the logical data. It does not consider how data are stored physically in the memory. Hence when the designer designs the database, he concentrates on how he sees the data. This reduces the burden on the designer.
- ✓ Because of simplicity and data independence, this kind of data model is easy to maintain and access.
- ✓ This model supports structured query language – SQL. Hence it helps the user to retrieve and modify the data in the database. By the use of SQL, user can get any specific information from the database.

Disadvantages

Compared to the advantages above, the disadvantages of this model can be ignored.

- High hardware cost: - In order to separate the physical data information from the logical data, more powerful system hardware's – memory is required. This makes the cost of database high.
- Sometimes, design will be designed till the minute level, which will lead to complexity in the database.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID	PROJ_ID
100	Joseph	Clinton Town	10	206
101	Rose	Fraser Town	20	205
102	Mathew	Lakeside Village	10	206
103	Stewart	Troy	30	204
104	William	Holland	30	202

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

PROJECT

PROJ_ID	PROJ_NAME
201	C Programming
202	Web development
204	Database Design
205	Testing
206	Pay Slip Generation

Fig 1.11

Comparison of Record Based Model:

Hierarchical Data Model	Network Data Models	Relational Data Models
Supports One-Many Relationship	Supports both one to many and Many to Many relationship	Supports both one to many and Many to Many relationship
Because of single parent-child relationship, difficult to navigate through the child	It establishes the relationship between most of the objects, hence easy to access compared to hierarchical model	It provides SQL, which makes the access to the data simpler and quicker.
Flexibility among the different object is restricted to the child.	Because of the mapping among the sub level tables, flexibility is more	Primary and foreign key constraint makes the flexibility much simpler than other models.
Based on the physical storage Details	Based on the physical storage details	Based on the logical data view

Relationship

A relationship defines how two or more entities are inter-related. For example, STUDENT and CLASS entities are related as 'Student X **studies** in a Class Y'. Here 'Studies' defines the relationship between Student and Class. Similarly, Teacher and Subject are related as 'Teacher A **teaches** Subject B'. Here 'teaches' forms the relationship between both Teacher and Subject.

Degrees of Relationship

In a relationship two or more number of entities can participate. The number of entities who are part of a particular relationship is called degrees of relationship. If only two entities participate in the mapping, then degree of relation is 2 or binary. If three entities are involved, then degree of relation is 3 or ternary. If more than 3 entities are involved then the degree of relation is called n-degree or n-nary.

Cardinality of Relationship

How many number of instances of one entity is mapped to how many number of instances of another entity is known as cardinality of a relationship. In a 'studies' relationship above, what we observe is only one Student X is studying in on Class Y. i.e.; single instance of entity student mapped to a single instance of entity Class. This means the cardinality between Student and Class is 1:1.

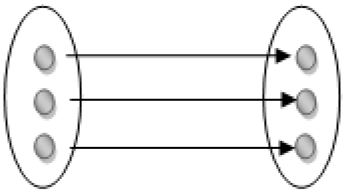
DBA Responsibilities

- Maintaining all databases required for development, training, testing, and production usage
- Installation and configuration of DBMS server software and associated products.
- Upgrading and patching/hot-fixing of DBMS server software and related products.
- Migrate database to another server.
- Evaluate DBMS features and DBMS related products.
- Establish and maintain sound backup and recovery policies and procedures.
- Implement and maintain database security (create and maintain logins, users and roles, assign privileges).
- Performance tuning and health monitoring on DBMS, OS and application.
- Plan growth and changes (capacity planning).
- Do general technical troubleshooting and give consultation to development teams.
- Troubleshooting on DBMS and Operating System performance issue
- Documentation of any implementation and changes (database changes, reference data changes and application UI changes etc.)

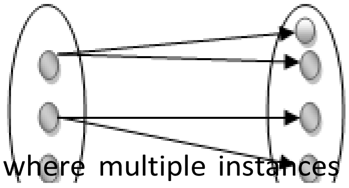
Based on the cardinality, there are 3 types of relationship.

- **One-to-One (1:1)**: As we saw in above example, one instance of the entity is mapped to only one instance of another entity.

Consider, HOD of the Department. There is only one HOD in one department. That is there is 1:1 relationship between the entity HOD and Department.

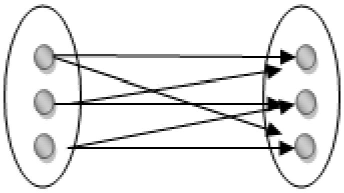


- **One-to-Many (1: M)**: As we can guess now, one to many relationships has one instance of entity related to multiple instances of another entity. One manager manages multiple employees in his department. Here Manager and Employee are entities, and the relationship is one to many. Similarly, one teacher teaches multiple classes is also a 1: M relationship.



- **Many-to-Many (M: N)**: This is a relationship where multiple instances of entities are related to multiple instances of another entity. A relationship between TEACHER and STUDENT is many to many. How? Multiple Teachers teach multiple numbers of Students.

Similarly, above example of 1:1 can be M:N !! Surprised?? Yes, it can be M:N relationship, provided, how we relate these two entities. Multiple Students enroll for multiple classes/courses makes this relationship M:N. The relationship 'studies' and 'enroll' made the difference here. That means, it all depends on the requirement and how we are relating the entities.



DBA Functions/Responsibilities

DBA Responsibilities

- Maintaining all databases required for development, training, testing, and production usage
- Installation and configuration of DBMS server software and associated products.
- Upgrading and patching/hot-fixing of DBMS server software and related products.
- Migrate database to another server.
- Evaluate DBMS features and DBMS related products.
- Establish and maintain sound backup and recovery policies and procedures.
- Implement and maintain database security (create and maintain logins, users and roles, assign privileges).
- Performance tuning and health monitoring on DBMS, OS and application.
- Plan growth and changes (capacity planning).
- Do general technical troubleshooting and give consultation to development teams.
- Troubleshooting on DBMS and Operating System performance issue
- Documentation of any implementation and changes (database changes, reference data changes and application UI changes etc.)

Types of DBA

1. **Administrative DBA –** Work on maintaining the server and keeping it running. Concerned with installation, backups, security, patches, replication, OS configuration and tuning, storage management etc. Things that concern the actual server software.
2. **Development DBA** - works on building queries, stored procedures, etc. that meet business needs. This is the equivalent of the programmer. You primarily write T-SQL or PL-SQL.
3. **Database Architect** – Design schemas. Build tables, FKs, PKs, etc. Work to build a structure that meets the business needs in general. The design is then used by developers and development DBAs to implement the actual application.
4. **Data Warehouse DBA** - responsible for merging data from multiple sources into a data warehouse. May have to design warehouse, but cleans, standardizes, and scrubs data before loading. In SQL Server, this DBA would use SSIS heavily.
5. **OLAP DBA** – Builds multi-dimensional cubes for decision support or OLAP systems. The primary language in SQL Server is MDX, not SQL here
6. **Application DBA-** Application DBAs straddle the fence between the DBMS and the application software and are responsible for ensuring that the application is fully optimized for the database and vice versa. They usually manage all the application components that interact with the database and carry out activities such as application installation and patching, application upgrades, database cloning, building and running datacleanup routines, data load process management, etc.

Generalization

- It is a bottom-up approach in which two lower level entities combine to form higher entity. In generalization, the higher-level entity can also association with other lower level entity to make further higher-level entity.
- Generalization proceeds from the recognition that a number of entity sets share some common features. On the basis of the commonalities, generalization synthesizes these entity sets into a single, higher-level entity set.
- Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences in the schema.

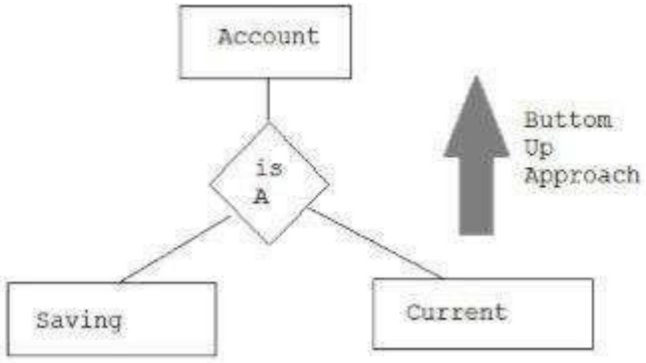


Fig 1.13

Specialization

- It is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into lower level entity.
- Example: The specialization of student allows us to distinguish among students according to whether they are Ex-Student or Current Student.
- Specialization can be repeatedly applied to refine a design schema.

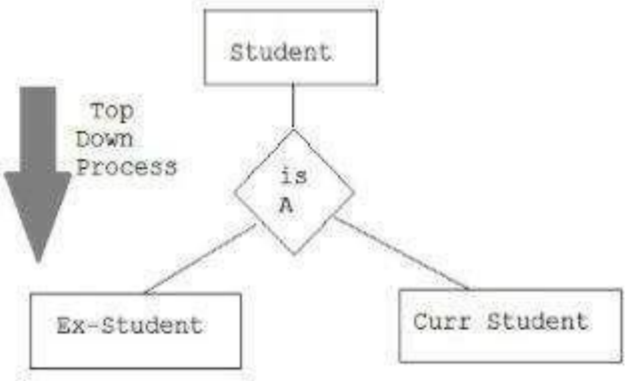


Fig 1.14

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships. To illustrate the need for such a construct, quaternary relationships are used which lead to redundancy in data storage.
- The best way to mode such situations is to use aggregation.
- Aggregation is an abstraction through which relationships are treated as higher-level entities.
- Below is the example of aggregation relation between offer (which is binary relation between center

and course) and visitor.

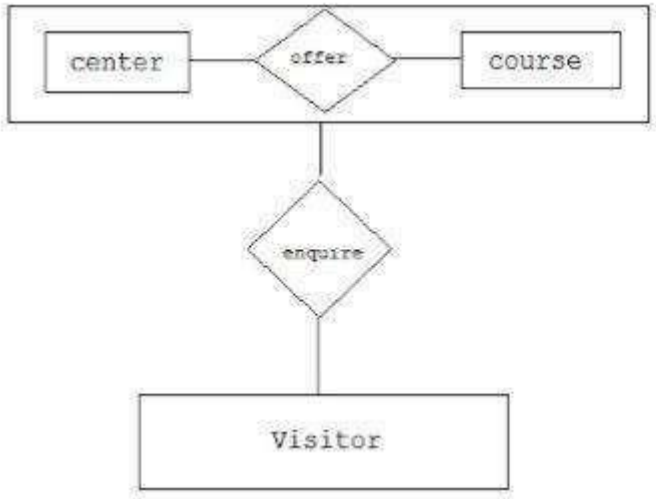


Fig 1.15

Data Independence

1. **Logical data independence:**
    - **It** 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 (byadding 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**:
    - **It** 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.

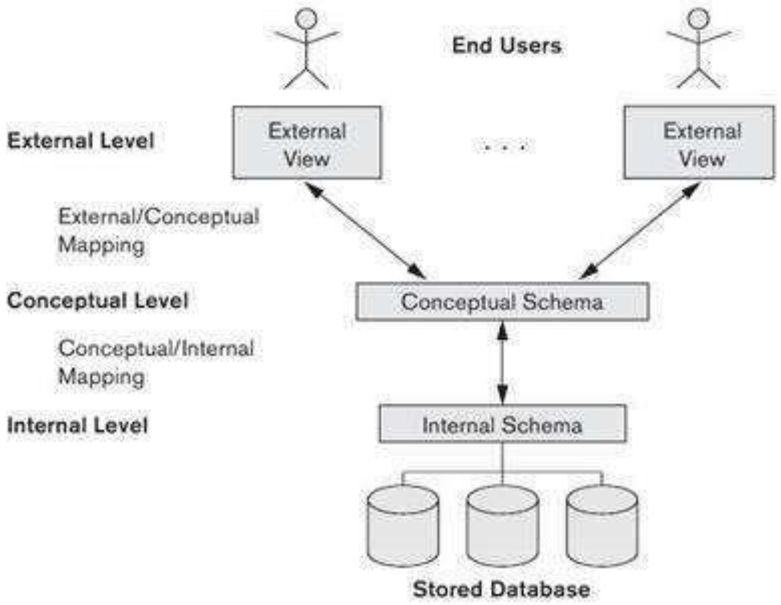


Fig 1.16

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories −

Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.