INTRODUCTION TO DBMS

~~UNIT I~~

Chapter-1

INTRODUCTION

1.1 An Overview of database Management System:

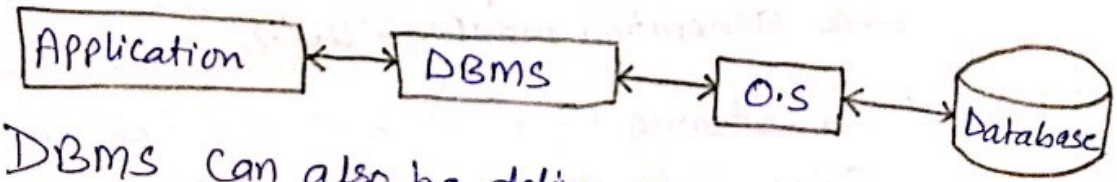
"A database Management System (DBMS) is a collection of interrelated data and a set of programs to access those data." The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Database is the combination of two words

Database + Management System = DBMS

A Database is a collection of related information Stored, so that it is available to many users for different purposes.

On the other hand Management system is a collection of Programs that enables users to create and maintain the database.



DBMS can also be define as an interface between the application program and the operating system to access or manipulate that database.

A DBMS is a software system that allows access to data contained in the database. It's objective is to provide a convenient and effective method of defining, storing and retrieving the information contained in the database. The DBMS interfaces with application programs so that the data contained in the database can be used by multiple applications & users.

1.2 Applications of DBMS

Some of the important application fields of DBMS are:

1. Banking: For maintaining customer information, accounts, loans and banking transactions.
2. Universities: For maintaining student records, course registration and grades.
3. Railway Reservation: For checking the availability of reservation in different trains, tickets, etc.
4. Airlines: For reservation and schedule information.
5. Telecommunication: For keeping records of calls made, generating monthly bills etc.
6. Finance: For storing information about holdings, sales and purchase of financial instruments.
7. Sales: For customer, product and purchase information.

Advantages & Disadvantages of DBMS.

1.3 Advantages:

1. Less Redundancy and Inconsistency: DBA has the Centralized control of the data, so it will avoid the inconsistency because now we have to make the changes at one place only.
2. Data Independence: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.
3. Reduced Application development time: Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS.
4. Efficient data Accessing: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This features is especially important if the data is stored on external storage devices.
5. Data Integrity and Security: A DBMS provides the integrity constraints on the data. It also provides the access control to the data. Not every user of the database system should be allowed to access all the data.

Easy Data Administration: When several users share the data, centralizing the administration of data can offer significant improvements.

7: Concurrent access & Crash Recovery: A DBMS schedules concurrent 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 it also ② Protect users from the

effects of the system failure.

1.4 Disadvantages:

1. Problems associated with Centralization: Centalization ~~requires~~ increases the security problems and disruption due to the downtimes and failures.

2. Cost of Software: Today's there are several software which are very costly, Hence from Economic point of view it is the drawbacks.

3. Cost of Hardware: It incurred in the application of DBMS is its major disadvantage. Cost of the H/W is also one of the major drawback.

4. Complexity of backup and recovery: DBMS provides the centralization of the data, which ~~gives~~ requires the adequate backups of the data so that in case of failure, the data can be recovered. Hence backup problem is also the drawback.

1.5 Database System Versus File systems

Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.

On the other hand file processing system is supported by a convential operating system. The system stores permanent records in various files,

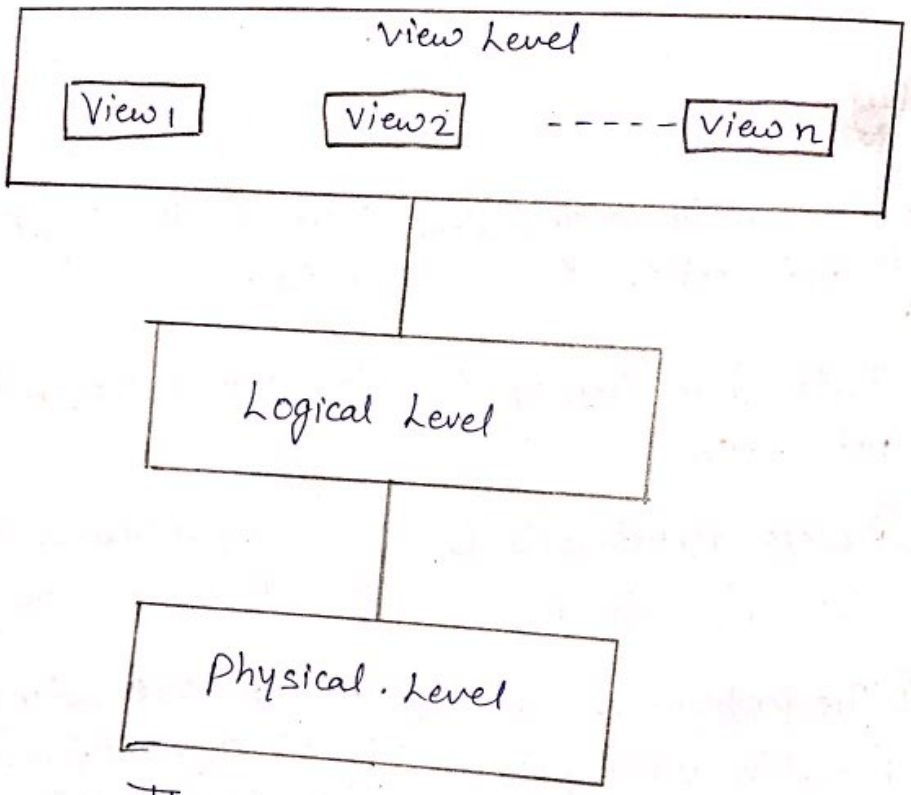
efficient series of operations.

and it needs different application programs to extract records from, and add records to the appropriate files. Before DBMS came along, organizations usually stored information in such systems.

DBMS	File Processing System
1. Data Redundancy problem is not found.	Here Redundancy problem exist.
2. Data Inconsistency does not exist.	It also exist in this.
3. Accessing database is easier.	Accessing database is Comparatively difficult.
4. The problem of data Isolation is not found in it.	Data are scattered in various files in different formats. Writing new program to retrieve appropriate data is difficult.
5. Atomicity & Integrating problems are not found.	Atomicity & Integrating problems are found.
6. Security of data is in DBMS.	Here it is not good.
7. Concurrent Access and Crash Recovery.	Here there is no Concurrent access & no Crash Recovery.

1.6 Database system Concepts and Architecture

Data Abstraction:



The three levels of data abstraction.

1. Physical Level: This is the lowest level of abstraction and it describes how the data are actually stored. The physical level describes complex low-level data structures in details.
2. Logical Level: This is the next-higher level of abstraction and it describes ~~how~~ what data are stored in the database, and what relationship exist among those data. The

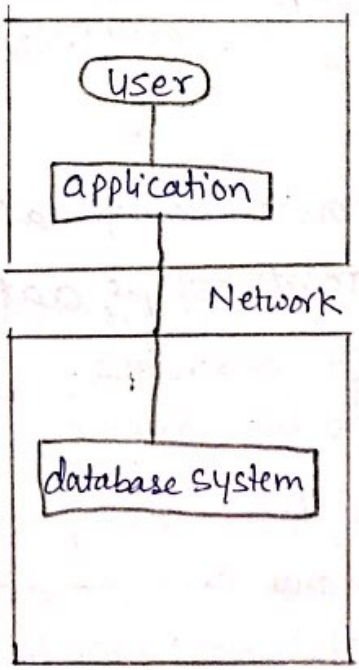
Logical level thus describes the entire database in terms of a small number of relatively simple structures.

3. View Level: This is the highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

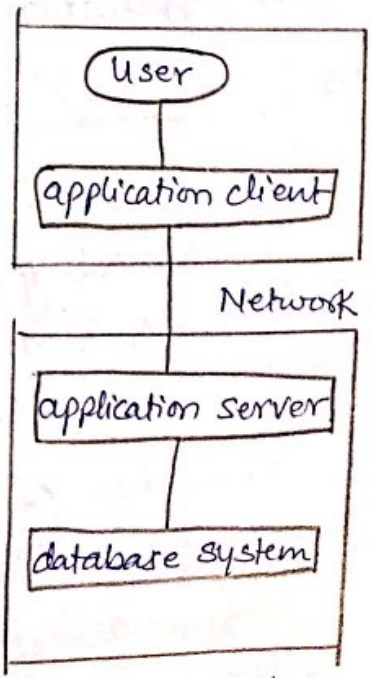
1.7 Application Architecture of DBMS

Basically there are two types of architecture of DBMS:

1. Two-tier architecture.
2. Three-tier architecture.



2-tier architecture.



3-tier architecture

1.7.1 In a two-tier architecture, the application is partitioned into a component that resides at the client machine, which invokes database system funtionality at the server machine through query language statements. Application program interface standards like ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) are used for interaction between the client and the server.

1.7.2 In a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. Three-tier applications are more appropriate for large applications, and for applications that run on the world wide web.

1.8 Data Models

Data Model is a collection of Conceptual tools for describing data, data relationships, data Semantics and consistency constraints.
A Data Model can also be define as the collection of high-level data description Constructs that hide many low-level Storage details. A data model provides a way to describe the design of a database at the physical, logical & view levels.

There are five types of data Models:

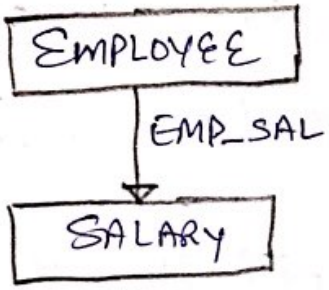
1. Relational dataModel. (New-2)
2. Object-oriented Data Model. (New-4) ER-model (New-3)
3. Network Data Model. (New 1)
4. Hierarchical Data Model. (oldest)
5. Object Relational Data Model. (New-5)

1.8.1 Relational Data Model: The Relational Model uses a collection of tables to represent both data and the relationship among those data. Each table has multiple columns, and each column has a unique name. This model uses the certain mathematical operations from relational Algebra and relational calculus on the relation, such as Projection, Union and joins etc.

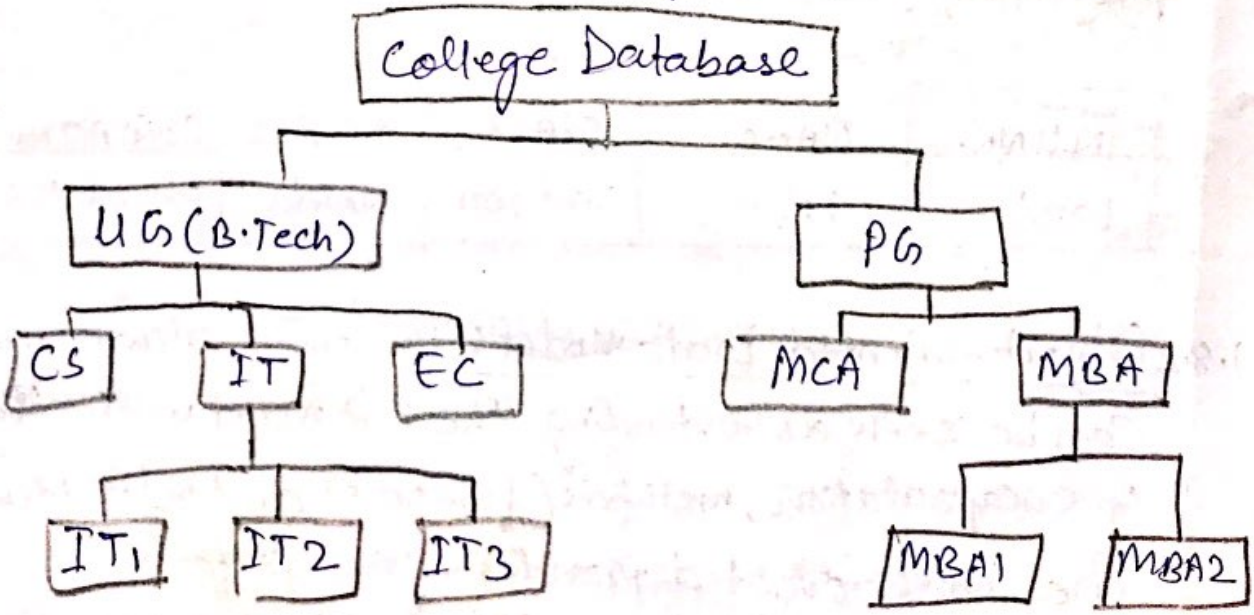
ROLL NO.	NAME	CLASS	ADDRESS	CONTACT NO.
1001	RAM	3rd year	Noida	09426 1846

1.8.2 Object-oriented DataModel: The object-oriented model can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-oriented provides higher performance mangement of objects, and it enables better management of the inter-relationships between objects.

1.8.3 Network Model: The network model permits the modeling of many to many relationships in data. The basic data model construct in the network model is the set construct.



1.8.4 Hierarchical Model: The hierarchical model organizes data in a tree structure. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These are 1:N mapping between record types.



E-R Model: Entity relationship Model.

Chapter – 2

Data Modeling using the ER Model.

2.1 ER Model Concepts

The Entity-relationship (E-R) model is a high-level data model. It is based on a perception of a real world that consists of a collection of basic objects, called entities, and of relationships among these objects. It was developed to facilitate database design by allowing specification of an enterprise schema, which represent the overall logical structure of a database.

An entity: is a "thing" or "object" in the real world that is distinguishable from all other objects.

OR

Anything about which we store information is called an Entity.

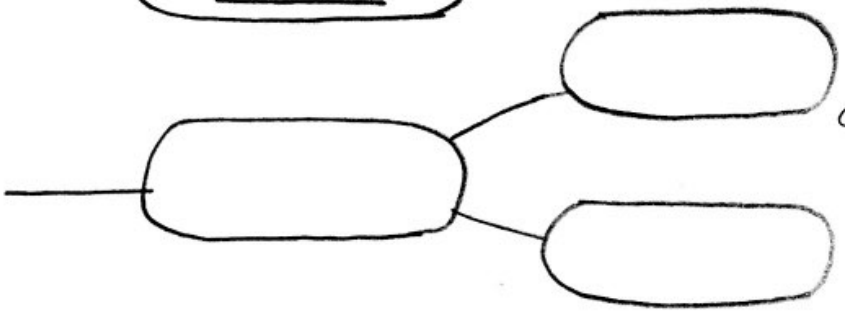
An entity set: is a set of entities of the same type that share the same properties, or attributes. The set of all persons who are customers at a given bank, for eg:- can be defined as the entity set customer.

Attributes: An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. Attributes describe the entity to which they are associated.

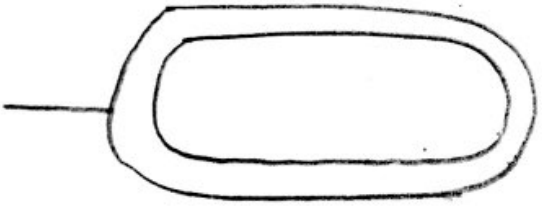
2.2 Notation for E-R Diagram:

ATTRIBUTES

KEY ATTRIBUTES

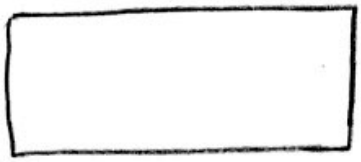


COMPOSITE ATTRIBUTES

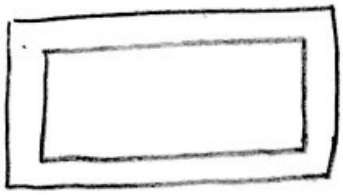


MULTIVALUED ATTRIBUTES

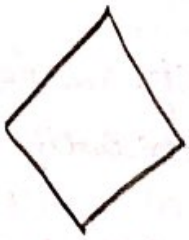
DERIVED ATTRIBUTES



STRONG ENTITY SET



WEAK ENTITY SET



RELATIONSHIP

LINKS

Single-valued and multivalued attributes: The attributes in our examples all have a single value for a particular entity. For instances, the loan-number attributes for a specific loan entity refers to only one loan numbers. Such attributes are said to be "single valued". An employee may have zero, one, or several phone numbers, and different employees may have different numbers of phones. This type of attribute is said to be "multivalued".

Simple and composite attributes: Simple attributes are those, which are not divided into subparts. Whereas composite attributes, are those which can be divided into subparts. Composite attributes help us to group together related attributes,

Derived Attributes: The value for this type of attribute can be derived from the values of other related attributes or entities.

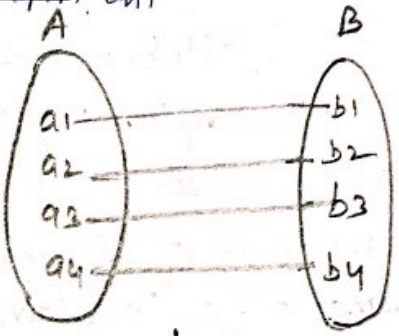
An attribute takes a null value when an entity does not have a value for it. The null value may indicate "not applicable".

2.3 Mapping Constraints

Mapping Cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
Mapping cardinalities are most useful in describing binary relationship sets.
There are 4 types.

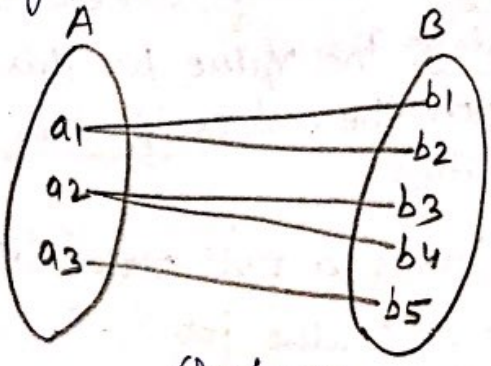
2.3.1. One to one : An entity in A is associated with atmost one entity in B, and an entity in B is associated with atmost one entity in A.

~~2. One to Many : An ent~~



One to one

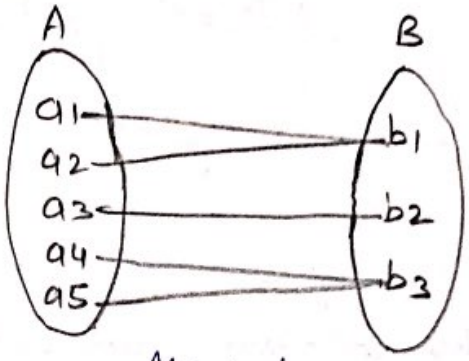
2.3.2 One to Many : An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with atmost one entity in A.



One to many

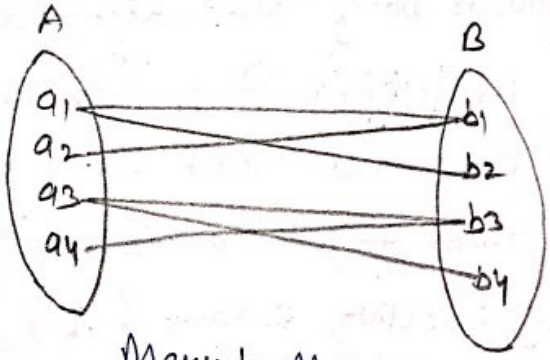
~~2.26~~

2.3.3 Many to one: An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.



Many to one

2.3.4 Many to Many: An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



Many to Many

2.4 KEYS: A key is a value which can always be used to uniquely identify an object instance. It allows us to identify a set of attributes that suffice to distinguish entities from each other.

It is basically of 3 types:

① Super key ② Candidate key ③ Primary key.