



Tulsiramji Gaikwad-Patil College of Engineering & Technology

Department of Information Technology

Subject Notes

Academic Session: 2018 – 2019

Subject: DBMS

Semester: I

UNIT: I

UNIT 1: Syllabus

Database Concepts –Introduction, Data, Information, Metadata, Terminology of file, Association between fields, Entities and their attributes, relationship, records and files, abstraction and data integration, Association between files(record types) conventional file processing systems, Database System, Components of Database Management system-(Classification of DBMS users, DBMS facilities, Structure of a DBMS, Database access), Advantages and Disadvantages of DBMS, Three level architecture proposal for DBMS, Mapping between views, Data Independence.

(A) Explain the following in the detail

i)Concurrency control: In [information technology](#) and [computer science](#), especially in the fields of [computer programming](#), [operating systems](#), [multiprocessors](#), and [databases](#), **concurrency control** ensures that correct results for [concurrent](#) operations are generated, while getting those results as quickly as possible.

Computer systems, both [software](#) and [hardware](#), consist of modules, or components. Each component is designed to operate correctly, i.e., to obey or to meet certain consistency rules. When components that operate concurrently interact by messaging or by sharing accessed data (in [memory](#) or [storage](#)), a certain component's consistency may be violated by another component. The general area of concurrency control provides rules, methods, design methodologies, and [theories](#) to maintain the consistency of components operating concurrently while interacting, and thus the consistency and correctness of the whole system. Introducing concurrency control into a system means applying operation constraints which typically result in some performance reduction. Operation consistency and correctness should be achieved with as good as possible efficiency, without reducing performance below reasonable levels. Concurrency control can require significant additional complexity and overhead in a [concurrent algorithm](#) compared to the simpler [sequential algorithm](#).

For example, a failure in concurrency control can result in [data corruption](#) from [torn read or write operations](#)

ii) Atomicity property: In [database systems](#), **atomicity** (or **atomicness**^[citation needed]; from [Greek](#) *atomos, undividable*) is one of the [ACID transaction](#) properties. An **atomic transaction** is an *indivisible* and *irreducible* series of database operations such that either *all* occur, or *nothing* occurs.^[1] A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright. As a consequence, the transaction cannot be observed to be in progress by another database client. At one moment in time, it has not yet happened, and at the next it has already occurred in whole (or nothing happened if the transaction was cancelled in progress).

An example of an atomic transaction is a monetary transfer from bank account A to account B. It consists of two operations, withdrawing the money from account A and saving it to account B. Performing these operations in an atomic transaction ensures that the database remains in a [consistent state](#), that is, money is not lost nor created if either of those two operations fail.^[2]

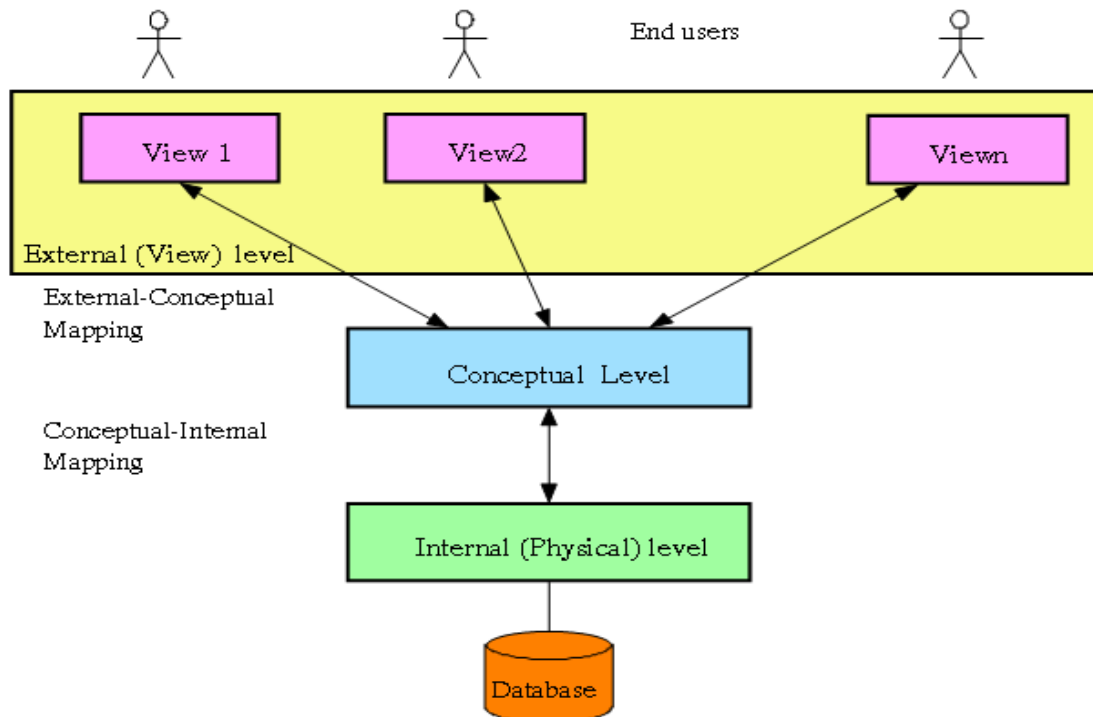
(B) Give the level architecture proposal for DBMS ?

Ans: Objective of three level architecture proposal for DBMS

- All users should be able to access same data.
- A user's view is immune to changes made in other views.
- Users should not need to know physical database storage details.
- DBA should be able to change database storage structures without affecting the users' views.
- Internal structure of database should be unaffected by changes to physical aspects of storage.
- DBA should be able to change conceptual structure of database without affecting all users.

The architecture of a database management system can be broadly divided into three levels :

- a. **External level**
- b. **Conceptual level**
- c. **Internal level**



Above three points are explain in detail given bellow:-

External Level:

This is the highest level, one that is closest to the user. It is also called the user view. The user view is different from the way data is stored in the database. This view describes only a part of the actual database. Because each user is not concerned with the entire database, only the part that is relevant to the user is visible. For example, end users and application programmers get different external views.

Each user uses a language to carry out database operations. The application programmer uses either a conventional third-generation language, such as COBOL or C, or a fourth-generation language specific to the DBMS, such as visual FoxPro or MS Access.

The end user uses a query language to access data from the database. A query language is a combination of three subordinate language :

Data	Definition	Language	(DDL)
Data	Manipulation	Language	(DML)
Data	Control	Language	(DCL)

The data definition language defines and declares the database object, while the data manipulation language performs operations on these objects. The data control language is used to control the user's access to database objects.

Conceptual Level: - This level comes between the external and the internal levels. The conceptual level represents the entire database as a whole, and is used by the DBA. This level is the view of the data "as it really is". The user's view of the data is constrained by the language that they are using. At the conceptual level, the data is viewed without any of these constraints

Internal Level: - This level deals with the physical storage of data, and is the lowest level of the architecture. The internal level describes the physical sequence of the stored records

So that objective of three level of architecture proposal for DBMS are suitable explain in above.

(C) Describe the structure of DBMS?

Ans: DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

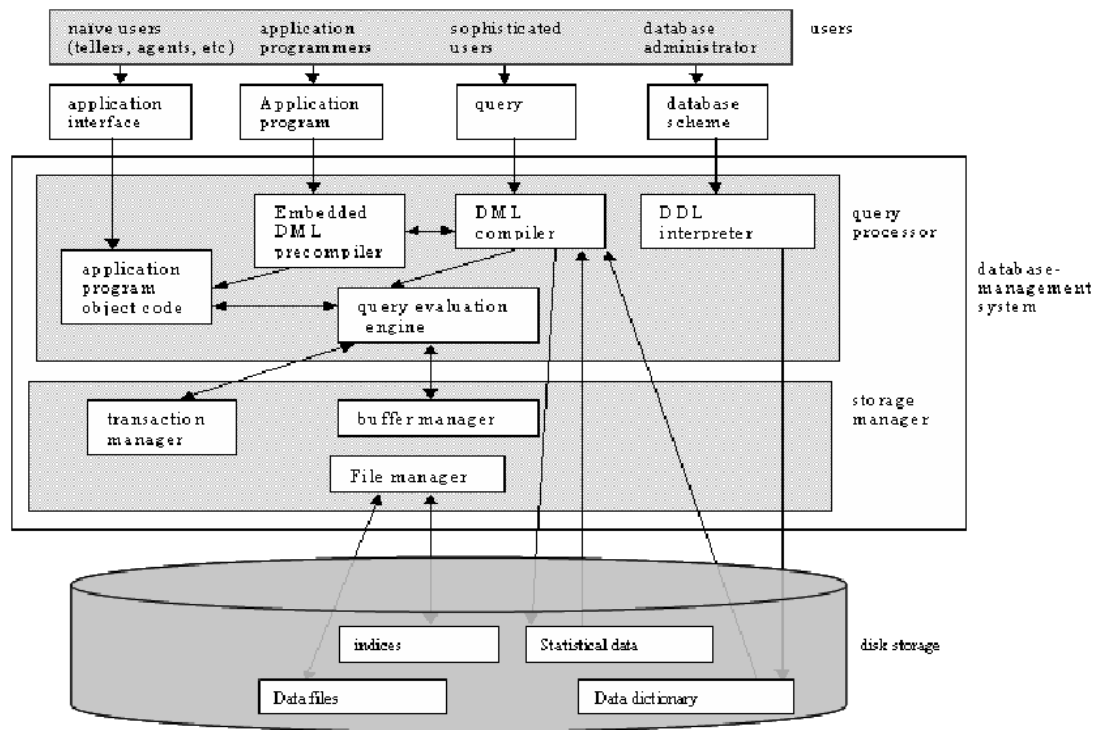


Fig: Structure of Database Management System

Components of DBMS: -

DDL Compiler

Data Manager

File Manager

Disk Manager

Query Processor

Telecommunication System

Data Files

Data Dictionary

Access Aids

1. DDL Compiler - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. DML Compiler and Query optimizer - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. Data Manager - The Data Manager is the central software component of the DBMS also known as Database Control System.

The Main Functions Of Data Manager Are: –

Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.

- Controls DBMS information access that is stored on disk.
- It also controls handling buffers in main memory.
- It also enforces constraints to maintain consistency and integrity of the data.
- It also synchronizes the simultaneous operations performed by the concurrent users.

- It also controls the backup and recovery operations.

4. Data Dictionary - Data Dictionary is a repository of description of data in the database. It contains information about

1. Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
2. Relationships between database transactions and data items referenced by them which is useful in determining which transactions are affected when certain data definitions are changed.
3. Constraints on data i.e. range of values permitted.
4. Detailed information on physical database design such as storage structure, access paths, files and record sizes.
5. Access Authorization - is the Description of database users their responsibilities and their access rights.
6. Usage statistics such as frequency of query and transactions.
7. Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.
8. Importance of Data Dictionary -
9. Data Dictionary is necessary in the databases due to following reasons:
10. It improves the control of DBA over the information system and user's understanding of use of the system.
11. • It helps in document ting the database design process by storing documentation of the result of every design phase and design decisions.

5. Data Files - It contains the data portion of the database.

6. Compiled DML - The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

7. End Users : The users of the database system can be classified in the following groups, depending on their degree of expertise or the mode of their interactions with the DBMS.

1. Naïve users
2. Online Users
3. Application Programmers
4. Database administrator

i) **Naïve User:** Naive users who need not have aware of the present of the database system or any other system.. A user of an automatic teller falls under this category. The user is instructed through each step of a transaction; he or she responds by pressing a coded key or entering a numeric value. The operations that can be performed by this calls of users are very limited and affect a precise portion of the database; in case of the user of the automatic teller machine, only one or more of her or his own accounts. Other such naive users are where the type and range of response is always indicated to the user. Thus, a very competent database designer could be allowed to use a particular database system only as a naive user.

ii) **Online users:** There are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the presence of the database system and may have acquired a certain amount of expertise in the limited interaction they are permitted with the database through the intermediate application program. The more sophisticated of these users may also use a data manipulation language to manipulate the database directly. On-line users can also be naive users requiring help such as menus.

iii) **Application Users:** Professional programmers who are responsible for developing application programs or user interfaces utilized by the naive and online users fall into this category. The application programs could be written in a general purpose programming language such as Assembler C, COBOL, FORTRAN, PASCAL, or PL/I and include the commands required to manipulate the database.

iv) **Database Administrator:** Centralized control of the database is exerted by a person or group of persons under the supervision of a high level administrator. This person or group is referred to as the database administrator (DBA). They are users who are the most familiar with the database and are responsible for creating, modifying and maintaining its three levels.

The DBA us the custodian of the data and controls the database structure. The DBA administers the three levels of the database and in consultation with the overall user community, sets up the definition of the global view or conceptual level of the database. The DBA further specifies the external view of the various users and applications and is responsible for definition and implementation of the internal level, including the storage structure and access methods to be used for the optimum performance of the DBMS.

(D) What are the advantages of using a DBMS over the conventional file processing system?

Ans: A database is a collection of non-redundant data which can be shared by different application systems stresses the importance of multiple applications, data sharing the spatial database becomes a common resource for an agency implies separation of physical storage from use of the data by an application program, i.e. program/data independence the user or programmer or application specialist need not know the details of how the data are stored such details are "transparent to the user" changes can be made to data without affecting other components of the system. e.g. change format of data items (real to integer, arithmetic operations) change file structure (reorganize data internally or change mode of access) relocate from one device to another, e.g. from optical to magnetic storage, from tape to disk

Advantages:

1. Control of data redundancy.
2. Data consistency
3. More information from the same amount of data.
4. Sharing of data.
5. Improved data integrity.
6. Improved security.
7. Enforcement of standards.
8. Economy of scale.

1. Controlling Data Redundancy - In the conventional file processing system,

Every user group maintains its own files for handling its data files. This may lead to

- Duplication of same data in different files.
- Wastage of storage space, since duplicated data is stored.
- Errors may be generated due to duplication of the same data in different files.
- Time in entering data again and again is wasted.
- Computer Resources are needlessly used.
- It is very difficult to combine information

2. Elimination of Inconsistency - In the file processing system information is duplicated throughout the system. So changes made in one file may be necessary be carried over to another file. This may lead to inconsistent data. So we need to remove this duplication of data in multiple file to eliminate inconsistency.

3. Better service to the users - A DBMS is often used to provide better services to the users. In conventional system, availability of information is often poor, since it normally difficult to obtain information that the existing systems were not designed for. Once several conventional systems are combined to form one centralized database, the availability of information and its update ness is likely to improve since the data can now be shared and DBMS makes it easy to respond to anticipated information requests.

Centralizing the data in the database also means that user can obtain new and combined information easily that would have been impossible to obtain otherwise. Also use of DBMS should allow users that don't know programming to interact with the data more easily, unlike file processing system where the programmer may need to write new programs to meet every new demand.

4. Flexibility of the System is improved - Since changes are often necessary to the contents of the data stored in any system, these changes are made more easily in a centralized database than in a conventional system. Applications programs need not to be changed on changing the data in the database.

5. Integrity can be improved - Since data of the organization using database approach is centralized and would be used by a number of users at a time. It is essential to enforce integrity-constraints. In the conventional systems because the data is duplicated in multiple files so updating or changes may sometimes lead to entry of incorrect data in some files where it exists.

6. Standards can be enforced - Since all access to the database must be through DBMS, so standards are easier to enforce. Standards may relate to the naming of data, format of data, structure of the data etc. Standardizing stored data formats is usually desirable for the purpose of data interchange or migration between systems.

7. Security can be improved - In conventional systems, applications are developed in an adhoc/temporary manner. Often different system of an organization would access different components of the operational data, in such an environment enforcing security can be quiet difficult. Setting up of a database makes it easier to enforce security restrictions since data is now centralized. It is easier to control who has access to what parts of the database. Different checks can be established for each type of access (retrieve, modify, delete etc.) to each piece of information in the database.

8. Organization's requirement can be identified - All organizations have sections and departments and each of these units often consider the work of their unit as the most important and therefore consider their need as the most important. Once a database

has been setup with centralized control, it will be necessary to identify organization's requirement and to balance the needs of the competing units. So it may become necessary to ignore some requests for information if they conflict with higher priority need of the organization. It is the responsibility of the DBA (Database Administrator) to structure the database system to provide the overall service that is best for an organization.

9. Overall cost of developing and maintaining systems is lower - It is much easier to respond to unanticipated requests when data is centralized in a database than when it is stored in a conventional file system. Although the initial cost of setting up of a database can be large, one normally expects the overall cost of setting up of a database, developing and maintaining application programs to be far lower than for similar service using conventional systems, Since the productivity of programmers can be higher in using non-procedural languages that have been developed with DBMS than using procedural languages.
10. Data Model must be developed - Perhaps the most important advantage of setting up of database system is the requirement that an overall data model for an organization be build. In conventional systems, it is more likely that files will be designed as per need of particular applications demand. The overall view is often not considered. Building an overall view of an organization's data is usual cost effective in the long terms.
11. Provides backup and Recovery - Centralizing a database provides the schemes such as recovery and backups from the failures including disk crash, power failures, software errors which may help the database to recover from the inconsistent state to the state that existed prior to the occurrence of the failure, though methods are very complex.

E) Explain with diagram structure of DBMS and steps involved in database access.

Ans: DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

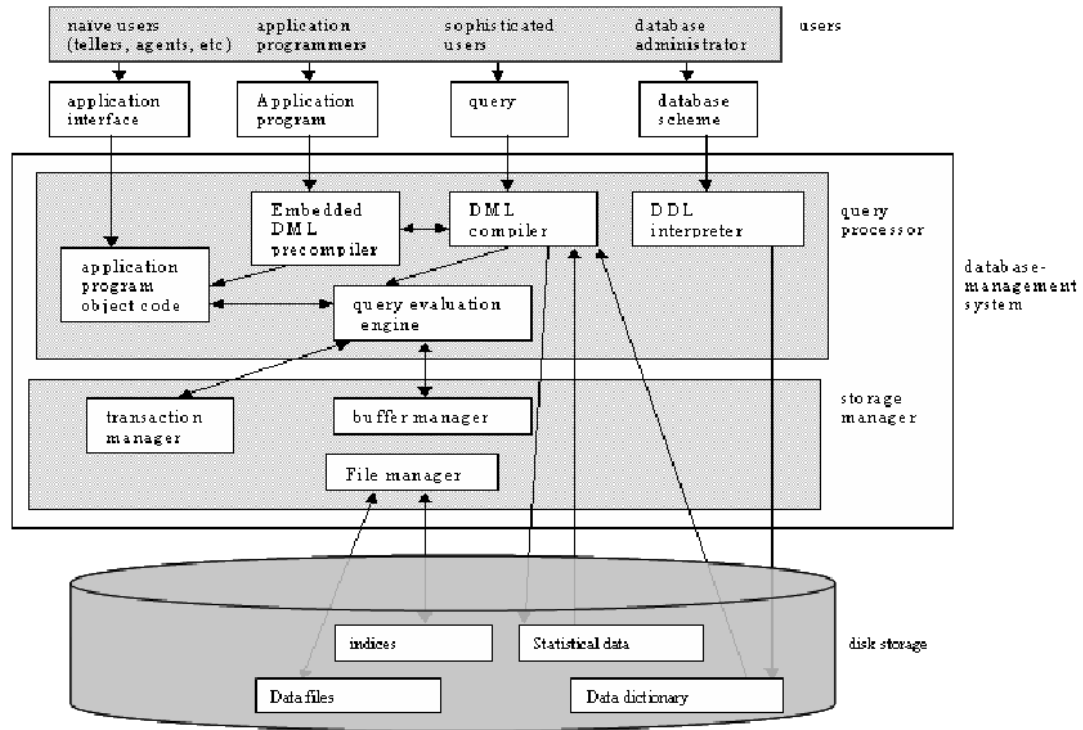


Fig: Structure of Database Management System

Components of DBMS

DDL Compiler

Data Manager

File Manager

Disk Manager

Query Processor

Telecommunication System

Data Files

Data Dictionary

Access Aids

1. DDL Compiler - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. DML Compiler and Query optimizer - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. Data Manager - The Data Manager is the central software component of the DBMS also known as Database Control System.

The Main Functions Of Data Manager Are: –

Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.

- Controls DBMS information access that is stored on disk.
- It also controls handling buffers in main memory.
- It also enforces constraints to maintain consistency and integrity of the data.
- It also synchronizes the simultaneous operations performed by the concurrent users.
- It also controls the backup and recovery operations.

4. Data Dictionary - Data Dictionary is a repository of description of data in the database. It contains information about

12. Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.

13. Relationships between database transactions and data items referenced by them which is useful in determining which transactions are affected when certain data definitions are changed.

14. Constraints on data i.e. range of values permitted.

15. Detailed information on physical database design such as storage structure, access paths, files and record sizes.
16. Access Authorization - is the Description of database users their responsibilities and their access rights.
17. Usage statistics such as frequency of query and transactions.
18. Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.
19. Importance of Data Dictionary -
20. Data Dictionary is necessary in the databases due to following reasons:
21. It improves the control of DBA over the information system and user's understanding of use of the system.
22. • It helps in document ting the database design process by storing documentation of the result of every design phase and design decisions.
23. It helps in searching the views on the database definitions of those views.

5. Data Files - It contains the data portion of the database.

6. Compiled DML - The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

7. End Users : The users of the database system can be classified in the following groups, depending on their degree of expertise or the mode of their interactions with the DBMS.

1. Naïve users
2. Online Users
3. Application Programmers
4. Database administrator

i)Naïve User: Naive users who need not have aware of the present of the database system or any other system.. A user of an automatic teller falls under this category. The user is instructed through each step of a transaction; he or she responds by pressing a coded key or entering a numeric value. The operations that can be performed by this calls of users are very limited and affect a precise portion of the database; in case of the user of the automatic teller machine, only one or more of her or his own accounts. Other such naive users are where the type and range of response is always indicated to the user. Thus, a very competent database designer could be allowed to use a particular database system only as a naive user.

ii)Online users: There are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the presence of the database system and may have acquired a certain amount of expertise in the limited interaction they are permitted with the database through the intermediate application program. The more sophisticated of these users may also use a data manipulation language to manipulate the database directly. On-line users can also be naive users requiring help such as menus.

iii)Application Users: Professional programmers who are responsible for developing application programs or user interfaces utilized by the naive and online users fall into this category. The application programs could be written in a general purpose programming language such as Assembler C, COBOL, FORTRAN, PASCAL, or PL/I and include the commands required to manipulate the database.

iv)Database Administrator: Centralized control of the database is exerted by a person or group of persons under the supervision of a high level administrator. This person or group is referred to as the database administrator (DBA). They are users who are the most familiar with the database and are responsible for creating, modifying and maintaining its three levels.

The DBA us the custodian of the data and controls the database structure. The DBA administers the three levels of the database and in consultation with the overall user community, sets up the definition of the global view or conceptual level of the database. The DBA further specifies the external view of the various users and applications and is responsible for definition and implementation of the internal level, including the storage structure and access methods to be used for the optimum performance of the DBMS.

F) Write short note on :-

- (i) DBMS user:
- (ii) DBMS facilities

Ans: **DBMS user :**

1. Application programmers or Ordinary users
2. End users
3. Database Administrator (DBA)
4. System Analyst

1. Application programmers or Ordinary users: These users write application programs to interact with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language. Such

programs access the database by issuing the appropriate request, typically a SQL statement to DBMS.

2. End Users: End users are the users, who use the applications developed. End users need not know about the working, database design, the access mechanism etc. They just use the system to get their task done. End users are of two types:

a) Direct users b) Indirect users

a) Direct users: Direct users are the users who see the computer, database system directly, by following instructions provided in the user interface. They interact using the application programs already developed, for getting the desired result. E.g. People at railway reservation counters, who directly interact with database.

b) Indirect users: Indirect users are those users, who desire benefit from the work of DBMS indirectly. They use the outputs generated by the programs, for decision making or any other purpose. They are just concerned with the output and are not bothered about the programming part.

3. Database Administrator (DBA): Database Administrator (DBA) is the person which makes the strategic and policy decisions regarding the data of the enterprise, and who provide the necessary technical support for implementing these decisions. Therefore, DBA is responsible for overall control of the system at a technical level. In 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).

4. System Analyst: System Analyst determines the requirement of end users, especially naïve and parametric end users and develops specifications for transactions that meet these requirements. System Analyst plays a major role in database design, its properties; the structure prepares the system requirement statement, which involves the feasibility aspect, economic aspect, technical aspect etc. of the system.

(ii) DBMS facilities

Typically, a DBMS provides the following facilities

■ Data Definition Language (DDL)

It allows a database designer to define the database using a **Data Definition Language (DDL)** provided for the particular DBMS. The DDL allows the designer to specify the data types and structures, and the constraints on the data to be stored in the database (see Figure 5.3 on page 60).

■ **Data Manipulation Language (DML)**

It allows users to insert, update, delete and retrieve data from the database through a **Data Manipulation Language (DML)**. Having a central repository for all data and data descriptions allows the DML to provide a general enquiry facility to this data, called a **query language**. Using a query language, directly or indirectly, enables new lines of enquiry to be constructed and satisfied quickly. A query language is sufficiently high level to allow non-technical personnel to use it, easily. The most common query language is the **Structured Query Language (SQL** – pronounced ‘S-Q-L’).

■ **View Mechanism**

The DBMS provides a **view mechanism** that allows each user to have his or her own view of the database. The DDL is used to define a view that is a subset of the database. For example, a program to print a list of staff names, their qualifications and subjects that they teach would be granted a view of the database that included just these data items and excluded all others as shown in Figure 2.2.

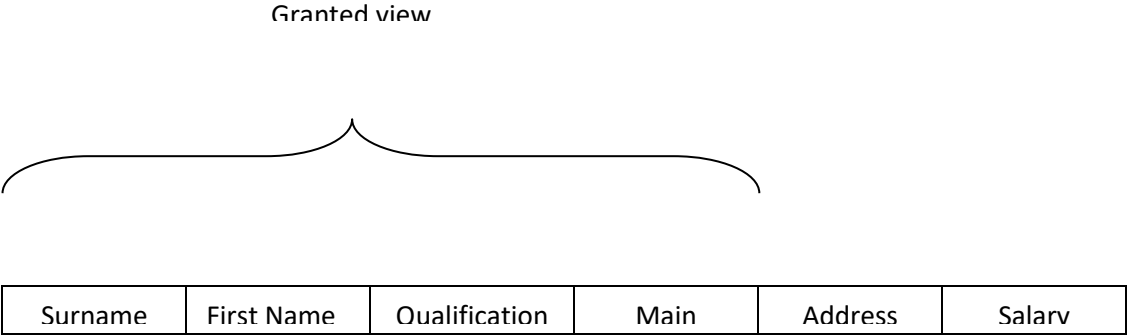


Figure 2.2 Restricting an application’s view of the database

■ **Multiple indexes**

An index is a mechanism for reducing the time taken to find a specific item of data in a database. A database index works in a similar way to a user of the index in this book. If you want to use this book to find out about the topic “multiple indexes“ then you have a choice. You could open this book at the beginning and work your way, line by line, page by page, through the text looking for the phrase “multiple indexes”. Eventually, your search will be successful on page 19. Luckily for you, by choosing the phrase “multiple indexes” you only had to search 19 pages, had you chosen the phrase “solutions to questions” you would not have struck lucky until page 107. However, there is a much quicker way to search based upon searching the index pages at the back of this book. Searching the index line by line, page by page for the phrase “multiple indexes” is much quicker. This index stores the number of the page that we need to visit to read about “multiple indexes”. So, having found the correct entry in the index we can now turn directly to the desired page.

An index in a database can store each value of an indexed data item (field), e.g. student enrolment number, together with the page number in the storage medium where the data belonging to this value is stored. For example, information stored in a database about a particular student such as surname, home address, et cetera may be quickly found if an index has been created that stores every student’s enrolment number and the location of the corresponding information.

Indexes may also be created on other fields of a student’s record, e.g. surname, address. However, since some fields such as the surname are unlikely to be unique, the entries in the index may reference more than one location, just like the entries in the index for this book. An index on a unique field is known as a **primary index** whereas an index on a non-unique field is known as a **secondary index**.

■ Indexing Overheads

Indexes have to be constantly kept up to date. When a new data value is added or modified the corresponding index must be updated. This takes time. This is called an update overhead.

G) Explain three level architecture proposal of DBMS with its disadvantages .

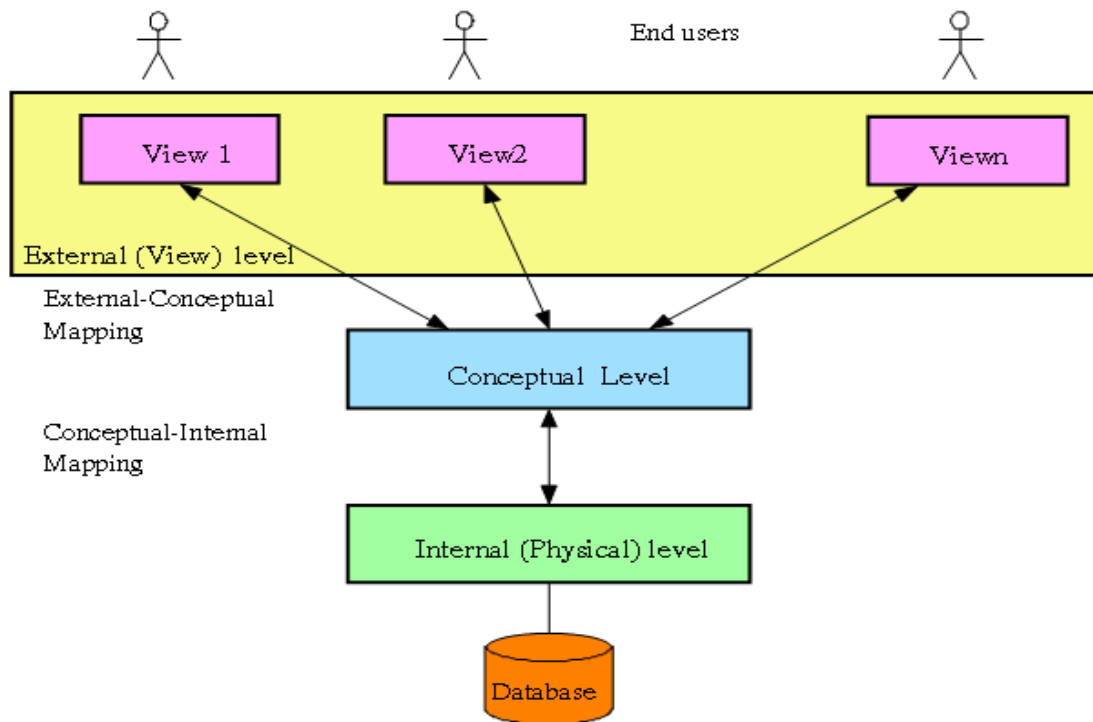
Ans: Objective of three level architecture proposal for DBMS

- All users should be able to access same data.
- A user's view is immune to changes made in other views.
- Users should not need to know physical database storage details.

- DBA should be able to change database storage structures without affecting the users' views.
- Internal structure of database should be unaffected by changes to physical aspects of storage.
- DBA should be able to change conceptual structure of database without affecting all users.

The architecture of a database management system can be broadly divided into three levels :

- External level**
- Conceptual level**
- Internal level**



Above three points are explain in detail given bellow:-

External Level:

This is the highest level, one that is closest to the user. It is also called the user view. The user view is different from the way data is stored in the database. This view describes only a part of the actual database. Because each user is not concerned with the entire database, only the part that is relevant to the user is visible. For example, end users and application programmers get different external views.

Each user uses a language to carry out database operations. The application programmer uses either a conventional third-generation language, such as COBOL or C, or a fourth-generation language specific to the DBMS, such as visual FoxPro or MS Access.

The end user uses a query language to access data from the database. A query language is a combination of three subordinate language :

Data	Definition	Language	(DDL)
Data	Manipulation	Language	(DML)
Data	Control	Language	(DCL)

The data definition language defines and declares the database object, while the data manipulation language performs operations on these objects. The data control language is used to control the user's access to database objects.

Conceptual Level: - This level comes between the external and the internal levels. The conceptual level represents the entire database as a whole, and is used by the DBA. This level is the view of the data "as it really is". The user's view of the data is constrained by the language that they are using. At the conceptual level, the data is viewed without any of these constraints

Internal Level: - This level deals with the physical storage of data, and is the lowest level of the architecture. The internal level describes the physical sequence of the stored records

So that objective of three level of architecture proposal for DBMS are suitable explain in above.

Disadvantages of DBMS:-

Although there are many advantages of DBMS, the DBMS may also have some minor disadvantages. These are:

- **Cost of Hardware and Software**

A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also very costly,.

- **Cost of Data Conversion**

When a computer file-based system is replaced with database system, the data stored into data file must be converted to database file. It is very difficult and costly method to convert data of data file into database. You have to hire database system designers along with application programmers. Alternatively, you have to take the services of some software house. So a lot of money has to be paid for developing software.

- **Cost of Staff Training**

Most database management system are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to be paid a lot of amount for the training of staff to run the DBMS.

- **Appointing Technical Staff**

The trained technical persons such as database administrator, application programmers, data entry operations etc. are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

- **Database Damage**

In most of the organization, all data is integrated into a single database. If database is damaged due to electric failure or database is corrupted on the storage media, the your valuable data may be lost forever.

H) Explain the following term :-

(i) Data Independence

Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user application to make changes in the definition and organization of data.

Physical data independence deals with hiding the details of the storage structure from user applications. The application should not be involved with these issues, since there is no difference in the operation carried out against the data.

The data independence and operation independence together gives the feature of data abstraction. There are two levels of data independence.

Logical data independence:

- the capacity to change the conceptual schema without having to change external schema or application prgms

ex: Employee (E#, Name, Address, Salary)

A view including only E# and Name is not affected by changes in any other attributes.

Physical data independence:

- the capacity to change the internal schema without having to change the conceptual (or external) schema
- internal schema may change to improve the performance

(e.g., creating additional access structure)

- easier to achieve logical data independence, because application programs are dependent on logical structures

(ii) Instance and Schema :

Schema:

A schema is plan of the database that give the names of the entities and attributes and the relationship among them. A schema includes the definition of the database name, the record type and the components that make up the records. Alternatively, it is defined as a framework into which the values of the data items are fitted. The values fitted into the frame-work changes regularly but the format of schema remains the same e.g., consider the database consisting of three files ITEM, CUSTOMER and SALES. The data structure diagram for this schema is shown in following figure:

```

Schema name is ITEM_SALES_REC

type    ITEM = record
        ITEM_ID: string;
        ITEM_DESC: string;
        ITEM_COST: integer;
end
} Attributes/data items

type    CUSTOMER = record
        CUSTOMER_ID = integer;
        CUSTOMER_NAME = string;
        CUSTOMER_ADD = string;
        CUSTOMER_CITY = string;
        CUSTOMER_BAL = integer;
end

type    SALES = RECORD
        CUSTOMER_ID = integer;
        ITEM_ID = string;
        ITEM_QTY = integer;
        ITEM_PRICE = integer;
end

```

Generally, a schema can be partitioned into two categories, i.e., (i) **Logical schema** and (ii) **Physical schema**.

i) The logical schema is concerned with exploiting the data structures offered by the DBMS so that the schema becomes understandable to the computer. It is important as programs use it to construct applications.

ii) The physical schema is concerned with the manner in which the conceptual database get represented in the computer as a stored database. It is hidden behind the logical schema and can usually be modified without affecting the application programs.

The DBMS's provide DDL and DSDL to specify both the logical and physical schema.

Instances:

The data in the database at a particular moment of time is called an instance or a database state. In a given instance, each schema construct has its own current set of instances. Many instances or database states can be constructed to correspond to a particular database schema. Everytime we update (i.e., insert, delete or modify) the value of a data item in a record, one state of the database changes into another state.

The following figure shows an instance of the ITEM relation in a database schema.

ITEM		
ITEM-ID	ITEM_DESC	ITEM_COST
1111A	Nutt	3
1112A	Bolt	5
1113A	Belt	100
1144B	Screw	2

(I)What are the advantages and disadvantages of a DBMS ?

Ans:-DBMS: A database is a collection of non-redundant data which can be shared by different application systems stresses the importance of multiple applications, data sharing the spatial database

becomes a common resource for an agency implies separation of physical storage from use of the data by an application program, i.e. program/data independence the user or programmer or application specialist need not know the details of how the data are stored such details are "transparent to the user" changes can be made to data without affecting other components of the system. e.g. change format of data items (real to integer, arithmetic operations) change file structure (reorganize data internally or change mode of access) relocate from one device to another, e.g. from optical to magnetic storage, from tape to disk

Advantages:

Control of data redundancy.

Data consistency

More information from the same amount of data.

Sharing of data.

Improved data integrity.

Improved security.

Enforcement of standards.

8. Economy of scale.

1. Controlling Data Redundancy - In the conventional file processing system, Every user group maintains its own files for handling its data files. This may lead to

- Duplication of same data in different files.
- Wastage of storage space, since duplicated data is stored.
- Errors may be generated due to pupation of the same data in different files.
- Time in entering data again and again is wasted.
- Computer Resources are needlessly used.
- It is very difficult to combine information

2. Elimination of Inconsistency - In the file processing system information is duplicated throughout the system. So changes made in one file may be necessary be carried over to another file. This may lead to inconsistent data. So we need to remove this duplication of data in multiple file to eliminate inconsistency.

3 Better service to the users - A DBMS is often used to provide better services to the users. In conventional system, availability of information is often poor, since it normally difficult to obtain information that the existing systems were not designed for. Once several conventional

systems are combined to form one centralized database, the availability of information and its update ness is likely to improve since the data can now be shared and DBMS makes it easy to respond to anticipated information requests.

Centralizing the data in the database also means that user can obtain new and combined information easily that would have been impossible to obtain otherwise. Also use of DBMS should allow users that don't know programming to interact with the data more easily, unlike file processing system where the programmer may need to write new programs to meet every new demand.

4. Flexibility of the System is improved - Since changes are often necessary to the contents of the data stored in any system, these changes are made more easily in a centralized database than in a conventional system. Applications programs need not to be changed on changing the data in the database.

5. Integrity can be improved - Since data of the organization using database approach is centralized and would be used by a number of users at a time. It is essential to enforce integrity-constraints.

In the conventional systems because the data is duplicated in multiple files so updating or changes may sometimes lead to entry of incorrect data in some files where it exists.

6. Standards can be enforced - Since all access to the database must be through DBMS, so standards are easier to enforce. Standards may relate to the naming of data, format of data, structure of the data etc. Standardizing stored data formats is usually desirable for the purpose of data interchange or migration between systems.

7. Security can be improved - In conventional systems, applications are developed in an adhoc/temporary manner. Often different system of an organization would access different components of the operational data, in such an environment enforcing security can be quiet difficult. Setting up of a database makes it easier to enforce security restrictions since data is now centralized. It is easier to control who has access to what parts of the database. Different checks can be established for each type of access (retrieve, modify, delete etc.) to each piece of information in the database.

8. Organization's requirement can be identified - All organizations have sections and departments and each of these units often consider the work of their unit as the most important and therefore consider their need as the most important. Once a database has been setup with centralized control, it will be necessary to identify organization's requirement and to balance the needs of the competating units. So it may become necessary to ignore some requests for information if they conflict with higher priority need of the organization. It is the responsibility of the DBA (Database Administrator) to structure the database system to provide the overall service that is best for an organization.

9. Overall cost of developing and maintaining systems is lower - It is much easier to respond to unanticipated requests when data is centralized in a database than when it is stored in a conventional file system. Although the initial cost of setting up of a database can be large, one

normal expects the overall cost of setting up of a database, developing and maintaining application programs to be far lower than for similar service using conventional systems, Since the productivity of programmers can be higher in using non-procedural languages that have been developed with DBMS than using procedural languages.

10. Data Model must be developed - Perhaps the most important advantage of setting up of database system is the requirement that an overall data model for an organization be build. In conventional systems, it is more likely that files will be designed as per need of particular applications demand. The overall view is often not considered. Building an overall view of an organization's data is usual cost effective in the long terms.

11. Provides backup and Recovery - Centralizing a database provides the schemes such as recovery and backups from the failures including disk crash, power failures, software errors which may help the database to recover from the inconsistent state to the state that existed prior to the occurrence of the failure, though methods are very complex.

Disadvantages:

Complexity

Size

Cost of DBMS

Additional hardware costs

Cost of conversion

Performance

Higher impact of a failure

(J) Explain the following in detail ?

(1) Abstraction and data integration

(2) Metadata

Ans:- 1) Abstraction and data integration:

i) Data Abstraction

Major aim of a DBMS is to provide users with an abstract view of data

Hides certain details of how the data are stored & maintained DBMS must retrieve data efficiently Need for efficiency has led designers to use complex data structures to represent the data in the database Most DB users are not computer trained, developers hide complexity through several levels of abstraction to simplify user's interaction with the systems

Physical or Internal Level

Lowest level of abstraction describes how data are actually stored

Describes complex low-level data structures in detail

Logical or Conceptual Level

Describes what data are stored in the DB & what relationships exist among those data Describes the entire DB in terms of relatively simpler structures

View or External Level

Highest level of abstraction which describes only a part of the DB

User's view of the DB. This level describes that part of the DB that is relevant to each user

ii) Data Integration involves combining data residing in different sources and providing users with a unified view of these data. This process becomes significant in a variety of situations both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories, for example). Data integration appears with increasing frequency as the volume and the need to share existing data explodes.

The problem of providing

Uniform (sources transparent to users)

Access to (query)

Multiple (even 2 is a problem)

Autonomous (not affect the behavior of sources)

Heterogeneous (different data models, schemas)

Structured (at least semi structured)

Data sources (not only databases)

2)Metadata:-Metadata (meta data, or sometimes metainformation) is "data about data", of any sort in any media. An item of metadata may describe an individual datum, or content item, or a collection of data including multiple content items and hierarchical levels, for example a database schema. In data processing, metadata is definitional data that provides information about or documentation of other data managed within an application or environment

Metadata is structured data which describes the characteristics of a resource. It shares many similar characteristics to the cataloguing that takes place in libraries, museums and archives. The term "meta" derives from the Greek word denoting a nature of a higher order or more fundamental kind. A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values.

Each metadata schema will usually have the following characteristics:

a limited number of elements

the name of each element

the meaning of each element

(K)Define association between files. Name and discuss the type of association between files.
?

Ans:-files-A collection of related records. For example, a file might contain data about ROBCOR Company's vendors; or, a file might contain the records for the students currently enrolled at Gigantic University. a collection of pages, each containing a collection of records.
Must support:

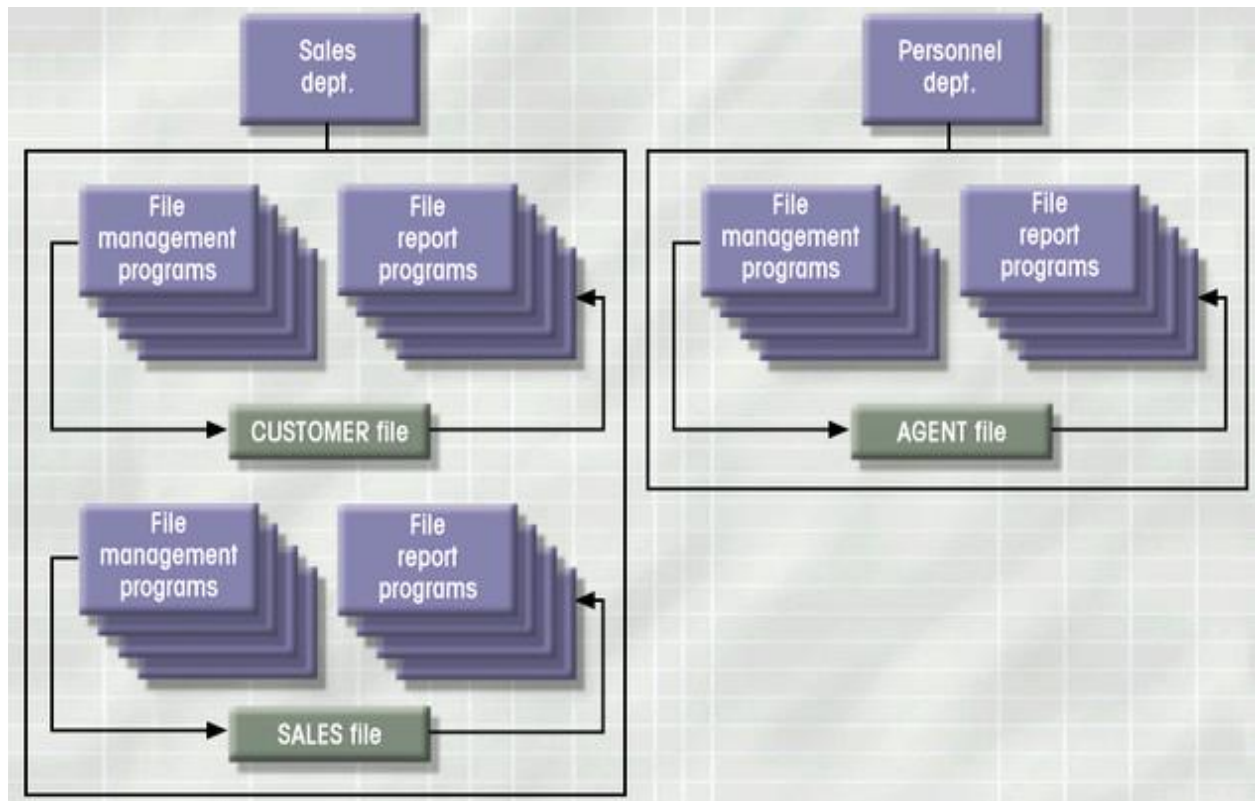
insert/delete/modify record

read a particular record (specified using *record id*)

scan all records (possibly with some conditions on the records to be retrieved)

Association between files-

1. One-to-one association (1:1 < ---- >)
2. One-to-many association (1:M < ----- >>)
3. Many-to-many association (M:M <<--- >>)
4. Many-to-one association (M:1 << ---->)
5. One-to-one conditional association (<----o--- >)



(d)What are the components of a Database Management System ?

Ans:- Database Management System:- A database is a collection of non-redundant data which can be shared by different application systems, A shared collection of logically related data *and its data description, defined once but used simultaneously by many applications and users*

- stresses the importance of multiple applications, data sharing
- the spatial database becomes a common resource for an agency

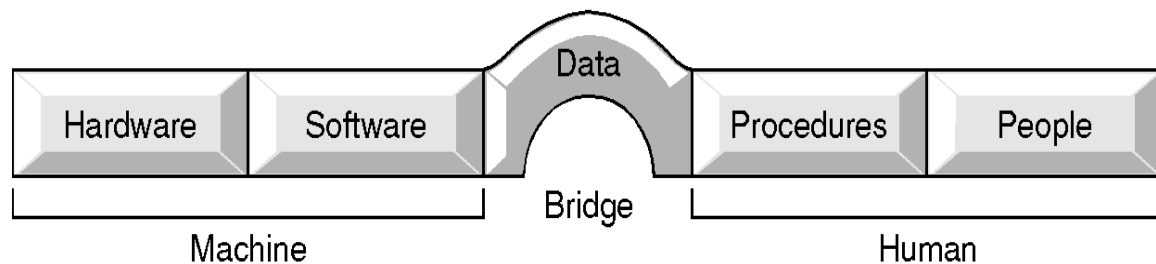


fig.Database Management System

- Hardware
 - Can range from a PC to a network of computers.
- Software
 - DBMS, operating system, network software (if necessary) and also the application programs.
- Data
 - Used by the organization and a description of this data called the schema.
- Procedures
 - Instructions and rules that should be applied to the design and use of the database and DBMS.
- People
 - It includes various types users.

Components of DBMS

DDL Compiler
Data Manager
File Manager
Disk Manager
Query Processor

Telecommunication System

Data Files

Data Dictionary

Access Aids

1. DDL Compiler - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. DML Compiler and Query optimizer - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. Data Manager - The Data Manager is the central software component of the DBMS also known as Database Control System.

The Main Functions Of Data Manager Are: –

Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.

- Controls DBMS information access that is stored on disk.
- It also controls handling buffers in main memory.
- It also enforces constraints to maintain consistency and integrity of the data.
- It also synchronizes the simultaneous operations performed by the concurrent users.
- It also controls the backup and recovery operations.

4. Data Dictionary - Data Dictionary is a repository of description of data in the database. It contains information about

24. Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.

25. Relationships between database transactions and data items referenced by them which is useful in determining which transactions are affected when certain data definitions are changed.

26. Constraints on data i.e. range of values permitted.

27. Detailed information on physical database design such as storage structure, access paths, files and record sizes.
28. Access Authorization - is the Description of database users their responsibilities and their access rights.
29. Usage statistics such as frequency of query and transactions.
30. Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.
31. Importance of Data Dictionary -
32. Data Dictionary is necessary in the databases due to following reasons:
33. It improves the control of DBA over the information system and user's understanding of use of the system.
34. • It helps in document ting the database design process by storing documentation of the result of every design phase and design decisions.
35. It helps in searching the views on the database definitions of those views.
36. It provides great assistance in producing a report of which data elements (i.e. data values) are used in all the programs.
37. It promotes data independence i.e. by addition or modifications of structures in the database application program are not effected.

5. Data Files - It contains the data portion of the database.

6. Compiled DML - The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

7. End Users : The users of the database system can be classified in the following groups, depending on their degree of expertise or the mode of their interactions with the DBMS.

1. Naïve users
2. Online Users
3. Application Programmers
4. Database administrator

i) Naïve User: Naive users who need not have aware of the present of the database system or any other system.. A user of an automatic teller falls under this category. The user is instructed through each step of a transaction; he or she responds by pressing a coded key or entering a numeric value. The operations that can be performed by this calls of users are very limited and affect a precise portion of the database; in case of the user of the automatic teller machine, only one or more of her or his own accounts.

Other such naive users are where the type and range of response is always indicated to the user. Thus, a very competent database designer could be allowed to use a particular database system only as a naive user.

ii) Online users: There are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the presence of the database system and may have acquired a certain amount of expertise in the limited interaction they are permitted with the database through the intermediate application program. The more sophisticated of these users may also use a data manipulation language to manipulate the database directly. On-line users can also be naive users requiring help such as menus.

iii) Application Users: Professional programmers who are responsible for developing application programs or user interfaces utilized by the naive and online users fall into this category. The application programs could be written in a general purpose programming language such as Assembler C, COBOL, FORTRAN, PASCAL, or PL/I and include the commands required to manipulate the database.

iv) Database Administrator: Centralized control of the database is exerted by a person or group of persons under the supervision of a high level administrator. This person or group is referred to as the database administrator (DBA). They are users who are the most familiar with the database and are responsible for creating, modifying and maintaining its three levels.

The DBA is the custodian of the data and controls the database structure. The DBA administers the three levels of the database and in consultation with the overall user community, sets up the definition of the global view or conceptual level of the database. The DBA further specifies the external view of the various users and applications and is responsible for definition and implementation of the internal level, including the storage structure and access methods to be used for the optimum performance of the DBMS.



Tulsiramji Gaikwad-Patil College of Engineering & Technology

Department of Information Technology

Subject Notes

Academic Session: 2018 – 2019

Subject: DBMS

Semester: I

UNIT: II

UNIT 2:

Unit -2

Data Models – Introduction, Data Association (Entities, Attributes, and Association, Relationship among entities, representation of association and relationships), Data Model Classification – (Approaches to the relational model, Hierarchical model, and Network Model with an example). Entity – Relationship model, **Concepts of file organization** – Sequential Files, index Sequential Files, Direct Files, Secondary Key retrieval.

(A) Explain ER model with suitable example.

Ans: It is a “top-down” approach

This data model allows us to describe how data is used in a real-world enterprise an iterative process A team-oriented process, with all business managers (or designates) involved should validate with a “bottom-up” approach Has three primary components: entity, relationship, attributes

Many notation methods, Chen was the first to become established.

The building blocks of E-R model are entities, relationships and attributes

Entity: An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of some domain. When we speak of an entity we normally speak of some aspect of

the real world which can be distinguished from other aspects of the real world. An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order.. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

Attributes: It is a Characteristic of an entity. Student's (entity) attributes: student ID, student name, address, etc

Attributes are of various types:

- Simple/Single Attributes
- Composite Attributes
- Multivalued attributes
- Derived attributes

Relationship: Relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an *owns* relationship between a company and a computer, a *supervises* relationship between an employee and a department, a *performs* relationship between an artist and a song, a *proved* relationship between a mathematician and a theorem. Relationships are represented as diamonds, connected by lines to each of the entities in the relationship. Types of relationships are as follows:

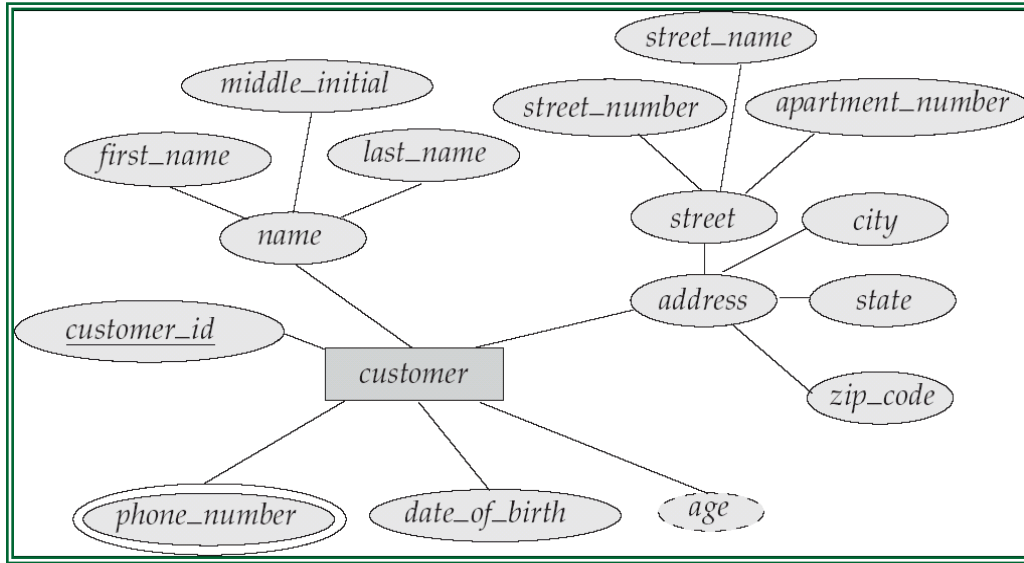
- One to many 1<----- M
- Many to one M----->1
- Many to many M-----M

Symbols and their meanings

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
- Double ellipses represent Multivalued attributes.
- Dashed ellipses denote derived attributes.
- Underline indicates primary key attributes

Example:

Given Entity Customer with attributes: customer_id(primary key), name(first_name, last_name, middle_name), phone_number, date_of_birth, address(city,state,zip_code,street), Street(Street_name,street_number,apartment_number)



B) Differentiate between sequential file and Indexed sequential files

SEQUENTIAL FILES	INDEXED FILES	RELATIVE FILES
Sequential files are called as QSAM files.	Indexed files are also called as VSAM files.	Relative called as VSAM files (RRDS).
Data is entered in entry sequential order.	Data is entered in key sequential order.	Data is entered in RRN number.
Duplicate data is allowed.	Duplicate data is not allowed.	Duplicate data is not allowed
Data is not in sorted order.	Data is in sorted order based on key.	Date is in sorted order based on RRN.
Delete is not applicable.	Delete is applicable.	Delete is applicable.
Access is slow.	Access is faster.	Access is faster than indexed files.
Key is not available.	Key is available. Key is user defined. Key is part of a record.	Key is available. Key is system defined. key is outside of a record.
	Data is stored on disk only.	Data is stored on disk only.

C) Illustrate the construction of secondary key retrieval with a suitable example .

Ans: In sequential File, Index Sequential file and Direct File we have considered the retrieval and update of data based on primary key.

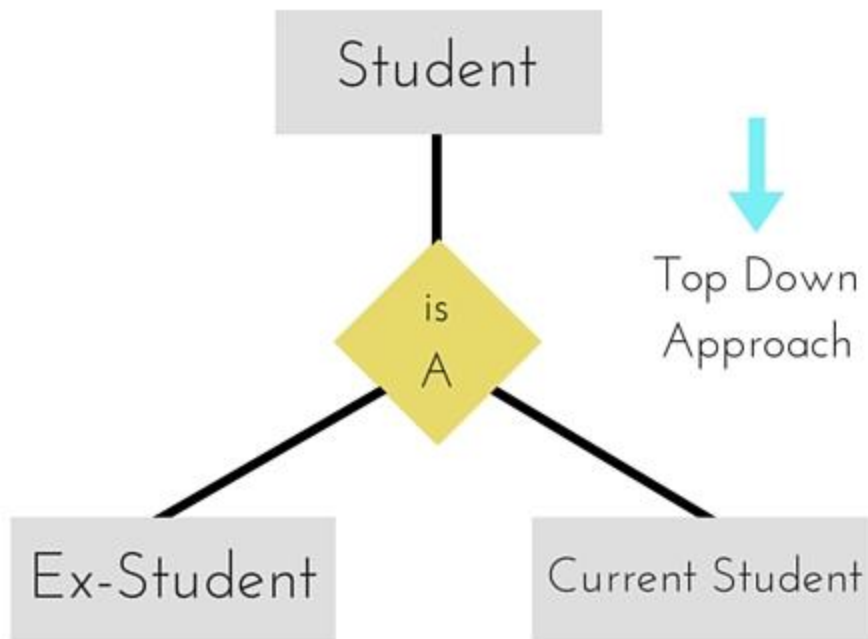
(i) We can retrieve and update data based on secondary key, called as secondary key retrieval.

(ii) In secondary key retrieval , there are multiple records satisfying a given key value.

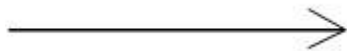
(iii) For e.g. if we search a student file based on the attribute “stud_name”, we can get the set of records which satisfy the given value.

(D) Define the following terms :-

Ans: Specialization : **pecialization** is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.



Association : Association is a relationship between two objects. In other words, association defines the multiplicity between objects. You may be aware of one-to-one, one-to-many, many-to-one, many-to-many all these words define an association between objects. Aggregation is a special form of association. Composition is a special form of aggregation.



Example: A Student and a Faculty are having an association.

(i)

Relationship : The **relational model (RM)** for [database](#) management is an approach to managing data using a structure and language consistent with [first-order predicate logic](#), first described in

1969 by [Edgar F. Codd](#),^{[1][2]} where all data is represented in terms of [tuples](#), grouped into [relations](#). A database organized in terms of the relational model is a [relational database](#).

Relational Model

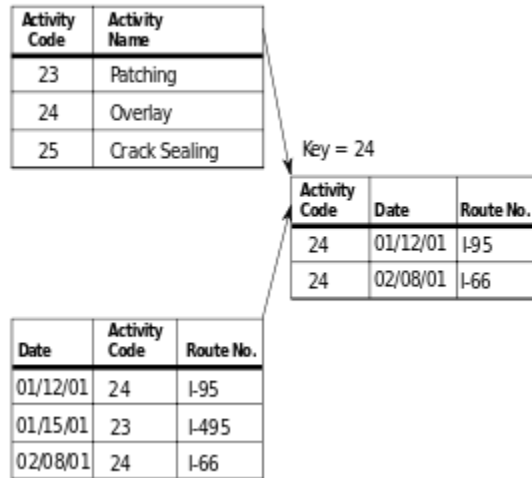
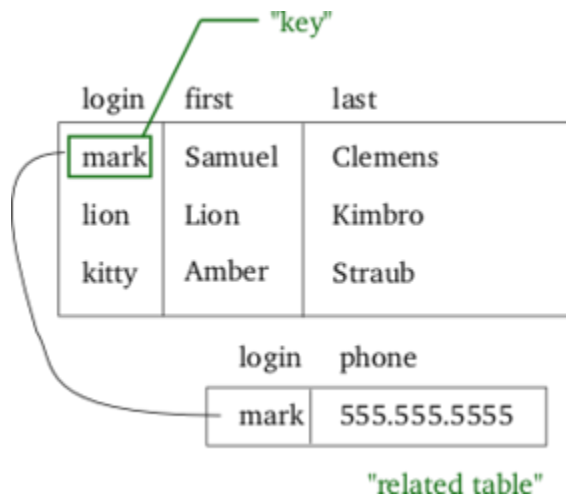


Diagram of an example database according to the relational model^[3]



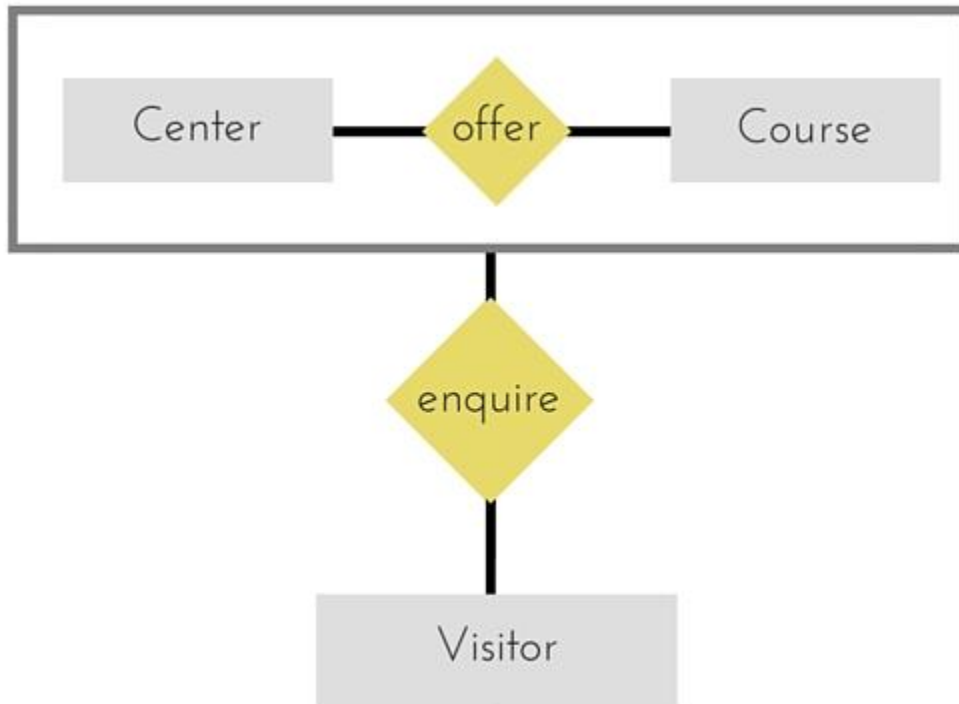
In the relational model, related records are linked together with a "key".

The purpose of the relational model is to provide a [declarative](#) method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

Most relational databases use the [SQL](#) data definition and query language; these systems implement what can be regarded as an engineering approximation to the relational model. A *table* in an SQL database schema corresponds to a predicate variable; the contents of a table to a

relation; key constraints, other constraints, and SQL queries correspond to predicates. However, SQL databases [deviate from the relational model in many details](#), and Codd fiercely argued against deviations that compromise the original principles.

Aggregation : Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.



(D) Define entity set. What is the difference between strong and weak entity set? Explain the technique used to convert any weak entity set into a strong entity set

Ans:- Entity:- Anything (such as a person, place, thing, or event) about which data are to be collected and stored

Could be physical objects (customer, product, student, etc.) or abstraction (enrollment, flight route, order, etc.)

Entity set = collection of similar entities.

Similar to a class in object-oriented languages.

Attribute = property of (the entities of) an entity set.

Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

Example:-

Entity : Employee

Attributes: Emp_id, Ename, Salary, Dept, Desig

A file is a collection of identical record type occurrence pertaining to an entity set and is labeled to identify the entity set.

Emp_id	Ename	Salary	Dept	Desig
101	Sam	10000	Sales	Manager
102	Allen	15000	Accounting	Project Leader

Difference Between Strong and weak entity set

Strong Entity :- We use a column for each attribute of the set. Each row in the table corresponds to one entity of the entity set. For the entity set Account, in table We can add, delete and modify row. A row of a table will be an n-tuple where n is the number of attributes. Actually, the table contains a subset of the set of all possible row. We refer to the set of all possible row as the Cartesian product of the sets of all attribute values. We may denote this as

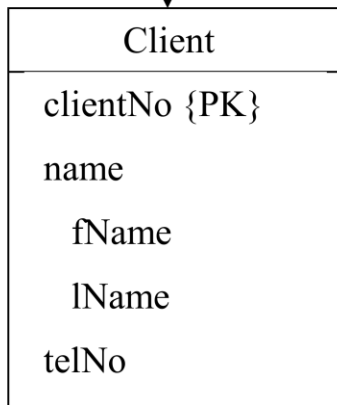
$$D_1 \times D_2 \text{ or } \times_{i=1}^n D_i$$

For the account table, where D_1 and D_2 denote the set of all account number and all account balances, respectively. In general, for a table of n columns, we may denote the Cartesian of D_1, D_2, \dots, D_n by

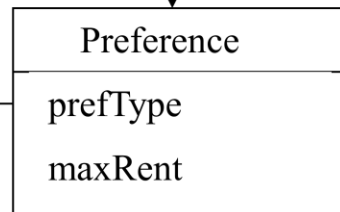
$$x_{i=1}^n D_i$$

Weak Entity:- For a weak entity set, we add columns to the table corresponding to the primary key of the strong entity set on which the weak set is dependent. For example, the weak entity set Transaction has three attributes: transaction#, date and Money. The primary key of Account is account#.

STRONG ENTITY



WEAK ENTITY



States →

(E) Explain Different types of Data Model with example?

Ans:-Data Model: A data model is a mechanism that provides this abstraction for database applications. Data modeling is used for representing entities of interest and their relationships in the database. It allows the conceptualization of the association between various entities and their attributes

From user’s viewpoint a DB is a collection of data that models a certain portion of the reality

The logic abstraction used to present data to the user defines a **data model**, more precisely:

A data model is a collection of concepts used to describe data, their associations and the constraints that these data have to satisfy

Types of Data Model:

Data Models can be classified as

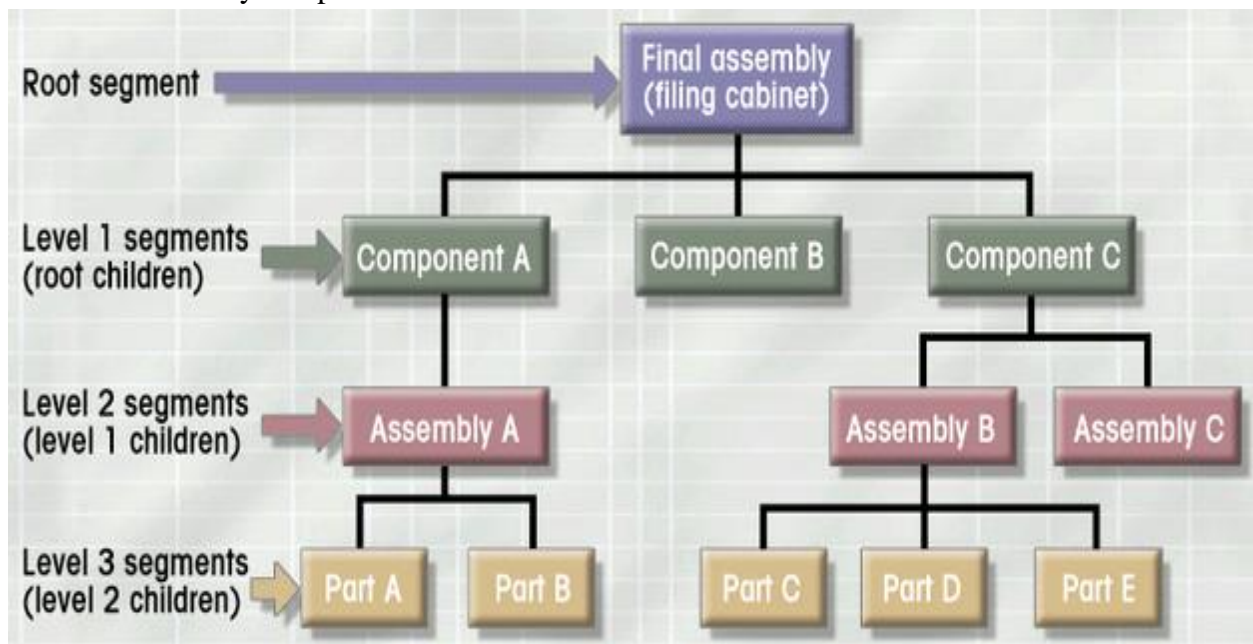
- File based Systems or Primitive Models
- Traditional data models
 - Hierarchical data model
 - Network data model
 - Relational data model
- Semantic data models

Hierarchical Database Model

Logically represented by an upside down tree

Each parent can have many children

Each child has only one parent



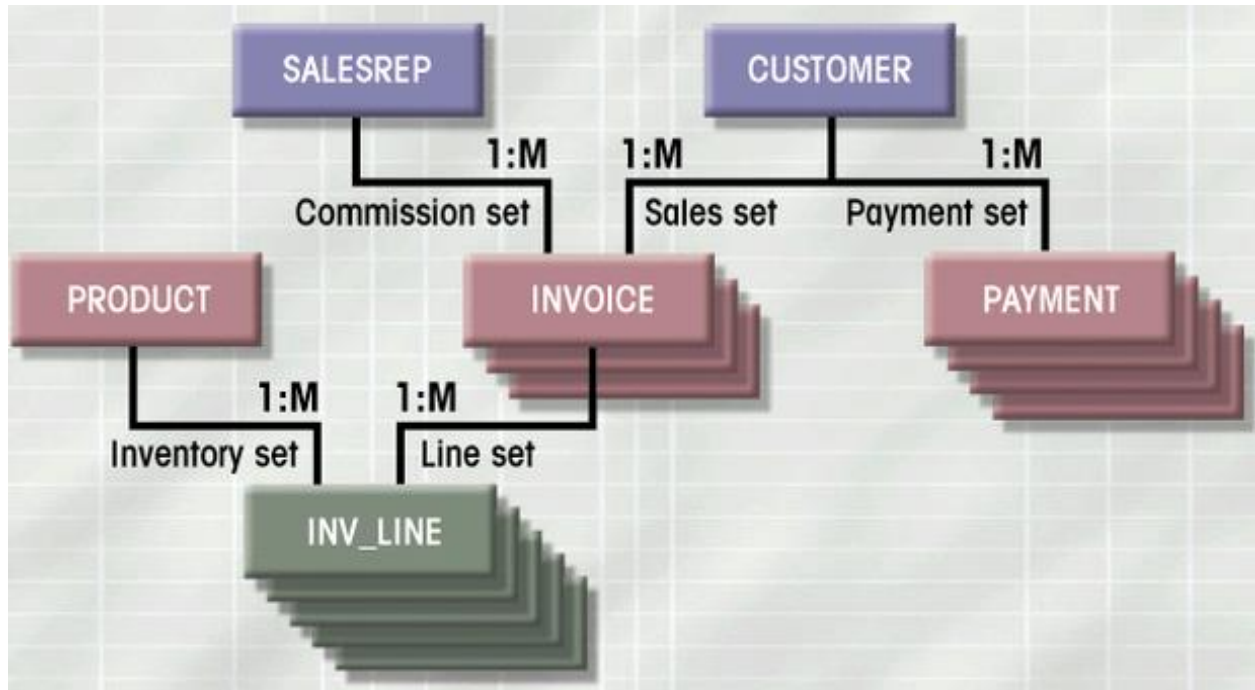
Network Database Model

Each record can have multiple parents

Composed of sets

Each set has owner record and member record

Member may have several owners



Retational Database Model:-

Perceived by user as a collection of tables for data storage

Tables are a series of row/column intersections

Tables related by sharing common entity characteristics

Example:-

Table name: AGENT

	AGENT_CODE	AGENT_LIAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
▶	501	Alby	Alex	B	713	228-1249
	502	Hahn	Leah	F	615	882-1244
	503	Okon	John	T	615	123-5589

Link through AGENT code

Table name: CUSTOMER

	CUS_CODE	CUS_LIAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
▶	10010	Ramas	Alfred	A	615	844-2573	05-Apr-2002	502
	10011	Dunne	Leona	K	713	894-1238	16-Jun-2002	501
	10012	Smith	Kathy	vV	615	894-2285	29-Jan-2001	502
	10013	Olowski	Paul	F	615	894-2180	14-Oct-2002	502
	10014	Orlando	Myron		615	222-1672	28-Dec-2002	501
	10015	O'Brian	Amy	B	713	442-3381	22-Sep-2002	503
	10016	Brown	James	G	615	297-1228	25-Mar-2002	502
	10017	Williams	George		615	290-2556	17-Jul-2002	503
	10018	Farriss	Anne	G	713	382-7185	03-Dec-2002	501
	10019	Smith	Olette	K	615	297-3809	14-Mar-2002	503

(F) Explain the different types of file organizations

Ans:- Types of file Organisation

- 1) **Sequential File**
- Index-Sequential Files**
- Direct Files,**
- Secondary Key retrieval.**

Sequential File Organization

1. A **sequential file** is designed for efficient processing of records in **sorted order** on some **search key**.
 - Records are chained together by pointers to permit fast retrieval in search key order.
 - Pointer points to next record in order.
 - Records are stored physically in search key order (or as close to this as possible).
 - This minimizes number of block accesses.
 - Figure 10.15 shows an example, with *bname* as the search key.
2. It is difficult to maintain physical sequential order as records are inserted and deleted.
 - Deletion can be managed with the pointer chains.
 - Insertion poses problems if no space where new record should go.
 - If space, use it, else put new record in an **overflow block**.
 - Adjust pointers accordingly.
 - Figure 10.16 shows the previous example after an insertion.
 - Problem: we now have some records out of physical sequential order.
 - If very few records in overflow blocks, this will work well.
 - If order is lost, reorganize the file.
 - Reorganizations are expensive and done when system load is low.
3. If insertions rarely occur, we could keep the file in physically sorted order and reorganize when insertion occurs. In this case, the pointer fields are no longer required.

The Sequential File

Fixed format used for records

Records are the same length

All fields the same (order and length)

Field names and lengths are attributes of the file

One field is the key field

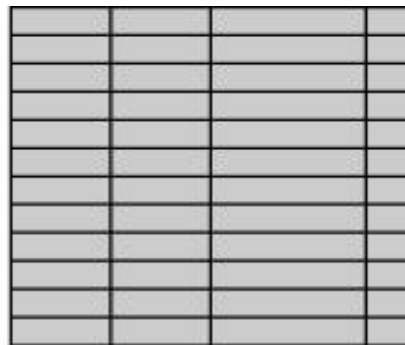
Uniquely identifies the record

Records are stored in key sequence

The Sequential File

New records are placed in a log file or transaction file

Batch update is performed to merge the log file with the master file



Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

Figure 12.3 Common File Organizations

ii) Index-Sequential Files

Ans: Indexed Sequential File

Index provides a lookup capability to quickly reach the vicinity of the desired record, Contains key field and a pointer to the main file, Indexed is searched to find highest key value that is equal or less than the desired key value, Search continues in the main file at the location indicated by the pointer

Indexed Sequential File

New records are added to an overflow file

Record in main file that precedes it is updated to contain a pointer to the new record. The overflow is merged with the main file during a batch update. Multiple indexes for the same key field can be set up to increase efficiency

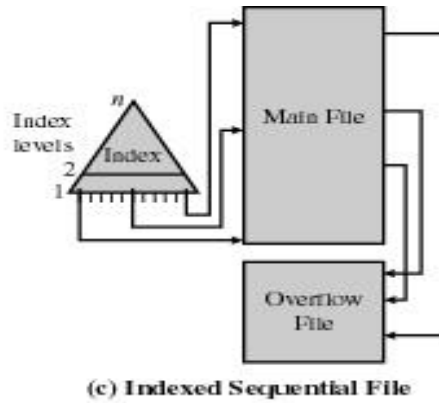
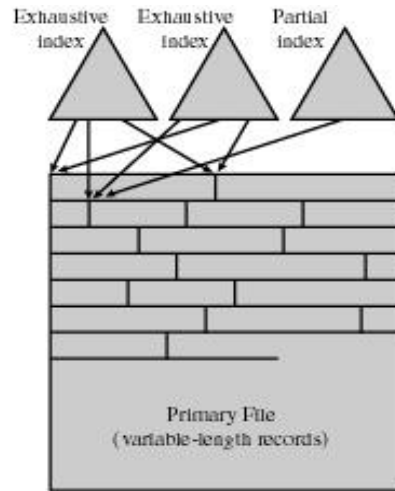


Figure 12.3 Common File Organizations

Indexed File Uses multiple indexes for different key fields

May contain an exhaustive index that contains one entry for every record in the main file. May contain a partial index



(d) Indexed File

Figure 12.3 Common File Organizations

Direct Files:- This organization allows only direct method to store and access records on the disk. The shortcoming of index schema is that an index must be accessed and read to find records. This, in itself, can become a bottleneck. Since two trips are needed to disk to access a record, one to read the index and another to access the record in a file. Maintaining a good index is an issue, it is addressed very well in B-tree organization

This direct file organization is used in reservation system for railways, airlines, hotels and car rentals where very high response time is required. Data is organized hierarchically on a disk, which is schematically shown below.

Cylinder index:

Cylinder	Highest Key Value
1	70
2	90
3	110

Track index:

Cylinder 1

Cylinder 2

Cylinder 2

Track Highest Key Value
value

1	20
2	50
3	70

Track Highest Key Value

1	75
2	80
3	95

Track Highest Key

1	98
2	105
3	110

Secondary Key retrieval.

G)What are the different types of data model ? Explain E-R models in detail.

Ans: Data model : A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. To use a common analogy, the data model is equivalent to an architect's building plans.

A data model is independent of hardware or software constraints. Rather than try to represent the data as a database would see it, the data model focuses on representing the data as the user sees it in the "real world". It serves as a bridge between the concepts that make up real-world events and processes and the physical representation of those concepts in a databas

High-level (conceptual) data models:

- use concepts such as entities, attributes, relationships
- object-based models: ER model, OO model

Representational (implementation) data models:

- most frequently used in commercial DBMSs
- record-based models: relational, hierarchical, network

Low-level (physical) data models:

- to describe the details of how data is stored
- captures aspects of database system implementation: record structures (fixed/variable length) and ordering, access paths (key indexing), etc.

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. To use a common analogy, the data model is equivalent to an architect's building plans.

A data model is independent of hardware or software constraints. Rather than try to represent the data as a database would see it, the data model focuses on representing the data as the user sees it in the "real world". It serves as a bridge between the concepts that make up real-world events and processes and the physical representation of those concepts in a database.

H) Explain the following terms:-

- (i) Identifying relationship

- An **identifying relationship** is when the existence of a row in a child table depends on a row in a parent table. This may be confusing because it's common practice these days to create a pseudokey for a child table, but *not* make the foreign key to the parent part of the child's primary key. Formally, the "right" way to do this is to make the foreign key part of the child's primary key. But the logical relationship is that the child cannot exist without the parent.

Example: A `Person` has one or more phone numbers. If they had just one phone number, we could simply store it in a column of `Person`. Since we want to support multiple phone numbers, we make a second table `PhoneNumbers`, whose primary key includes the `person_id` referencing the `Person` table.

We may think of the phone number(s) as belonging to a person, even though they are modeled as attributes of a separate table. This is a strong clue that this is an identifying relationship (even if we don't literally include `person_id` in the primary key of `PhoneNumbers`).

(ii) Discriminator : In [distributed computing](#), a **discriminator** is a typed tag field present in [OMG IDL discriminated union](#) type and value definitions that determines which union member is selected in the current union instance.^{[11][2]} Unlike in some conventional programming languages offering support for unions, discriminator in IDL is not identical to selected field name. Here is an example of IDL union type definition:

```
union Register switch (char)
{
  case 'a':
  case 'b': short AX;

  case 'c': long EAX;

  default: octet AL;
};
```

Effective value of the *Register* type may contain AX as selected field, but discriminator value may be either 'a' or 'b' and it is stored in memory separately. Therefore, IDL logically separates information about currently selected field name and union effective value from information about current discriminator value. In the example above, discriminator value may be anything of the following: 'a', 'b', 'c', as well as all other characters belonging to the IDL `char` type, since the `default` branch specified in the example *Register* type allows use of the remaining characters as well.

(iii) Repeating group : Any attribute that can have multiple values associated with a single instance of some entity. For example, a book might have multiple authors. Such a "-to-many" relationship might be represented in an unnormalised relational database as multiple author columns in the book table or a single author(s) column containing a string which was a list of authors. Converting this to "first normal form" is the first step in database normalisation. Each author of the book would appear in a separate row along with the book's primary key. Later normalisation stages would move the book-author relationship into a separate table to avoid repeating other book attributes (e.g. title, publisher) for each author.

(iv) Binary relationship : A binary relationship defines the relation between two entities. They can be:
 one-to-one: for each value in the first entity, there is one value in the second entity
 one-to-many: for each value in the first entity, there are multiple values in the second entity
 many-to-many: there can be many values in the first entity for many values in the second entity

Basically, these are descriptions of how you're going to define your relationships and structures within a relational storage database. One-to-one and one-to-many are usually defined by a foreign key. Many-to-many requires a mapping or interim table to allow for the multiple connections.

(I) Explain Index sequential file.

Indexed Sequential Files:
 Each record of a file has a key field which uniquely identifies that record. An index consists of keys and addresses (physical disc locations). An indexed sequential file is a sequential file (i.e. sorted into order of a key field) which has an index. A full index to a file is one in which there is an entry for every record. Indexed sequential files are important for applications where data needs to be accessed.....

- sequentially
- randomly using the index.

An indexed sequential file allows fast access to a specific record. Example: A company may store details about its employees as an indexed sequential file. Sometimes the file is accessed....

- sequentially. For example when the whole of the file is processed to produce payslips at the end of the month.

• randomly. Maybe an employee changes address, or a female employee gets married and changes her surname. An indexed sequential file can only be stored on a random access device eg magnetic disc, CD.

(J) Explain in detail Direct file organization with extendable hashing.

Ans: A bucket in a hash file is unit of storage (typically a disk block) that can hold one or more records.

The hash function, h , is a function from the set of all search-keys, K , to the set of all bucket addresses, B .

Insertion, deletion, and lookup are done in constant time.

Direct file organization provides the fastest direct access to records. When using direct access methods, records do not have to be arranged in any particular sequence on storage media. Characteristics of the direct access method include:

1. Computers must keep track of the storage location of each record using a variety of direct organization methods so that data can be retrieved when needed.
2. New transactions' data do not have to be sorted.
3. Processing that requires immediate responses or updating is easily performed.

Direct files are organized so as to facilitate access to records and to ensure their efficient storage. A tradeoff between these two requirements generally exists: if rapid access is required, more storage is required to make it possible.

Extendible hashing: is a type of [hash](#) system which treats a hash as a bit string, and uses a [trie](#) for bucket lookup.^[1] Because of the hierarchical nature of the system, re-hashing is an incremental operation (done one bucket at a time, as needed). This means that time-sensitive applications are less affected by table growth than by standard full-table rehashes.

- ✦ The efficiency of static hashing declines as the file grows.
 - ✦ Overflow pages increase search time.
 - ✦ One solution would be to use a range of hash functions based on a bit value and double the number of buckets (and the function range) whenever an overflow page is needed.

- ✦ Such a reorganization is expensive.
- ✦ Is it possible to make local changes?
- ✦ Use a directory of pointers to buckets.
 - ✦ Double the directory size when required.
 - ✦ Only split the pages that have overflowed.
 - ✦ The directory need only consist of an array of pointers to pages so is relatively compact.
 - ✦ The array index represents the value computed by the hash function.
 - ✦ At any time the array size is determined by how many bits of the hash result are being used.
 - ✦ Usually the last d (least significant) bits are used.

(K) Define entity set. What is the difference between strong and weak entity set? Explain the technique used to convert any weak entity set into a strong entity set

Ans:- Entity:- Anything (such as a person, place, thing, or event) about which data are to be collected and stored

Could be physical objects (customer, product, student, etc.) or abstraction (enrollment, flight route, order, etc.)

Entity set = collection of similar entities.

Similar to a class in object-oriented languages.

Attribute = property of (the entities of) an entity set.

Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

Example:-

Entity : Employee

Attributes: Emp_id, Ename, Salary, Dept, Desig

A file is a collection of identical record type occurrence pertaining to an entity set and is labeled to identify the entity set.

Emp_id	Ename	Salary	Dept	Desig
101	Sam	10000	Sales	Manager
102	Allen	15000	Accounting	Project Leader

Difference Between Strong and weak entity set

Strong Entity :- We use a column for each attribute of the set. Each row in the table corresponds to one entity of the entity set. For the entity set Account, in table We can add, delete and modify row. A row of a table will be an n-tuple where n is the number of attributes. Actually, the table contains a subset of the set of all possible row. We refer to the set of all possible row as the Cartesian product of the sets of all attribute values. We may denote this as

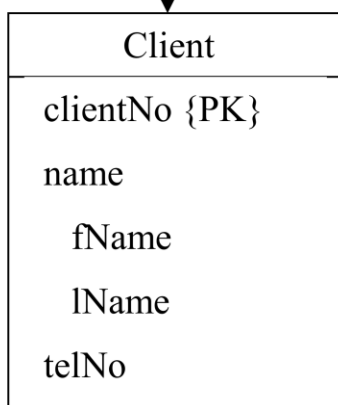
$$D_1 \times D_2 \text{ or } \prod_{i=1}^n D_i$$

For the account table, where D_1 and D_2 denote the set of all account number and all account balances, respectively. In general, for a table of n columns, we may denote the Cartesian of D_1, D_2, \dots, D_n by

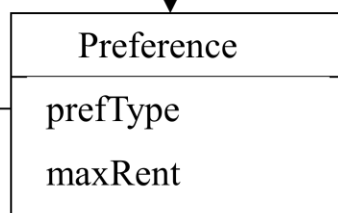
$$\prod_{i=1}^n D_i$$

Weak Entity:- For a weak entity set, we add columns to the table corresponding to the primary key of the strong entity set on which the weak set is dependent. For example, the weak entity set Transaction has three attributes: transaction#, date and Money. The primary key of Account is account#.

STRONG ENTITY



WEAK ENTITY



States →

(L) Explain Different types of Data Model with example?

Ans:-Data Model: A data model is a mechanism that provides this abstraction for database applications. Data modeling is used for representing entities of interest and their relationships in the database. It allows the conceptualization of the association between various entities and their attributes

From user’s viewpoint a DB is a collection of data that models a certain portion of the reality

The logic abstraction used to present data to the user defines a **data model**, more precisely:

A data model is a collection of concepts used to describe data, their associations and the constraints that these data have to satisfy

Types of Data Model:

Data Models can be classified as

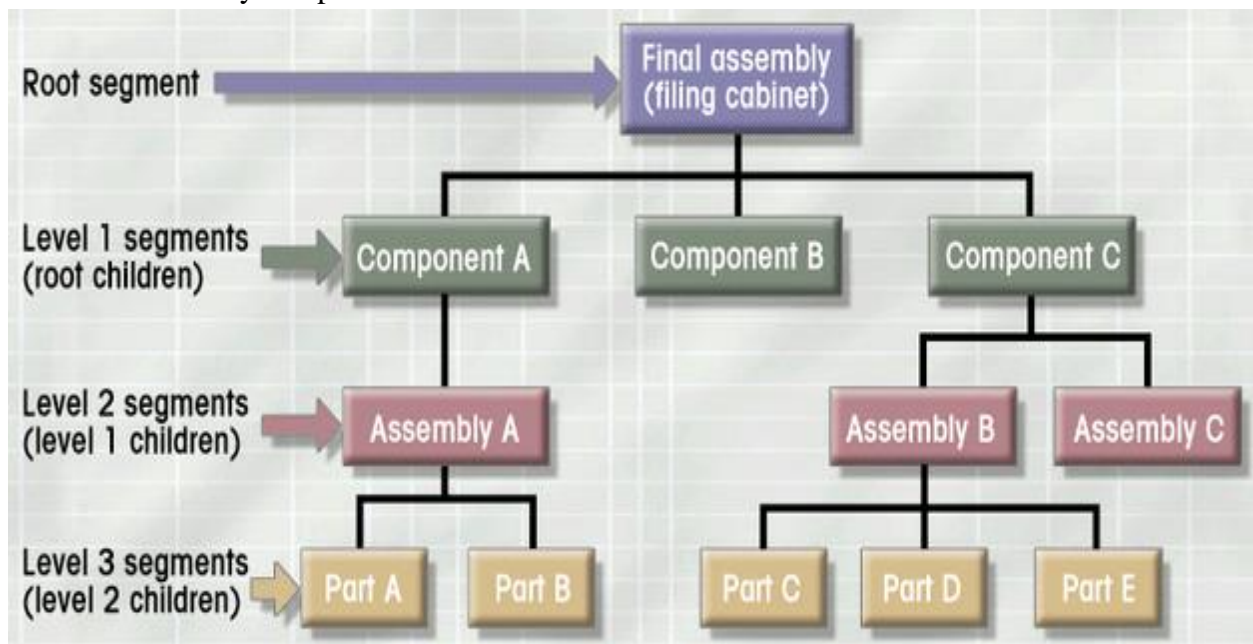
- File based Systems or Primitive Models
- Traditional data models
 - Hierarchical data model
 - Network data model
 - Relational data model
- Semantic data models

Hierarchical Database Model

Logically represented by an upside down tree

Each parent can have many children

Each child has only one parent



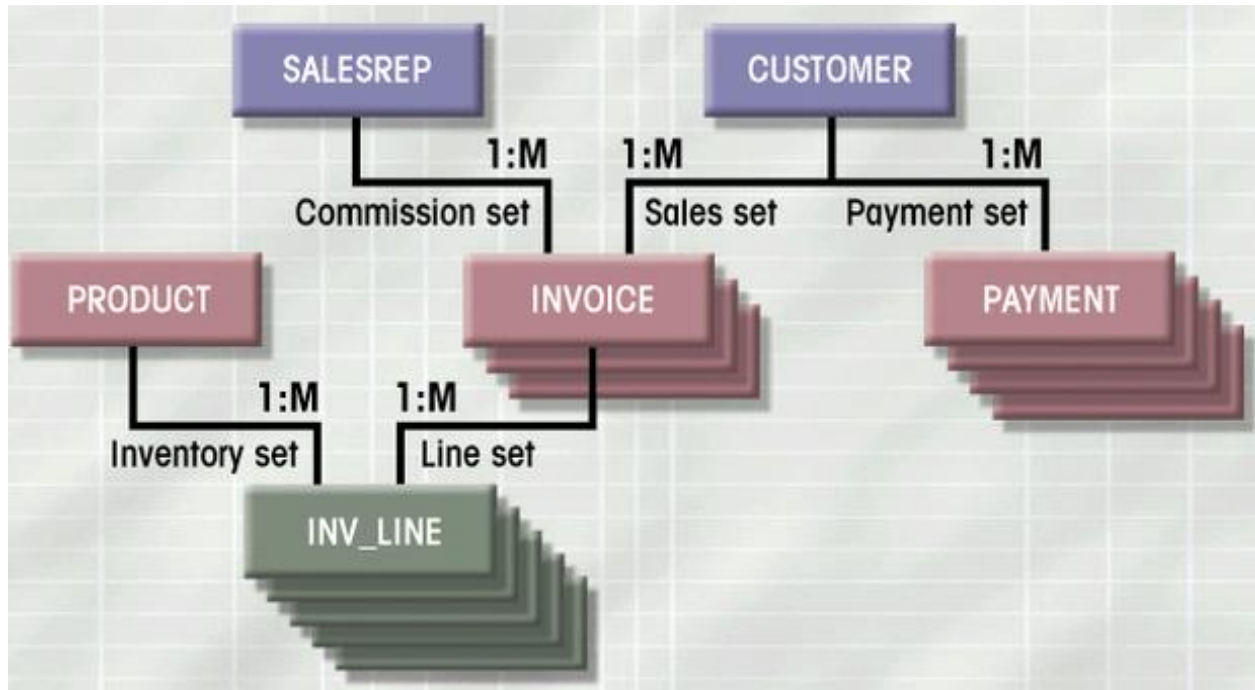
Network Database Model

Each record can have multiple parents

Composed of sets

Each set has owner record and member record

Member may have several owners



Retational Database Model:-

Perceived by user as a collection of tables for data storage

Tables are a series of row/column intersections

Tables related by sharing common entity characteristics

Example:-

Table name: AGENT

	AGENT_CODE	AGENT_LIAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
▶	501	Alby	Alex	B	713	228-1249
	502	Hahn	Leah	F	615	882-1244
	503	Okon	John	T	615	123-5589

Link through AGENT code

Table name: CUSTOMER

	CUS_CODE	CUS_LIAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
▶	10010	Ramas	Alfred	A	615	844-2573	05-Apr-2002	502
	10011	Dunne	Leona	K	713	894-1238	16-Jun-2002	501
	10012	Smith	Kathy	vV	615	894-2285	29-Jan-2001	502
	10013	Olowski	Paul	F	615	894-2180	14-Oct-2002	502
	10014	Orlando	Myron		615	222-1672	28-Dec-2002	501
	10015	O'Brian	Amy	B	713	442-3381	22-Sep-2002	503
	10016	Brown	James	G	615	297-1228	25-Mar-2002	502
	10017	Williams	George		615	290-2556	17-Jul-2002	503
	10018	Farriss	Anne	G	713	382-7185	03-Dec-2002	501
	10019	Smith	Olette	K	615	297-3809	14-Mar-2002	503

(M) Explain the different types of file organizations

Ans:-Types of file Organisation

Sequential File

Index-Sequential Files

Direct Files,

Secondary Key retrieval.

Sequential File Organization

1. A **sequential file** is designed for efficient processing of records in **sorted order** on some **search key**.
 - Records are chained together by pointers to permit fast retrieval in search key order.
 - Pointer points to next record in order.
 - Records are stored physically in search key order (or as close to this as possible).
 - This minimizes number of block accesses.
 - Figure 10.15 shows an example, with *bname* as the search key.
2. It is difficult to maintain physical sequential order as records are inserted and deleted.
 - Deletion can be managed with the pointer chains.
 - Insertion poses problems if no space where new record should go.
 - If space, use it, else put new record in an **overflow block**.
 - Adjust pointers accordingly.
 - Figure 10.16 shows the previous example after an insertion.
 - Problem: we now have some records out of physical sequential order.
 - If very few records in overflow blocks, this will work well.
 - If order is lost, reorganize the file.
 - Reorganizations are expensive and done when system load is low.
3. If insertions rarely occur, we could keep the file in physically sorted order and reorganize when insertion occurs. In this case, the pointer fields are no longer required.

The Sequential File

Fixed format used for records

Records are the same length

All fields the same (order and length)

Field names and lengths are attributes of the file

One field is the key field

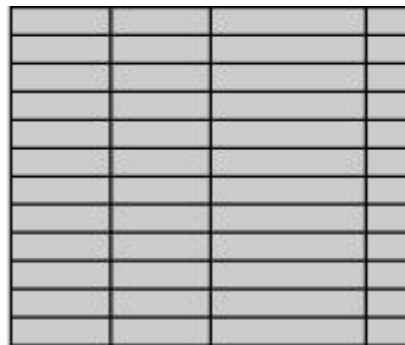
Uniquely identifies the record

Records are stored in key sequence

The Sequential File

New records are placed in a log file or transaction file

Batch update is performed to merge the log file with the master file



Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

Figure 12.3 Common File Organizations

ii) Index-Sequential Files

Ans: Indexed Sequential File

Index provides a lookup capability to quickly reach the vicinity of the desired record, Contains key field and a pointer to the main file, Indexed is searched to find highest key value that is equal or less than the desired key value, Search continues in the main file at the location indicated by the pointer

Indexed Sequential File

New records are added to an overflow file

Record in main file that precedes it is updated to contain a pointer to the new record. The overflow is merged with the main file during a batch update. Multiple indexes for the same key field can be set up to increase efficiency

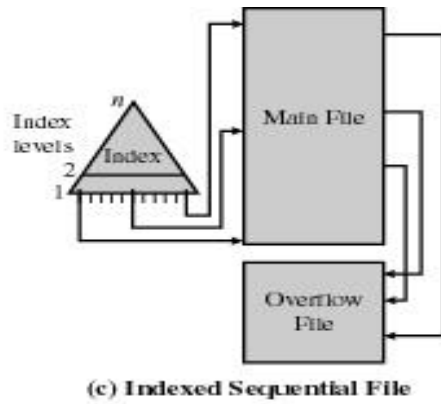


Figure 12.3 Common File Organizations

Indexed File Uses multiple indexes for different key fields

May contain an exhaustive index that contains one entry for every record in the main file. May contain a partial index

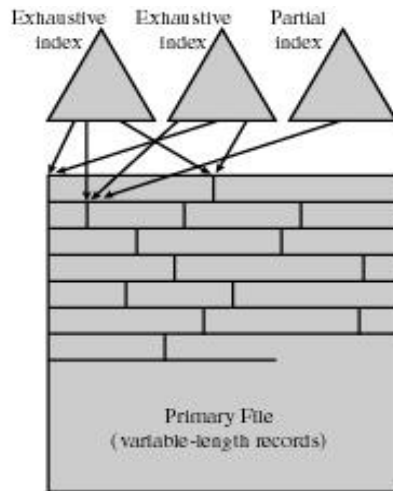


Figure 12.3 Common File Organizations

Direct Files:- This organization allows only direct method to store and access records on the disk. The shortcoming of index schema is that an index must be accessed and read to find records. This, in itself, can become a bottleneck. Since two trips are needed to disk to access a record, one to read the index and another to access the record in a file. Maintaining a good index is an issue, it is addressed very well in B-tree organization

This direct file organization is used in reservation system for railways, airlines, hotels and car rentals where very high response time is required. Data is organized hierarchically on a disk, which is schematically shown below.

Cylinder index:

Cylinder	Highest Key Value
1	70
2	90
3	110

Track index:

Cylinder 1		Cylinder 2		Cylinder 2	
Track	Highest Key Value	Track	Highest Key Value	Track	Highest Key
1	20	1	75	1	98
2	50	2	80	2	105
3	70	3	95	3	110

Secondary Key retrieval.

Que No.2.d)

(d)Illustrate the Construction of secondary key retrieval with a suitable example.

Ans:-



Tulsiramji Gaikwad-Patil College of Engineering & Technology

Department of Information Technology

Subject Notes

Academic Session: 2018 – 2019

Subject: DBMS

Semester: I

UNIT: III

UNIT 3: Syllabus

Unit -3

The Relational Model –Introduction, **Relationla Database** : Attributes and domains, tuples, Relations and their schemas, relational representation, Keys, relationship, relational operations, Integrity rules. **Relational Algebra** – Basic operations, Relational Algebra queries, Relational Calculus: tuple calculus, domain Calculus. **Relational Database Manipulations** : Introduction, SQL, Data Manipulation in SQL, Quel, Data Manipulations in Quel, QBE , Data Manipulations in QBE.

A) Define the following :-

(i) Domain

Definition: The domain of a database attribute is the set of all allowable values that attribute may assume.

Examples:

A field for gender may have the domain {male, female, unknown} where those three values are the only permitted entries in that column.

In [data management](#) and [database](#) analysis, a **data domain** refers to all the unique values which a [data element](#) may contain. The rule for determining the domain boundary may be as simple as a [data type](#) with an [enumerated](#) list of values.^[1]

For example, a [database table](#) that has information about people, with one record per person, might have a "[gender](#)" [column](#). This gender column might be declared as a [string data type](#), and allowed to have one of two known [code values](#): "M" for male, "F" for female—and [NULL](#) for

records where gender is unknown or not applicable (or arguably "U" for unknown as a [sentinel value](#)). The data domain for the gender column is: "M", "F".

In a [normalized data model](#), the [reference domain](#) is typically specified in a [reference table](#). Following the previous example, a Gender reference table would have exactly two records, one per allowed value—excluding NULL. Reference tables are formally related to other tables in a database by the use of [foreign keys](#).

Less simple domain boundary rules, if database-enforced, may be implemented through a [check constraint](#) or, in more complex cases, in a [database trigger](#). For example, a column requiring positive numeric values may have a check constraint declaring that the values must be greater than zero.

This definition combines the concepts of domain as an area over which control is exercised and the mathematical idea of a [set](#) of values of an [independent variable](#) for which a [function](#) is defined.

(ii) Degree and cardinality

The **degree of relationship** (also known as cardinality) is the number of occurrences in one entity which are associated (or linked) to the number of occurrences in another.

There are three degrees of relationship, known as:

1. one-to-one (1:1)
2. one-to-many (1:M)
3. many-to-many (M:N)

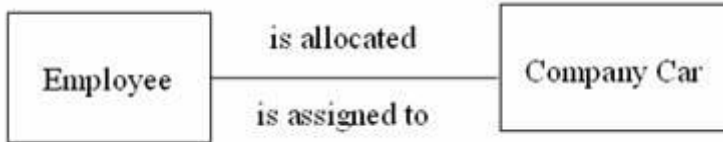
The latter one is correct, it is M:N and **not** M:M.

One-to-one (1:1)

This is where one occurrence of an entity relates to only one occurrence in another entity. A one-to-one relationship rarely exists in practice, but it can. However, you may consider combining them into one entity.

For example, an employee is allocated a company car, which can only be driven by that employee.

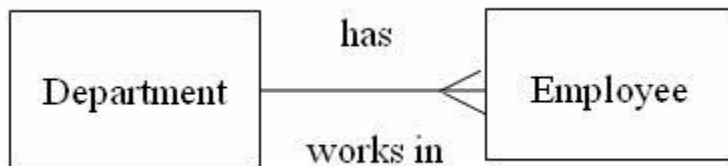
Therefore, there is a one-to-one relationship between employee and company car.



One-to-Many (1:M)

Is where one occurrence in an entity relates to many occurrences in another entity. For example, taking the employee and department entities shown on the previous page, an employee works in one department but a department has many employees.

Therefore, there is a one-to-many relationship between department and employee.



Many-to-Many (M:N)

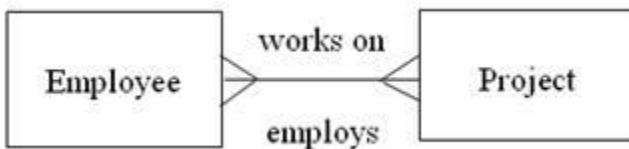
This is where many occurrences in an entity relate to many occurrences in another entity.

The normalisation process discussed earlier would prevent any such relationships but the definition is included here for completeness.

As with one-to-one relationships, many-to-many relationships rarely exist. Normally they occur because an entity has been missed.

For example, an employee may work on several projects at the same time and a project has a team of many employees.

Therefore, there is a many-to-many relationship between employee and project.



(iii) Integrity Rules

Integrity Rules are imperative to a good database design. Most RDBMS have these rules automatically, but it is safer to just make sure that the rules are already applied in the design. There are two types of integrity mentioned in integrity rules, entity and reference. Two additional rules that aren't necessarily included in integrity rules but are pertinent to database designs are business rules and domain rules.

Entity integrity exists when each primary key within a table has a value that is unique. this ensures that each row is uniquely identified by the primary key. One requirement for entity integrity is that a primary key cannot have a null value. The purpose of this integrity is to have each row to have a unique identity, and foreign key values can properly reference primary key values.

- Integrity rules are needed to inform the DBMS about certain constraints in the real world.
- Specific integrity rules apply to one specific database. Example: part weights must be greater than zero.
- General integrity rules apply to all databases. Two general rules will be discussed to deal with: primary keys and foreign keys.

(iv) Theta join .

Theta Join:

In theta join we apply the condition on input relation(s) and then only those selected

rows are used in the cross product to

be merged and included in the output. It means

that in normal cross product all the rows of one relation are mapped/merged with all

the rows of second relation, but here only selected rows of a relation are made cross

product with second relation. It is denoted as unde

If R and S are two relations then is the condition, which is applied for select operation on one relation and then only selected rows are cross product with all the rows of second relation. For Example there are two relations of $FACULTY$ and $COURSE$, now we will first apply select operation on the $FACULTY$ relation for selection certain specific rows then these rows will have across product with $COURSE$ relation, so this is the difference in between cross product and theta join. We will now see first both the relation their different attributes and then finally the cross product after carrying out select operation on relation. From this example the difference in between cross product and theta join.

B) What are view ? How they are managed ? Discuss the constrains on updating views.

Ans:A view is a "virtual table" in the database whose contents are defined by a query.

The tables of a database define the structure and organization of its data. However, SQL also lets you look at the stored data in other ways by defining alternative views of the data.

A view is a SQL query that is permanently stored in the database and assigned a name. The results of the stored query are "visible" through the view, and SQL lets you access these query results as if they were, in fact, a "real" table in the database.

Views are an important part of SQL, for several reasons:

- Views let you tailor the appearance of a database so that different users see it from different perspectives.
- Views let you restrict access to data, allowing different users to see only certain rows or certain columns of a table.
- Views simplify database access by presenting the structure of the stored data in the way that is most natural for each user.

(a) Explain the following fundamental operations in the relational algebra:-

- Select
- Project
- Union
- Set Difference

Ans:- Selection() Selects a subset of rows from relation.

Projection() Deletes unwanted columns from relation.

Set-difference() Tuples in relation 1, but not in relation 2.

Union () Tuples in relation 1 and in relation 2.

(i) *Selection*: $\sigma <condition(s)> (<relation>)$

- Picks tuples from the relation
- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of (only) input relation.
- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition*.)

Projection: Π <attribute-list> (<relation>)

- Picks columns from the relation
- Unary operation
- Denoted by uppercase pi Π .
- Returns a relation with only the specified attributes.
- Example: Π loan-number, amount (loan) lists all of the loan numbers and the amounts.

- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate *duplicates!* (Why??)
 - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

Union: (<relation>) U (<relation>)

- New relation contains all tuples from both relations, duplicate tuples eliminated.

- Denoted, as in set theory, by U.
- Binary operation
- Results in a relation with all of the tuples that appear in either or both of the argument relations.
- Unions must be between compatible relations
- Both relations must have the same number of attributes.
- Domains of the *i*th attribute of the first and the *i*th attribute of the second must be the same for all *i*.

- All of these operations take two input relations, which must be **union-compatible**:
 - Same number of fields.
 - 'Corresponding' fields have the same type.
- What is the *schema* of result?

Set Difference: $R - S$

- Produces a relation with tuples that are in R but NOT in S.
- Denoted by -
- Binary operation
- $R - S$ produces all tuples in R but not in S
- Relations must be compatible under the same conditions as the union operation.

(C)What is Referential Integrity ?How is it represented in the ER Model ? Explain the Referential Integrity Clauses in SQL.

Ans:-Defination: Referential integrity is a database concept that ensures that relationships between tables remain consistent. When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table. It also includes the techniques known as cascading update and cascading delete, which ensure that changes made to the linked table are reflected in the primary table.

Referential integrity :

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

Referential integrity is the relational property that each foreign key value in a table exists as a primary key in the referenced table.

Referential integrity relationships are defined with SQL FOREIGN KEY and PRIMARY KEY clauses in the CREATE TABLE statement and are automatically maintained both during load, update, and insert operations to a referencing table and during delete operations from a referenced table

(D)Explain Tuple relational calculus and its data manipulation facilities.

- Ans:- Relational Calculus
- Relational calculus query specifies *what* is to be retrieved rather than *how* to retrieve it.
 - No description of how to evaluate a query.
 - In first-order logic (or predicate calculus), *predicate* is a truth-valued function with arguments.
 - When we substitute values for the arguments, function yields an expression, called a *proposition*, which can be either true or false.
- Relational Calculus
- If predicate contains a variable (e.g. ‘*x* is a member of staff’), there must be a range for *x*.
 - When we substitute some values of this range for *x*, proposition may be true; for other values, it may be false.
 - When applied to databases, relational calculus has forms: *tuple* and *domain*.
- Tuple Relational Calculus
- Interested in finding tuples for which a predicate is true. Based on use of tuple variables.
 - Tuple variable is a variable that ‘ranges over’ a named relation: i.e., variable whose only permitted values are tuples of the relation.
 - Specify range of a tuple variable *S* as the Staff relation as:

- Staff(S)
- To find set of all tuples S such that P(S) is true:
{S | P(S)}
- Tuple Relational Calculus - Example
- To find details of all staff earning more than \$10,000:
{S | Staff(S) \wedge S.salary > 10000}
- To find a particular attribute, such as salary, write:
{S.salary | Staff(S) \wedge S.salary > 10000}
- Tuple Relational Calculus
- Can use two *quantifiers* to tell how many instances the predicate applies to:
 - Existential quantifier \exists ('there exists')
 - Universal quantifier \forall ('for all')
 - Tuple variables qualified by \forall or \exists are called *bound* variables, otherwise called *free* variables.
- Tuple Relational Calculus
- Existential quantifier used in formulae that must be true for at least one instance, such as:
Staff(S) \exists (\exists B)(Branch(B) \exists
(B.branchNo = S.branchNo) \exists B.city = 'London')
- Means 'There exists a Branch tuple with same branchNo as the branchNo of the current Staff tuple, S, and is located in London'.
- Tuple Relational Calculus
- Universal quantifier is used in statements about every instance, such as:
(\forall B) (B.city \neq 'Paris')
- Means 'For all Branch tuples, the address is not in Paris'.
- Can also use $\sim(\exists$ B) (B.city = 'Paris') which means 'There are no branches with an address in Paris'.
- Tuple Relational Calculus
- Formulae should be unambiguous and make sense.
- A (well-formed) formula is made out of atoms:
 - $R(S_i)$, where S_i is a tuple variable and R is a relation
 - $S_i.a_1 \text{ } q \text{ } S_j.a_2$
 - $S_i.a_1 \text{ } q \text{ } c$
- Can recursively build up formulae from atoms:
 - An atom is a formula
 - If F_1 and F_2 are formulae, so are their conjunction, $F_1 \wedge F_2$; disjunction, $F_1 \vee F_2$; and negation, $\sim F_1$
 - If F is a formula with free variable X , then $(\exists X)(F)$ and $(\forall X)(F)$ are also formulae.
- Example - Tuple Relational Calculus

a) List the names of all managers who earn more than \$25,000.

$\{S.fName, S.lName \mid Staff(S) \wedge S.position = 'Manager' \wedge S.salary > 25000\}$

b) List the staff who manage properties for rent in Glasgow.

$\{S \mid Staff(S) \wedge (\$P) (PropertyForRent(P) \wedge (P.staffNo = S.staffNo) \dot{\cup} P.city = 'Glasgow')\}$

- Tuple Relational Calculus
- Expressions can generate an infinite set. For example:

$\{S \mid \sim Staff(S)\}$

- To avoid this, add restriction that all values in result must be values in the domain of the expression.

- Data Manipulations in SQL
- Select, Update, Delete, Insert Statement
- Basic Data retrieval
- Condition Specification
- Arithmetic and Aggregate operators
- SQL Join: Multiple Table Queries
- Set Manipulation
 - Any, In, Contains, All, Not In, Not Contains, Exists, Union, Minus, Intersect
- Categorization
- Updates

- Creating Tables
- Empty tables are constructed using the CREATE TABLE statement.
- Data must be entered later using INSERT.

```
CREATE TABLE S ( SNO CHAR(5),
                 SNAME CHAR(20),
                 STATUS DECIMAL(3),
                 CITY CHAR(15),
                 PRIMARY KEY (SNO) )
```

- Creating Tables
- A table name and unique column names must be specified.
- Columns which are defined as primary keys will never have two rows with the same key value.
- **Primary key** may consist of more than one column (values unique *in combination*) called composite key.

- Creating Tables

```
CREATE TABLE SP ( SNO CHAR(5),
                  PNO CHAR(5),
                  QTY DECIMAL(5),
                  PRIMARY KEY (SNO,PNO),
                  FOREIGN KEY (SNO) REFERENCES S,
                  FOREIGN KEY (PNO) REFERENCES P )
```

- **Foreign** keys refer to other tables.
 - Any key in SP.SNO must also appear in S.SNO and any key in SP.PNO must appear in P.PNO - a shipment cannot exist unless the supplier and part also exist.
- Data Manipulation
- There are four basic SQL data manipulation operations.
 - **SELECT** - retrieves data
 - **INSERT** - add a new row
 - **UPDATE** - change values in existing records
 - **DELETE** - remove row(s)

- SELECT
- SELECT has the general form

SELECT-FROM-WHERE.

- The result is another (new) table.
- SELECT

```
SELECT DISTINCT P.COLOUR, P.CITY
FROM P
WHERE P.WEIGHT > 10
AND P.CITY <> 'Paris'
```

results in the table,

```
COLOUR  CITY
-----
```

```
Red     London
Blue    Rome
```

[Red London] - eliminated
because of
DISTINCT
statement which
removes multiple copies of rows

- SELECT
- DISTINCT - no duplicate rows.
- No WHERE - all rows of FROM table are returned.
- SELECT * is short for select the entire row (all columns).

- INSERT

```
INSERT
INTO SP ( SNO, PNO, QTY )
VALUES ( 'S4', 'P1', 1000 )
```

- Row added to Table SP.

- UPDATE

```
UPDATE S
SET STATUS = 2 * S.STATUS
```

WHERE S.CITY = 'London'

- Status doubled for suppliers in London (S1 and S4)

DELETE

DELETE

FROM P

WHERE P.WEIGHT > 15

- Rows deleted from P where WEIGHT > 15 (P2 and P3)



Tulsiramji Gaikwad-Patil College of Engineering & Technology

Department of Information Technology

Subject Notes

Academic Session: 2018 – 2019

Subject: DBMS

Semester: I

UNIT: IV

UNIT 4: Syllabus

Unit -4 Relational Database Design - Relational Schema, Relational Design, Functional dependency, Normalization, First-Second-Third Normal Forms, Relation with more than one Candidate Key, Good and Bad Decompositions, Multivalued Dependency, Fourth normal Form, Fifth Normal Form. **Network Data Model:** The Architecture of DBTG System, Schema & Subschema, and DBTG Data Manipulation Facility. **Hierarchical Data Model:** The tree Concept, Architecture of an IMS System, Data Manipulation.

(A) **What is join dependency ? Discuss 5NF .**

Ans: Join Dependencies (JD)

A join dependency can be described as follows:

1. If a table can be decomposed into three or more smaller tables, it must be capable of being joined again on common keys to form the original table

A table is in fifth normal form (5NF) or Projection-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables

Another way of expressing this is: and each join dependency is a consequence of the candidate keys.

It can also be expressed as: there are no pair wise cyclical dependencies in the primary key comprised of three or more attributes.

- Anomalies can occur in relations in 4NF if the primary key has three or more fields.
- 5NF is based on the concept of join dependence - if a relation cannot be decomposed any further then it is in 5NF.
- Pair wise cyclical dependency means that:
- You always need to know two values (pair wise).
- For any one you must know the other two (cyclical).

Example: Buying(buyer, vendor, item)

This is used to track buyers, what they buy, and from whom they buy.

Take the following sample data:

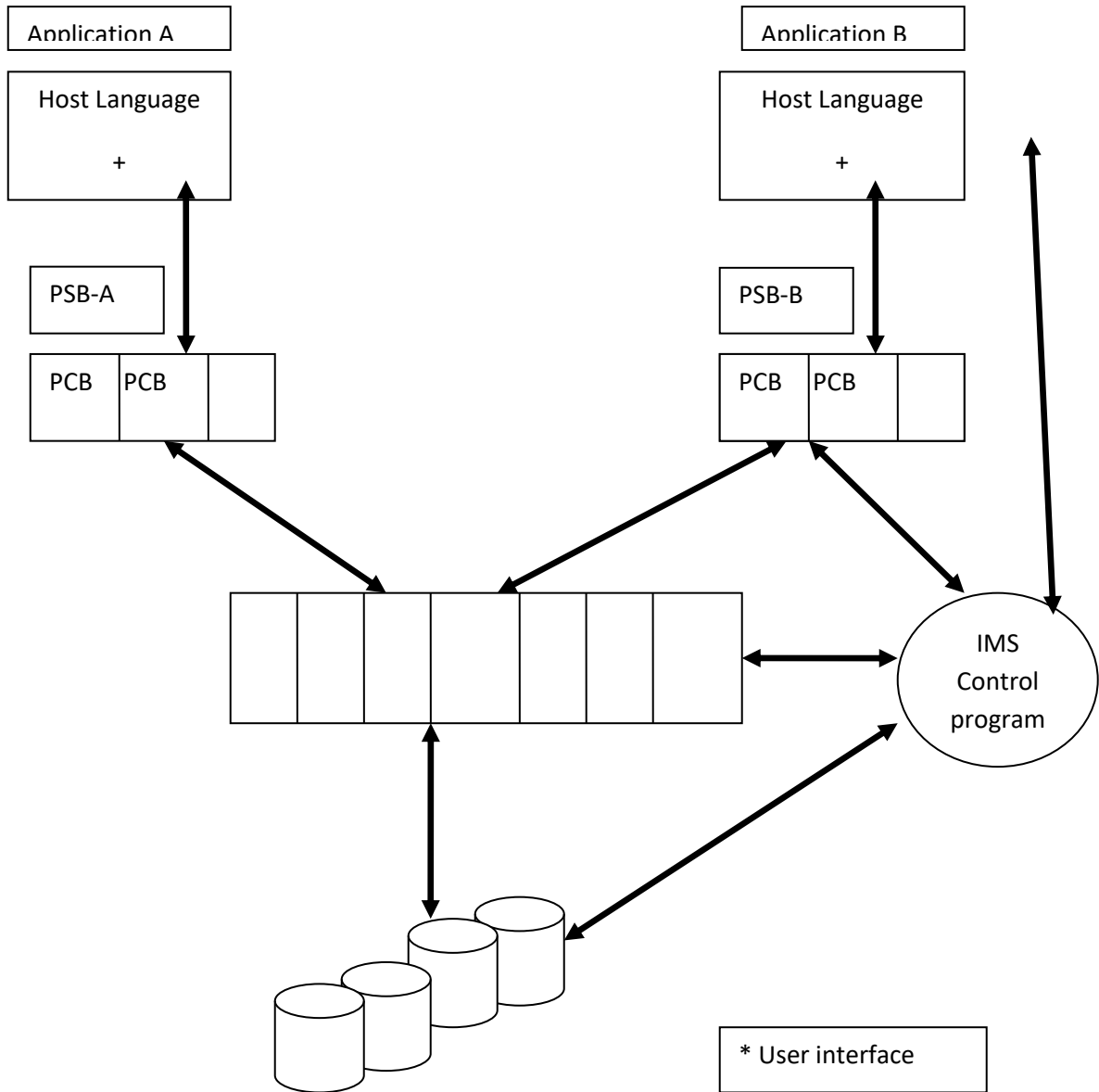
buyer	vendor	Item
Sally	Liz Claiborne	Blouses
Mary	Liz Claiborne	Blouses
Sally	Jordach	Jeans
Mary	Jordach	Jeans
Sally	Jordach	Sneakers

The question is, what do you do if Claiborne starts to sell Jeans? How many records must you create to record this fact?

The problem is there are pairwise cyclical dependencies in the primary key. That is, in order to determine the item you must know the buyer and vendor, and to determine the vendor you must know the buyer and the item, and finally to know the buyer you must know the vendor and the item. The solution is to break this one table into three tables; Buyer-Vendor, Buyer-Item, and Vendor-Item.

(B) Explain the architecture of an IMS System.

Ans: Information Management system (IMS) is an IBM program product that is designed to support both batch and online application programs.



Conceptual View

The conceptual view consists of collection of physical database. The “physical” is somewhat misleading in this context, since the user does not see such a database exactly as it is stored; indeed, IMS provides a fairly high degree of insulation of the user from the storage structure. Each physical database is defined by a database description (DBD). The mapping of the physical database to storage is also DBD’s corresponds to the conceptual schema plus the associated conceptual/internal mapping definition.

DBD (Database Description): Each physical database is defined, together with its mapping to storage, by a database description (DBD). The source form of the DBD is written using a special System/370 Assembler Language macro statements. Once written the DBD is assembled and the object form is stored in a system library, from which it may be extracted when required by the IMS control program.

All names of DBD’s in IMS are limited to a maximum length of eight characters.

Example:

1. DBD NAME:EDUCPDBD
2. SEGM NAME=COURSE,BYTES=256
3. FILED NAME=(COURSE#,SEQ),BYTES=3,START=1
4. FIELD NAME=TITLE, BYTES=33,START=4
5. FIELD NAME=DESCRIPN,BYETS=220,START=37
6. SEGM NAME=PREREQ,PARENT=COURSE,BYTES=36
7. FILED NAME=(COURSE#,SEQ),BYTES=3,START=1
8. FIELD NAME=TITLE, BYTES=33,START=4
9. SEGM NAME=OFFERING,PARENT=COURSE,BYTES=20
10. FILED NAME=(DATE,SEQ,M),BYTES=6,START=1
11. FIELD NAME=LOCATION, BYTES=12,START=7
12. FIELD NAME=FORMAT,BYETS=2,START=19
13. SEGM NAME=TEACHER,PARENT=OFFERING,BYTES=24

14. FIELD NAME=(EMP#,SEQ), BYTES=6,START=1

15. FIELD NAME=NAME,BYETS=18,START=7

16. SEGM NAME=STUDENT,PARENT=OFFERING,BYTES=25

17. FILED NAME=(EMP#,SEQ),BYTES=6,START=1

18. FIELD NAME=NAME, BYTES=18,START=7

19. FIELD NAME=GRADE,BYTES=1,START=25

External View:

The user does not operate directly at the physical database level but rather on an “external view” of the data. A particular user’s external view consists of a collection of “logical databases”, where each logical database is a subset of the corresponding physical database. Each logical database is defined by means of a program communication block (PCB). The set of all PCB’s for one user, corresponding to the external schema plus the associated mapping definition, is called program specification block (PSB).

PCB: Program Communication BLOCK: Each logical Database is defined by a program communication block (PCB). The PCB includes a specification of the mapping between the LDB and the corresponding PDB.

PSB: Program Specification BLOCK: The set of all PCB’s for a given user forms that user’s program specification block (PSB)

Example:

1. PCB TYPE=DB,DBNAME=EDUCPDBD,KEYLEN=15
2. SENSEG NAME=COURSE,PROCOPT=G
3. SENSEG NAME=OFFERING,PARENT=COURSE,PROCOPT=G
4. SENSEG NAME=STUDENT,PARENT=OFFERING,PROCOPT=G

PROCOPT: The PROCOPT entry specifies the types of operation that the user will be permitting to perform on this segment. In this example the entry is G (“get”), indicating retrieval only. Other possible values are I(“insert”), R(“replace”), and D(“delete”)

Internal View

The users are ordinary application programmers, using a host language from which the IMS data manipulation language DL/I- “Data Language/I”- may be invoked by subroutine call. End-users are supported via user-written on-line application programs. IMS does not provide an integrated query language.

(C) Explain the following :-

(i) **Functional dependency.**

Functional Dependency: The value of one attribute (the *determinant*) determines the value of another attribute.

Candidate Key: A possible key.

Each non-key field is functionally dependent on every candidate key.

No attribute in the key can be deleted without destroying the property of unique identification

Main characteristics of functional dependencies used in normalization:

have a 1:1 relationship between attribute(s) on left and right-hand side of a dependency; hold for all time; are nontrivial.

Complete set of functional dependencies for a given relation can be very large.

Important to find an approach that can reduce set to a manageable size.

Need to identify set of functional dependencies (X) for a relation that is smaller than complete set of functional dependencies (Y) for that relation and has property that every functional dependency in Y is implied by functional dependencies in X.

(ii) Multivalued dependency

No answer

(D) Explain 4 NF with examples.

Ans :Normalization: The process of decomposing unsatisfactory “bad” relations by breaking up their attributes into smaller relations. The normal form of a relation refers to the highest normal form condition that a relation meets and indicates the degree to which it has been normalized.

Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties.

Normalization in industry pays particular attention to normalization up to 3NF, BCNF, or 4NF.

We will pay particular attention up to 3NF.

The database designers *need not* normalize to the highest possible normal form.

Formal technique for analyzing a relation based on its primary key and functional dependencies between its attributes.

Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.

As normalization proceeds, relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

1. **NF2:** non-first normal form
2. **1NF:** R is in 1NF. **iff** all domain values are atomic
3. **2NF:** R is in 2. NF. **iff** R is in 1NF and every nonkey attribute is fully dependent on the key
4. **3NF:** R is in 3NF **iff** R is 2NF and every nonkey attribute is non-transitively dependent on the key
5. **BCNF:** R is in BCNF **iff** every determinant is a candidate key
6. **Determinant:** an attribute on which some other attribute is fully functionally dependent.

Fourth Normal Form:

Fourth normal form (or 4NF) requires that there are no non-trivial multi-valued dependencies of attribute sets on something other than a superset of a candidate key. A table is said to be in 4NF if and only if it is in the BCNF and multi-valued dependencies are functional dependencies. The 4NF removes unwanted data structures: multi-valued dependencies

There is no Multivalued dependency in the relation

There are Multivalued dependency but the attributes are dependent between themselves

Either of these conditions must hold true in order to be fourth normal form

The relation must also be in BCNF. Fourth normal form differs from BCNF only in that it uses Multivalued dependencies.

E) What are inference axioms ? Explain its significance in Relational Database Design .

Ans: Inference Axioms (A-axioms or Armstrong's Axioms)

An inference axiom is a rule that states if a relation satisfies certain FDs then it must satisfy certain other FDs.

F1. Reflexivity $X \twoheadrightarrow X$

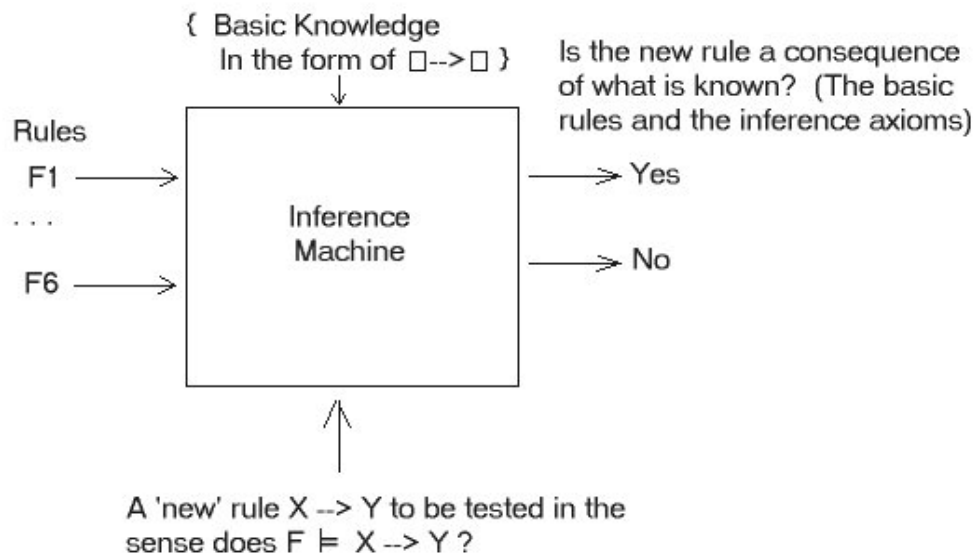
F2. Augmentation If $(Z \twoheadrightarrow W; X \twoheadrightarrow Y)$ then $XW \twoheadrightarrow YZ$

F3. Additivity If $\{ (X \twoheadrightarrow Y) (X \twoheadrightarrow Z) \}$ then $X \twoheadrightarrow YZ$

F4. Projectivity If $(X \twoheadrightarrow YZ)$ then $X \twoheadrightarrow Y$

F5. Transitivity If $(X \twoheadrightarrow Y)$ and $(Y \twoheadrightarrow Z)$ then $(X \twoheadrightarrow Z)$

F6. Pseudotransitivity If $(X \twoheadrightarrow Y)$ and $(YZ \twoheadrightarrow W)$ then $XZ \twoheadrightarrow W$



Examples of the use of Inference Axioms

[From Ullman]

1. Consider $R = (\text{Street}, \text{Zip}, \text{City})$; $F = \{\text{City Street} \sqsubseteq \text{Zip}, \text{Zip} \sqsubseteq \text{City}\}$

We want to show: $\text{Street Zip} \sqsubseteq \text{Street Zip City}$

Proof:

1. $\text{Zip} \sqsubseteq \text{City}$ – Given
2. $\text{Street Zip} \sqsubseteq \text{Street City}$ – Augmentation of (1) by Street
3. $\text{City Zip} \sqsubseteq \text{Zip}$ – Given
4. $\text{City Street} \sqsubseteq \text{City Street Zip}$ – Augmentation of (3) by City Street
5. $\text{Street Zip} \sqsubseteq \text{City Street Zip}$ – Transitivity of (2) and (4)

[From Maier]

1. Let $R = (\text{ABCDEFGHI})$ $F = \{\text{AB} \sqsubseteq \text{E}, \text{AG} \sqsubseteq \text{J}, \text{BE} \sqsubseteq \text{I}, \text{E} \sqsubseteq \text{G}, \text{GI} \sqsubseteq \text{H}\}$

Show that $\text{AB} \sqsubseteq \text{GH}$ is derived by F

1. $\text{AB} \sqsubseteq \text{E}$ - Given
2. $\text{AB} \sqsubseteq \text{AB}$ – Reflexivity
3. $\text{AB} \sqsubseteq \text{B}$ - Projectivity from (2)
4. $\text{AB} \sqsubseteq \text{BE}$ – Additivity from (1) and (3)
5. $\text{BE} \sqsubseteq \text{I}$ - Given
6. $\text{AB} \sqsubseteq \text{I}$ – Transitivity from (4) and (5)
7. $\text{E} \sqsubseteq \text{G}$ – Given
8. $\text{AB} \sqsubseteq \text{G}$ – Transitivity from (1) and (7)
9. $\text{AB} \sqsubseteq \text{GI}$ – Additivity from (6) and (8)
10. $\text{GI} \sqsubseteq \text{H}$ – Given
11. $\text{AB} \sqsubseteq \text{H}$ – Transitivity from (9) and (10)
12. $\text{AB} \sqsubseteq \text{GH}$ – Additivity from (8) and (11)

Significance in Relational Database design: A database structure commonly used in GIS in which data is stored based on 2 dimensional tables where multiple relationships between data elements can be defined and established in an ad-hoc manner. relational Database Management System - a database system made up of files with data elements in two-dimensional array (rows and columns). This database management system has the capability to recombine data elements to form different relations resulting in a great flexibility of data usage

A database that is perceived by the user as a collection of twodimensional tables

- Are manipulated a set at a time, rather than a record at a time

- SQL is used to manipulate relational databases Proposed by Dr. Codd in 1970
- The basis for the relational database management system (RDBMS)
- The relational model contains the following components:
 - Collection of objects or relations
 - Set of operations to act on the relations
 - Data integrity for accuracy and consistency

(a) List and explain different types of functional dependency in detail.

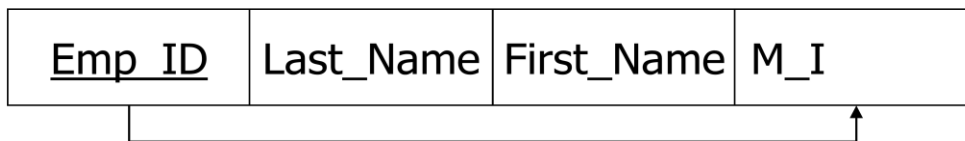
Ans: Functional Dependency: The value of one attribute (the *determinant*) determines the value of another attribute.

Candidate Key: A possible key.

Each non-key field is functionally dependent on every candidate key.

No attribute in the key can be deleted without destroying the property of unique identification.

EMPLOYEE

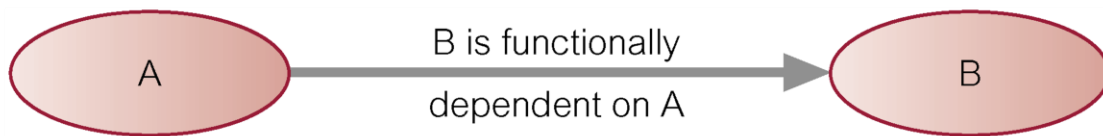


Alternate way to show dependencies

Emp_ID → Last_Name, First_Name, M_I

- Main concept associated with normalization.
- Functional Dependency
 - Describes relationship between attributes in a relation.
 - If A and B are attributes of relation R, B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with exactly one value of B in R.

- Property of the meaning (or semantics) of the attributes in a relation.
- Diagrammatic representation:



□ *Determinant* of a functional dependency refers to attribute or group of attributes on left-hand side of the arrow.

- Complete set of functional dependencies for a given relation can be very large.
- Important to find an approach that can reduce set to a manageable size.
- Need to identify set of functional dependencies (X) for a relation that is smaller than complete set of functional dependencies (Y) for that relation and has property that every functional dependency in Y is implied by functional dependencies in X.
- Set of all functional dependencies implied by a given set of functional dependencies X called closure of X (written X^+).
- Set of inference rules, called Armstrong's axioms, specifies how new functional dependencies can be inferred from given ones.
- Let A, B, and C be subsets of the attributes of relation R. Armstrong's axioms are as follows:
 1. Reflexivity

If B is a subset of A, then $A \twoheadrightarrow B$

2. Augmentation

If $A \twoheadrightarrow B$, then $A, C \twoheadrightarrow B, C$

3. Transitivity

If $A \twoheadrightarrow B$ and $B \twoheadrightarrow C$, then $A \twoheadrightarrow C$

- Y is **functionally dependent** on X in R iff for each $x \in R.X$ there is precisely one $y \in R.Y$
- Y is **fully functional dependent** on X in R if Y is functional dependent on X and Y is not functional dependent on any proper subset of X

D) Explain fourth Normal form with suitable example.

Que Normalization: The process of decomposing unsatisfactory “bad” relations by breaking up their attributes into smaller relations

The normal form of a relation refers to the highest normal form condition that a relation meets and indicates the degree to which it has been normalized.

Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties

5 Fourth normal form (or 4NF) requires that there are no non-trivial multi-valued dependencies of attribute sets on something other than a superset of a candidate key. A table is said to be in 4NF if and only if it is in the BCNF and multi-valued dependencies are functional dependencies. The 4NF removes unwanted data structures: multi-valued dependencies

There is no Multivalued dependency in the relation

There are Multivalued dependency but the attributes are dependent between themselves

Either of these conditions must hold true in order to be fourth normal form

The relation must also be in BCNF

Fourth normal form differs from BCNF only in that it uses Multivalued dependencies

The fifth normal form deals with join-dependencies which is a generalizations of the MVD. The aim of fifth normal form is to have relations that cannot be decomposed further.

A relation in 5NF cannot be constructed from several smaller relations.

A relation R satisfies join dependency (R_1, R_2, \dots, R_n) if and only if R is equal to the join of R_1, R_2, \dots, R_n where R_i are subsets of the set of attributes of R .

(E)What is Normalization ?Convert any unnormalized relation into 1NF 2NF and 3NF.

- Ans: **Normalization:** The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- Normalization in industry pays particular attention to normalization up to 3NF, BCNF, or 4NF.
- We will pay particular attention up to 3NF.

■ **NF²:** non-first normal form

■ **1NF:** R is in 1NF. **iff** all domain values are atomic

■ **2NF:** R is in 2. NF. **iff** R is in 1NF and every nonkey attribute is fully dependent on the key

■ **3NF:** R is in 3NF **iff** R is 2NF and every nonkey attribute is non-transitively dependent on the key

Unnormalized Form (UNF)

■ A table that contains one or more repeating groups.

■ To create an unnormalized table:

- transform data from information source (e.g. form) into table format with columns and rows.

■ First Normal Form (1NF)

■ A relation in which intersection of each row and column contains one and only one value.

■ If a table of data meets the definition of a relation, it is in first normal form

- Every relation has a unique name.
- Every attribute value is atomic (single-valued).
- Every row is unique.
- Attributes in tables have unique names.
- The order of the columns is irrelevant.
- The order of the rows is irrelevant

■ UNF to 1NF

■ Nominate an attribute or group of attributes to act as the key for the unnormalized table.

■ Identify repeating group(s) in unnormalized table which repeats for the key attribute(s).

■ Remove repeating group by:

- entering appropriate data into the empty columns of rows containing repeating data ('flattening' the table).

Or by

- placing repeating data along with copy of the original key attribute(s) into a separate relation.

■ Second Normal Form (2NF)

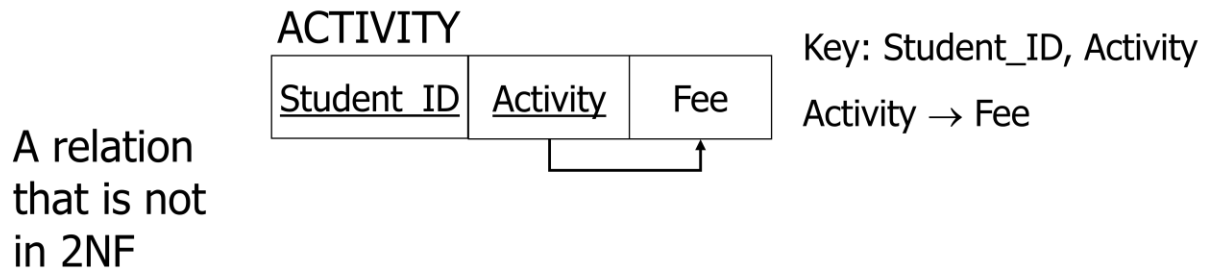
■ Based on concept of full functional dependency:

- A and B are attributes of a relation,
- B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.
- 2NF - A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.

■ Second Normal Form (2NF)

- 1NF and no partial functional dependencies.
- Partial functional dependency: when one or more non-key attributes are functionally dependent on part of the primary key.
- Every non-key attribute must be defined by the entire key, not just by part of the key.
- If a relation has a single attribute as its key, then it is automatically in 2NF.

Second Normal Form (2NF)

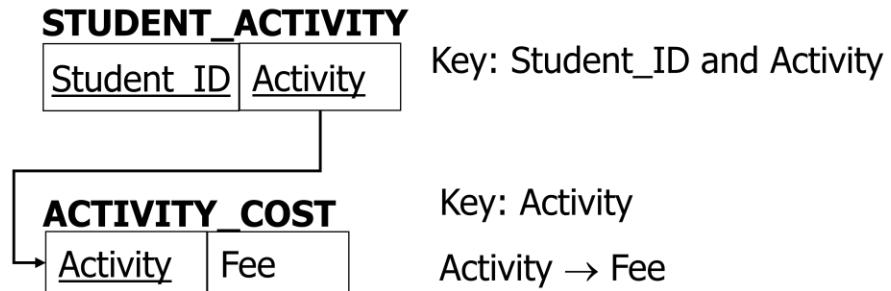


Fee is determined by Activity

Student_ID	Activity	Fee
222-22-2020	Swimming	30
232-22-2111	Golf	100
222-22-2020	Golf	100
255-24-2332	Hiking	50

Second Normal Form (2NF)

Divide the relation into two relations that now meet 2NF



Student_ID	Activity
222-22-2020	Swimming
232-22-2111	Golf
222-22-2020	Golf
255-24-2332	Hiking

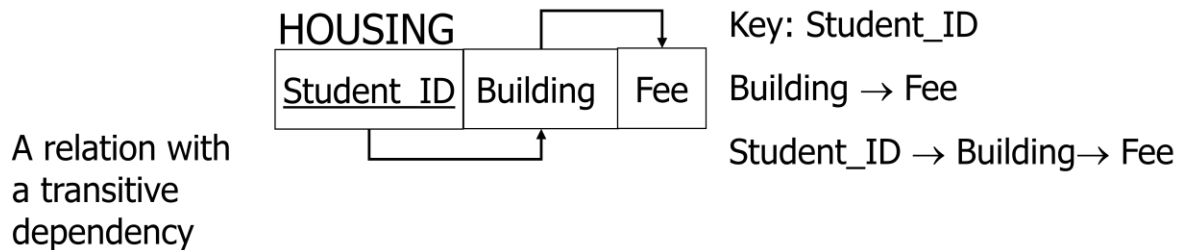
Activity	Fee
Swimming	30
Golf	100
Hiking	50

- 1NF to 2NF
- Identify primary key for the 1NF relation.
- Identify functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them by placing them in a new relation along with copy of their determinant.
- Third Normal Form (3NF)
- 2NF and no transitive dependencies
- Transitive dependency: a functional dependency between two or more non-key attributes

Based on concept of transitive dependency:

- ❑ A, B and C are attributes of a relation such that if $A \rightarrow B$ and $B \rightarrow C$, then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).
- ❑ 3NF - A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.

Third Normal Form (3NF)



Student_ID	Building	Fee
222-22-2020	Dabney	1200
232-22-2111	Liles	1000
222-22-5554	The Range	1100
255-24-2332	Dabney	1200
330-25-7789	The Range	1100

- 2NF to 3NF
- Identify the primary key in the 2NF relation.
- Identify functional dependencies in the relation.
- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with copy of their determinant.

(D) Explain the tree concept in Hierarchical Data Model.

And:-

In **hierarchical model**, data is organized into a tree like structure with each record is having one parent record and many children. The main drawback of this model is that, it can have only one to many relationships between nodes.

Sample

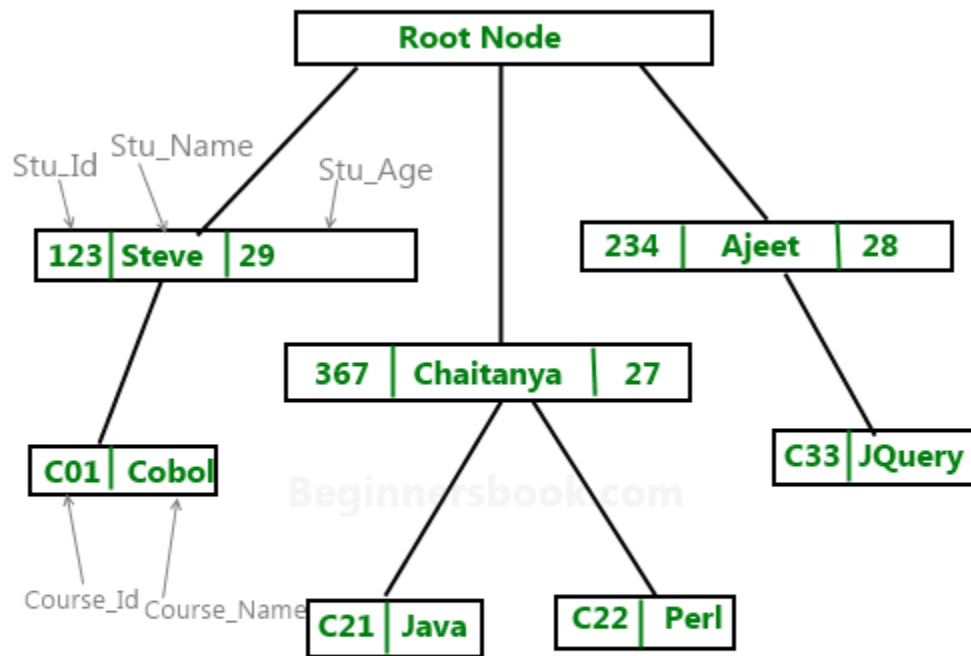
Hierarchical

Model

Diagram:

Lets say we have few students and few courses and a course can be assigned to a single student only,

however a student take any number of courses so this relationship becomes one to many.





Tulsiramji Gaikwad-Patil College of Engineering & Technology

Department of Information Technology

Subject Notes

Academic Session: 2018 – 2019

Subject: DBMS

Semester: I

UNIT: V

UNIT V: Syllabus

Unit –V

Database Operations and Maintenance - Database Administrator (DBA), Database Security, Integrity And Control (User with Password and Complete/Limited Authorization, Encryption of Data). **Concurrency Control:** Problem of Concurrent Access, resource locking, Deadlock. **Database Recovery:** Restore, Backward & Forward Recovery. **Distributed Database:** Introduction, Data Distribution, Deadlock in Distributed Systems, Security and Protection, Homogeneous and Heterogeneous Systems. Knowledge Base and Database Systems, Expert Database Systems, Object Database System.

(A) What are object oriented database systems? What are its features.

Ans: Object databases are a niche field within the broader DBMS market dominated by relational database management systems (RDBMS). Object databases have been considered since the early 1980s and 1990s but they have made little impact on mainstream commercial data proc

Features of object oriented database systems:

Most object databases also offer some kind of query language, allowing objects to be found by a more declarative programming approach. It is in the area of object query languages, and the integration of the query and navigational interfaces, that the biggest differences between products are found. An attempt at standardization was made by the ODMG with the Object Query Language, OQL.

Access to data can be faster because joins are often not needed (as in a tabular implementation of a relational database). This is because an object can be retrieved directly without a search, by following pointers. (It could, however, be argued that "joining" is a higher-level abstraction of pointer following.)

Another area of variation between products is in the way that the schema of a database is defined. A general characteristic, however, is that the programming language and the database schema use the same type definitions.

Multimedia applications are facilitated because the class methods associated with the data are responsible for its correct interpretation.

Many object databases, for example VOSS, offer support for versioning. An object can be viewed as the set of all its versions. Also, object versions can be treated as objects in their own right. Some object databases also provide systematic support for triggers and constraints which are the basis of active databases.

The efficiency of such a database is also greatly improved in areas which demand massive amounts of data about one item. For example, a banking institution could get the user's account information and provide them efficiently with extensive information such as transactions, account information entries etc.

B) How database recovery is done? Discuss its different types.

Ans: SQL Server database recovery models give you backup-and-restore flexibility. The model used will determine how much time and space your backups will take and how great your risk of data loss will be when a breakdown occurs.

System breakdowns happen all the time even to the best configured systems. This is why you have to explore the options available in order to prepare for the worst!

SQL server database recovery can be easier achieved if you are running on at least the SQL server 2000. It has a built in feature known as the **database recovery model** that controls the following

- Both the speed and size of your transaction log backups.
- The degree to which you might be at risk of losing committed transactions in the event of media failure.

Models

There are three types of database recovery models available

- Full Recovery
- Bulk Logged Recovery
- Simple Recovery

Full

Recovery:

This is your best guarantee for full data recovery. The SQL Server fully logs all operations, so every row inserted through a bulk copy program (bcp) or BULK INSERT operation is written in its entirety to the transaction log. When data files are lost because of media failure the transaction log can be backed up.

- Database restoration up to any specified time can be achieved after media failure for a database file has occurred. If your log file is available after the failure, you can restore up to the last transaction committed.
- Log Marks feature allows you to place reference points in the transaction log that allow you to recover a log mark.
- Logs CREATE INDEX operations. Recovery from a transaction log backup that includes index creations is done at a faster pace because the index does not have to be rebuilt.

BulkLoggedRecoveryModel

This model allows for recovery in case of media failure and gives you the best performance using the least log space for certain bulk operations, including BULK INSERT, bcp, CREATE INDEX, WRITETEXT, and UPDATETEXT.

Simple Recovery Model

It allows for the fastest bulk operations and the simplest backup-and-restore strategy. Under this model, SQL Server truncates the transaction log at regular intervals, removing committed transactions. Only full database backups and differential backups are allowed.

C) Describe Deadlocks a Distributed System.

Ans: A **deadlock** is a condition in a system where a set of processes (or threads) have requests for resources that can never be satisfied. Essentially, a process cannot proceed because it needs to obtain a resource held by another process but it itself is holding a resource that the other process needs. More formally, Coffman defined four conditions have to be met for a deadlock to occur in a system:

1. **Mutual exclusion** A resource can be held by at most one process.
2. **Hold and wait** Processes that already hold resources can wait for another resource.
3. **Non-preemption** A resource, once granted, cannot be taken away.
4. **Circular wait** Two or more processes are waiting for resources held by one of the other processes.

A directed graph model used to record the resource allocation state of a system. This state consists of n processes, $P_1 \dots P_n$, and m resources, $R_1 \dots R_m$. In such a graph:

$P_1 \rightarrow R_1$ means that resource R_1 is allocated to process P_1 .

$P_1 \leftarrow R_1$ means that resource R_1 is requested by process P_1 .

Deadlock is present when the graph has a directed cycles. An example is shown in Figure 1. Such a graph is called a **Wait-For Graph (WFG)**.

Deadlock in distributed systems

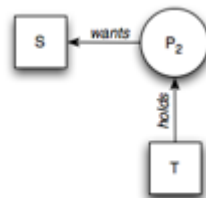


Figure 2. Resource graph on A

Figure 3. Resource graph on B

The same conditions for deadlock in uniprocessors apply to distributed systems. Unfortunately, as in many other aspects of distributed systems, they are harder to detect, avoid, and prevent. Four strategies can be used to handle deadlock:

1. **ignorance**: ignore the problem; assume that a deadlock will never occur. This is a surprisingly common approach.
2. **detection**: let a deadlock occur, detect it, and then deal with it by aborting and later restarting a process that causes deadlock.
3. **prevention**: make a deadlock impossible by granting requests so that one of the necessary conditions for deadlock does not hold.
4. **avoidance**: choose resource allocation carefully so that deadlock will not occur. Resource requests can be honored as long as the system remains in a safe (non-deadlock) state after resources are allocated.

The last of these, deadlock avoidance through resource allocation is difficult and requires the ability to predict precisely the resources that will be needed and the times that they will be needed. This is difficult and not practical in real systems. The first of these is trivially simple but, of course, ineffective for actually doing anything about deadlock conditions. We will focus on the middle two approaches.

In a conventional system, the operating system is the component that is responsible for resource allocation and is the ideal entity to detect deadlock. Deadlock can be resolved by killing a process. This, of course, is not a good thing for the process. However, if processes are transactional in nature, then aborting the transaction is an anticipated operation. Transactions are designed to withstand being aborted and, as such, it is perfectly reasonable to abort one or more transactions to break a deadlock. The transaction can be restarted later at a time when, we hope, it will not create another deadlock.

Centralized deadlock detection

Centralized deadlock detection attempts to imitate the nondistributed algorithm through a central coordinator. Each machine is responsible for maintaining a resource graph for its processes and resources. A central coordinator maintains the resource utilization graph for the entire system: the **Global Wait-For Graph**. This graph is the union of the individual Wait-For Graphs. If the coordinator detects a cycle in the global wait-for graph, it aborts one process to break the deadlock.

In the non-distributed case, all the information on resource usage lives on one system and the graph may be constructed on that system. In the distributed case, the individual subgraphs have to be propagated to a central coordinator. A message can be sent each time an arc is added or deleted. If optimization is needed, a list of added or deleted arcs can be sent periodically to reduce the overall number of messages sent.

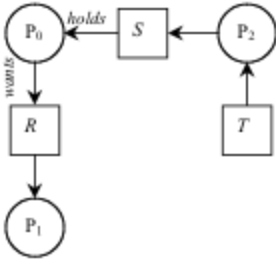


Figure 4. Resource graph on coordinator

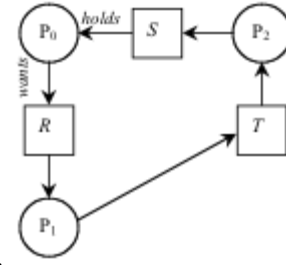


Figure 5.

False deadlock

Here is an example (from Tanenbaum). Suppose machine A has a process P_0 , which holds the resource S and wants resource R , which is held by P_1 . The local graph on A is shown in Figure 2. Another machine, machine B, has a process P_2 , which is holding resource T and wants resource S . Its local graph is shown in Figure 3. Both of these machines send their graphs to the central coordinator, which maintains the union (Figure 4).

All is well. There are no cycles and hence no deadlock. Now two events occur. Process P_1 releases resource R and asks machine B for resource T . Two messages are sent to the coordinator:

message 1 (from machine A): “releasing R ”

message 2 (from machine B): “waiting for T ”

This should cause no problems (no deadlock). However, if message 2 arrives first, the coordinator would then construct the graph in Figure 5 and detect a deadlock. Such a condition is known as **false deadlock**. A way to fix this is to use Lamport’s algorithm to impose global time ordering on all machines. Alternatively, if the coordinator suspects deadlock, it can send a reliable message to every machine asking whether it has any release messages. Each machine will then respond with either a release message or a negative acknowledgement to acknowledge receipt of the message.

Distributed deadlock detection

An algorithm for detecting deadlocks in a distributed system was proposed by Chandy, Misra, and Haas in 1983. Processes request resources from the current holder of that resource. Some processes may wait for resources, which may be held either locally or remotely. Cross-machine arcs make looking for cycles, and hence detecting deadlock, difficult. This algorithm avoids the problem of constructing a Global WFG.

The **Chandy-Misra-Haas algorithm** works this way: when a process has to wait for a resource, a **probe** message is sent to the process holding that resource. The probe message contains three components: the process ID that blocked, the process ID that is sending the request, and the destination. Initially, the first two components will be the same. When a process receives the probe: if the process itself is waiting on a resource, it updates the sending and destination fields

of the message and forwards it to the resource holder. If it is waiting on multiple resources, a message is sent to each process holding the resources. This process continues as long as processes are waiting for resources. If the originator gets a message and sees its own process number in the blocked field of the message, it knows that a cycle has been taken and deadlock exists. In this case, some process (transaction) will have to die. The sender may choose to commit suicide and abort itself or an election algorithm may be used to determine an alternate victim (e.g., youngest process, oldest process, ...).

Distributed deadlock prevention

An alternative to detecting deadlocks is to design a system so that deadlock is impossible. We examined the four conditions for deadlock. If we can deny at least one of these conditions then we will not have deadlock.

Mutual exclusion

To deny this means that we will allow a resource to be held (used) by more than one process at a time. If a resource can be shared then there is no need for mutual exclusion and deadlock cannot occur. Too often, however, a process requires mutual exclusion for a resource because the resource is some object that will be modified by the process.

Hold and wait

Denying this means that processes that hold resources cannot wait for another resource. This typically implies that a process should grab all of its resources at once. This is not practical either since we cannot always predict what resources a process will need throughout its execution.

Non-preemption

A resource, once granted, cannot be taken away. In transactional systems, allowing preemption means that a transaction can come in and modify data (the resource) that is being used by another transaction. This differs from mutual exclusion since the access is not concurrent but the same problem arises of having multiple transactions modify the same resource. We can support this with optimistic concurrency control algorithms that will check for out-of-order modifications at commit time and roll back (abort) if there are potential inconsistencies.

Circular wait

Avoiding circular wait means that we ensure that a cycle of waiting on resources does not occur. We can do this by enforcing an ordering on granting resources and aborting transactions or denying requests if an ordering cannot be granted.

One way of avoiding circular wait is to obtain a globally-unique timestamp (e.g., Lamport total ordering) for every transaction so that no two transactions get the same timestamp. When one process is about to block waiting for a resource that another process is using, check which of the two processes has a younger timestamp and give priority to the older process.

If a younger process is using the resource, then the older process (that wants the resource) waits. If an older process is holding the resource, the younger process (that wants the resource) aborts itself. This forces the resource utilization graph to be directed from older to younger processes, making cycles impossible. This algorithm is known as the **wait-die algorithm**.

Deadlocks in Distributed Systems

- Deadlocks in distributed systems are similar to deadlocks in single processor systems, only worse.
 - They are harder to avoid, prevent or even detect.
 - They are hard to cure when tracked down because all relevant information is scattered over many machines.
- People sometimes might classify deadlock into the following types:
 - Communication deadlocks -- competing with buffers for send/receive
 - Resources deadlocks -- exclusive access on I/O devices, files, locks, and other resources.
- We treat everything as resources, there we only have resources deadlocks.
- Four best-known strategies to handle deadlocks:
 - The ostrich algorithm (ignore the problem)
 - Detection (let deadlocks occur, detect them, and try to recover)
 - Prevention (statically make deadlocks structurally impossible)
 - Avoidance (avoid deadlocks by allocating resources carefully)

The FOUR Strategies for handling deadlocks

- The ostrich algorithm
 - No dealing with the problem at all is as good and as popular in distributed systems as it is in single-processor systems.
 - In distributed systems used for programming, office automation, process control, no system-wide deadlock mechanism is present -- distributed databases will implement their own if they need one.
- Deadlock detection and recovery is popular because prevention and avoidance are so difficult to implement.
- Deadlock prevention is possible because of the presence of atomic transactions. We will have two algorithms for this.
- Deadlock avoidance is never used in distributed system, in fact, it is not even used in single processor systems.
 - The problem is that the banker's algorithm need to know (in advance) how much of each resource every process will eventually need. This information is rarely, if ever, available.
- Hence, we will just talk about deadlock detection and deadlock prevention.

Distributed Deadlock Detection

- Since preventing and avoiding deadlocks to happen is difficult, researchers works on detecting the occurrence of deadlocks in distributed system.
- The presence of atomic transaction in some distributed systems makes a major conceptual difference.
 - When a deadlock is detected in a conventional system, we kill one or more processes to break the deadlock --- one or more unhappy users.
 - When deadlock is detected in a system based on atomic transaction, it is resolved by aborting one or more transactions.
 - But transactions have been designed to withstand being aborted.
 - When a transaction is aborted, the system is first restored to the state it had before the transaction began, at which point the transaction can start again.
 - With a bit of luck, it will succeed the second time.
 - Thus the difference is that the consequences of killing off a process are much less severe when transactions are used.

D) Explain deadlock detection, prevention and recovery technique in detail.

Ans:

Distributed Deadlock Detection

- Since preventing and avoiding deadlocks to happen is difficult, researchers work on detecting the occurrence of deadlocks in distributed systems.
- The presence of atomic transactions in some distributed systems makes a major conceptual difference.
 - When a deadlock is detected in a conventional system, we kill one or more processes to break the deadlock --- one or more unhappy users.
 - When a deadlock is detected in a system based on atomic transactions, it is resolved by aborting one or more transactions.
 - But transactions have been designed to withstand being aborted.
 - When a transaction is aborted, the system is first restored to the state it had before the transaction began, at which point the transaction can start again.
 - With a bit of luck, it will succeed the second time.
 - Thus the difference is that the consequences of killing off a process are much less severe when transactions are used.

Distributed Deadlock Prevention

- A method that might work is to order the resources and require processes to acquire them in strictly increasing order. This approach means that a process can never hold a high resource and ask for a low one, thus making cycles impossible.
- With global timing and transactions in distributed systems, two other methods are possible -- both based on the idea of assigning each transaction a global timestamp at the moment it starts.
- When one process is about to block waiting for a resource that another process is using, a check is made to see which has a larger timestamp.
- We can then allow the wait only if the waiting process has a lower timestamp.
- The timestamp is always increasing if we follow any chain of waiting processes, so cycles are impossible --- we can use decreasing order if we like.
- It is wiser to give priority to old processes because
 - they have run longer so the system has larger investment on these processes.
 - they are likely to hold more resources.
 - A young process that is killed off will eventually age until it is the oldest one in the system, and that eliminates starvation.

Deadlock recovery: Some systems facilitate deadlock recovery by implementing *checkpointing and rollback*. Checkpointing is saving enough state of a process so that the process can be restarted at the point in the computation where the checkpoint was taken. Autosaving file edits is a form of checkpointing. Checkpointing costs depend on the underlying algorithm. Very simple algorithms (like linear primality testing) can be checkpointed with a few words of data. More complicated processes may have to save all the process state and memory.

Checkpoints are taken less frequently than deadlock is checked for. If a deadlock is detected, one or more processes are restarted from their last checkpoint. The process of restarting a process from a checkpoint is called *rollback*. The hope is that the resource requests will not interleave again to produce deadlock.

Deadlock recovery is generally used when deadlocks are rare, and the cost of recovery (process termination or rollback) is low.

Process checkpointing can also be used to improve reliability (long running computations), assist in process migration (Sprite, Mach), or reduce startup costs (emacs).

