A

Seminar report

On

“Database Management System”

Submitted in partial fulfillment of the requirement for the award of degree
Of Bachelor of Technology in Computer Science

SUBMITTED TO:
www.studymafia.org

SUBMITTED BY:
www.studymafia.org

Preface

I have made this report file on the topic **Database Management System**; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

What is Database Management System?

- ✓ A Database Management System (DBMS), or simply a Database System (DBS) consist of :
    - A collection of interrelated and persistent data (usually referred to as the database (DB)).
    - A set of application programs used to access, update and manage that data (which form the data management system (MS)).
- ✓ The goal of a DBMS is to provide an environment that is both convenient and efficient to use in :
    - Retrieving information from the database.
    - Storing information into the database.

Brief History

- Early 1960s: first general purpose database by Charles Bachman from GE. Used the network data model.

- Late 1960s: IBM developed Information Management System (IMS). Used the hierarchical data model. Led to SABRE, the airline reservation system developed by AA and IBM. Still in use today.

- 1970: Edgar Code of IBM developed the relational data model. Led to several DBMS based on relational model, as well as important theoretical results. Code wins Turing award.
- 1980s: relational model dominant. SQL standard.
- Late 1980s, 1990s: DBMS vendors extend systems, allowing more complex data types (images, text).

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Purpose of DBMS

Data model

1. Data redundancy and inconsistency

- Same information may be duplicated in several places.
- All copies may not be updated properly.

2. Difficulty in new program to carry out each new task

3. Data isolation —

- Data in different formats.
- Difficult to write new application programs.
- files and formats

Security problems

Every user of the system should be able to access only the data they are permitted to see.

- E.g. payroll people only handle employee records, and cannot see customer accounts; tellers only access account data and cannot see payroll data.
- Difficult to enforce this with application programs.

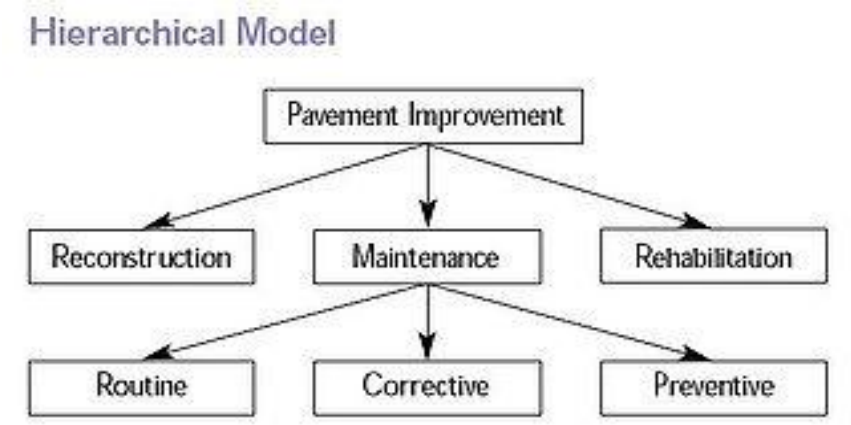
Integrity problems

- Data may be required to satisfy constraints.
- E.g. no account balance below $25.00.
- Again, difficult to enforce or to change constraints with the file-processing approach.

Hierarchical Model

The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1: N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths. For example, an organization might store information about an employee, such as name, employee number, department, salary.

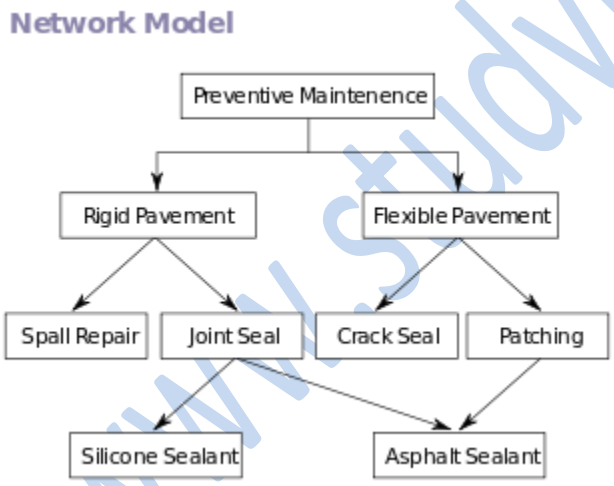
The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one too many. This restricts a child segment to having only one parent segment. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.



Network Model

The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type.

A member record type can have that role in more than one set; hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them. Thus, the complete network of relationships is represented by several pair wise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow). Usually, a set defines a 1:M relationship, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.



Relational Model

(RDBMS - relational database management system) A database based on the relational model developed by E.F. Code. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized in tables. A table is a collection of records and each record in a table contains the same fields.

Properties of Relational Tables:

- Values Are Atomic
- Each Row is Unique
- Column Values Are of the Same Kind
- The Sequence of Columns is Insignificant
- The Sequence of Rows is Insignificant
- Each Column Has a Unique Name

Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables. For example, an "orders" table might contain (customer-ID, product-code) pairs and a "products" table might contain (product-code, price) pairs so to calculate a given customer's bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables.

This can be extended to joining multiple tables on multiple fields. Because these relationships are only specified at retreival time, relational databases are classed as dynamic database management system. The RELATIONAL database model is based on the Relational Algebra.

Object-Oriented Model

Object DBMSs add database functionality to object programming languages. They bring much more than persistent storage of programming language objects. Object DBMSs extend the semantics of the C++, Smalltalk and Java object programming languages to provide full-featured database programming capability, while retaining native language compatibility. A major benefit of this approach is the unification of the application and database development into a seamless data model and language environment. As a result, applications require less code, use more natural data modeling, and code bases are easier to maintain. Object developers can write complete database applications with a modest amount of additional effort.

According to Rao (1994), "The object-oriented database (OODB) paradigm is the combination of object-oriented programming language (OOPL) systems and persistent systems. The power of the OODB comes from the seamless treatment of both persistent data, as found in databases, and transient data, as found in executing programs."

In contrast to a relational DBMS where a complex data structure must be flattened out to fit into tables or joined together from those tables to form the in-memory structure, object DBMSs have no performance overhead to store or retrieve a web or hierarchy of interrelated objects.

This one-to-one mapping of object programming language objects to database objects has two benefits over other storage approaches: it provides higher performance management of objects, and it enables better management of the complex interrelationships between objects. This makes object DBMSs better suited to support applications such as financial portfolio risk analysis systems, telecommunications service applications, World Wide Web document structures, design and manufacturing systems, and hospital patient record systems, which have complex relationships between data.

Semi structured Model

In semi structured data model, the information that is normally associated with a schema is contained within the data, which is sometimes called ``self-describing''. In such database there is no clear separation between the data and the schema, and the degree to which it is structured depends on the application. In some forms of semi structured data there is no separate schema, in others it exists but only places loose constraints on the data.

Semi-structured data is naturally modeled in terms of graphs which contain labels which give semantics to its underlying structure. Such databases subsume the modeling power of recent extensions of flat relational databases, to nested databases which allow the nesting (or encapsulation) of entities, and to object databases which, in addition, allow cyclic references between objects.

Semi structured data has recently emerged as an important topic of study for a variety of reasons. First, there are data sources such as the Web, which we would like to treat as databases but which cannot be constrained by a schema. Second, it may be desirable to have an extremely flexible format for data exchange between disparate databases. Third, even when dealing with structured data, it may be helpful to view it as semi structured for the purposes of browsing.

Architecture of DBMS

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

There are following three levels or layers of DBMS architecture:

1. External Level

2. Conceptual Level

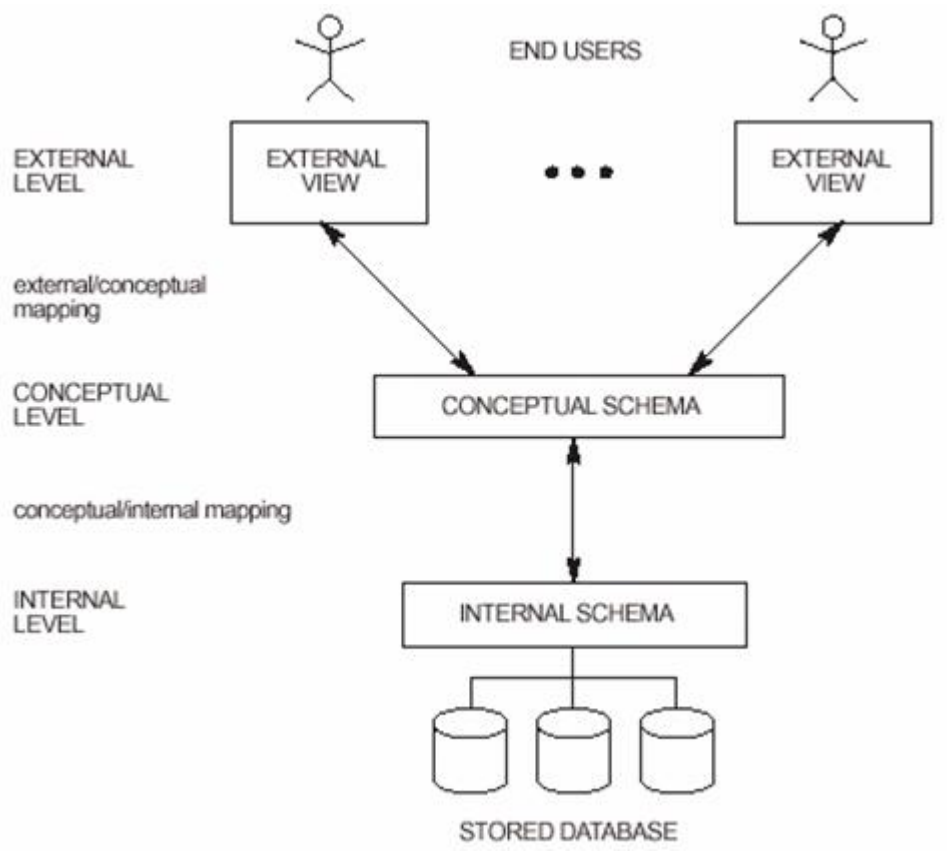
3. Internal Level

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

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

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

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



Components of DBMS:

- 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.
- People: Includes database designers, DBAs, application programmers, and end-users.
- Procedure: Instructions and rules that should be applied to the design and use of the database and DBMS.

Advantage of DBMS

- **Controlling Redundancy**

In file systems each application program has its own private files. In this case, the duplicated copies of the same data are created in many places. In DBMS, all data of an organization is integrated into a single database file. The data is recorded in only one place in the database and it is not duplicated.

- **Sharing of Data**

In DBMS, data can be shared by authorized users of the organization. The database administrator manages the data and gives rights to users to access the data. Many users can be authorized to access the same piece of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

- **Data Consistency**

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users. If the DBMS has controlled redundancy, the database system enforces consistency.

- **Integration of Data**

In Database management system, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

- **Integration Constraints**

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or they may be applied to relationships between records.

- **Data Security**

Form is very important object of DBMS. You can create forms very easily and quickly in DBMS. Once a form is created, it can be used many times and it can be modified very easily. The created forms are also saved along with database and behave like a software component. A form provides very easy way (user-friendly) to enter data into database, edit data and display data from database. The non-technical users can also perform various operations on database through forms without going into technical details of a fat abase.

- **Report Writers**

Most of the DBMSs provide the report writer tools used to create reports. The users can create very easily and quickly. Once a report is created, it can be used may times and it can be modified very easily. The created reports are also saved along with database and behave like a software component.

- **Control Over Concurrency**

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other. Most database management systems have sub-systems to control the concurrency so that transactions are always recorded with accuracy.

- **Backup and Recovery Procedures**

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damage due to failures to the computer system or application program. It is very time consuming method, if amount of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required.

- **Data Independence**

The separation of data structure of database from the application program that uses the data is called data independence. In DBMS, you can easily change the structure of database without modifying the application program.

Disadvantage

- **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.

DBMS Languages

Data Definition Language-DDL

Data Definition Language (DDL) statements are used to define the database structure or schema.

Some examples:

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Data Manipulation Language (DML)

Data Manipulation Language (DML) statements are used for managing data within schema objects.

Some examples:

- SELECT - Retrieve data from the a database
- INSERT - Insert data into a table
- UPDATE - Updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain
- MERGE - UPSERT operation (insert or update)
- CALL - Call a PL/SQL or Java subprogram
- EXPLAIN PLAN - explain access path to data
- LOCK TABLE - control concurrency