

A **Database Management System (DBMS)** is a set of software programs that controls the organization, storage, management, and retrieval of data in a database.

Examples of Database Management Systems include Oracle Database, Microsoft SQL Server, and PostgreSQL.

Categories of Database Management Systems (DBMS)

- a. **Relational Database Management System (RDBMS):** The Relational database management system organizes data in tabular files. E.g Oracle, Sybase, and Microsoft SQL Server. These databases support a standard language - SQL
- b. **Hierarchical Database Management System:** This category of database management system stores data in a tree-like structure.
- c. **Object-Oriented database management system:** Stores objects as opposed to tuples or records in a RDBMS.

Features of Database Management Systems (DBMS)

- a. **Query ability:** Querying is the process of requesting attribute information from various perspectives and combinations of factors.
- b. **Backup and replication:** A periodic copy of attributes created for a distant organization that cannot readily access the original.
- c. **Rule enforcement:** rules are applied to attributes so that the attributes are clean and reliable.
- d. **Security:** Often it is desirable to limit who can see or change which attributes or groups of attributes.
- e. **Computation:** Computations can be requested on attributes such as counting, summing, averaging, sorting, grouping, cross- referencing, etc.
- f. **Change and access logging:** Logging services allow this by keeping a record of access occurrences and changes.
- g. **Automated optimization:** If there are frequently occurring usage patterns or requests, some DBMS can adjust themselves to improve the speed of those interactions.

Database Servers

Database servers are computers that hold the actual databases and run only the database management system and related software.

Evolution of Database Management Systems

- Navigational DBMS
- Relational DBMS
- Multidimensional DBMS
- Object DBMS

Components of Database Management System (DBMS)

- a. Modeling Language: Serves to define the language of each database hosted via the DBMS. It the ability of the databases to communicate with the DBMS and thus operate on the system.
- b. Data Structures: Data structures enable DBMS to interact with the data without causing damage to the integrity of the data itself.
- c. Data Query Language: The data query language works with the data structures to make sure it is harder to input irrelevant data into any of the databases in use on the system.
- d. Transaction Mechanism: A database transaction mechanism ideally guarantees ACID properties in order to ensure data integrity despite concurrent user accesses (concurrency control),

DATA MODELS

Data models are a collection of conceptual tools for describing data, data relationships, data semantics and data constraints.

Object-Based Logical Models: Object-based logical models describe data at the conceptual and view levels. Categories of object-based logical models: Entity-relationship model, Object-oriented model, Binary model, Semantic data model, Infological model, Functional data model, etc.

Record-based Logical Models: Unlike object-oriented models, they are used to specify overall logical structure of the database, and provide a higher-level description of the implementation. Each record type defines a fixed number of fields with fixed length, or attributes.

Physical Data Models:

Physical data models are used to describe data at the lowest level. There are very few physical data models, e.g. Unifying model, Frame memory.

Object-Based Logical Models

A. **The E-R Model:** The entity-relationship model is based on a perception of the world as consisting of a collection of basic objects (entities) and relationships among these objects.

- **rectangles:** represent entity sets.
- **ellipses:** represent attributes.
- **diamonds:** represent relationships among entity sets.
- **lines:** link attributes to entity sets and entity sets to relationships.

B. **The Object-Oriented Model:** The object-oriented model is based on a collection of objects. An object contains values stored in instance variables within the object. These values are themselves objects. An object contains bodies of code that operate on the object. These bodies of code are called **methods**. Objects with same methods

and type of values are grouped into classes. An object can access the data of another object by invoking the method of that other object. This is called **sending a message** to the object.

Record-based Logical Models

The three most widely accepted models are the relational, network, and hierarchical.

A. **Relational model**: In the relational model, data and relationships are represented by a collection of tables. Each table has a number of columns with unique names

B. **The Network Model**: In the network model, data are represented by collections of records. Relationships among data are represented by links.

C. **The Hierarchical Model**: In this model organisation of the records is as a collection of trees.

INSTANCES AND SCHEMES

The information in a database at a particular point in time is called an **instance** of the database. While the overall design of the database is called the database **scheme**.

Categories of Schemes

There are several classes of schemes, corresponding to the levels of abstraction:

- Physical scheme
- Conceptual scheme
- Sub scheme (can be many)

Data independence refers to the ability to modify a scheme definition in one level without affecting a scheme definition in a higher level.

Classes of Data Independence

- **Physical data independence**: The ability to modify the physical scheme without causing application programs to be rewritten. Modifications at this level are usually to improve performance
- **Logical data independence**: The ability to modify the conceptual scheme without causing application programs to be rewritten. Usually done when logical structure of database is altered

Data Definition Language (DDL)

The data definition language (DDL) is a language used to specify a database scheme as a set of definitions expressed in a DDL.

Data Manipulation could refer to any of the following:

- **retrieval** of information from the database
- **insertion** of new information into the database
- **deletion** of information in the database
- **modification** of information in the database

A **Data Manipulation Language (DML)** is a language which enables users to access and manipulate data.

Types of Data Manipulation Language (DML)

- **procedural**: the user specifies what data is needed and how to get it
- **nonprocedural**: the user only specifies what data is needed

The term **Database Administrator** simply refers to a person having central control over data and programs accessing that data.

Duties of the Database Administrator

- **Scheme definition**: the creation of the original database scheme. This involves writing a set of definitions in a DDL (data storage and definition language), compiled by the DDL compiler into a set of tables stored in the data dictionary.
- **Storage structure and access method definition**:
- **Scheme and physical organisation modification**:
- **Granting of authorisation for data access**:
- **Integrity constraint specification**:

Database Users

The database users fall into several categories:

1. **Application programmers** are computer professionals interacting with the system through DML calls embedded in a program written in a host language (e.g. C, PL/1, Pascal). These programs are called application programs.
2. **Sophisticated users** interact with the system without writing programs. They form requests by writing queries in a database query language.
3. **Specialised users** are sophisticated users writing special database application programs.
4. **Naive users** are unsophisticated users who interact with the system by using permanent application programs (e.g. automated teller machine).

OVERALL SYSTEM STRUCTURE

Partitioning of Databases

Database systems are normally partitioned into modules for different functions.

Components of Data Structure

- **File manager** manages allocation of disk space and data structures used to represent information on disk.
- **Database manager** The interface between low-level data and application programs and queries.
- **Query processor** translates statements in a query language into low-level instructions the database manager understands.
- **DML precompiler** converts DML statements embedded in an application program to normal procedure calls in a host language. The precompiler interacts

with the query processor.

- **DDL compiler** converts DDL statements to a set of tables containing metadata stored in a data dictionary.

Data Structures for Physical Implementation

- **Data files:** store the database itself.
- **Data dictionary:** stores information about the structure of the database.
- **Indices:** provide fast access to data items holding particular values.

ENTITIES AND ENTITY SETS

What is an Entity?

An entity simply refers to an object that exists and is distinguishable from other objects.

Types of Entities

An entity may be concrete (a person or a book, for example) or abstract (like a holiday or a concept).

Entity Set

An entity set is a set of entities of the same type (e.g., all persons having an account in a particular bank).

Entity Representation

- An entity is represented by a set of **attributes** (function which maps an entity set into a domain.) as follows: E.g. name, S.I.N., street, city for —customerI' entity.
- The **domain** of the attribute is the **set** of permitted values (e.g. the telephone number must be seven positive integers).
- Consequently every entity is described by a set of (**attribute, data value**) pairs. Normally, there is one pair for each attribute of the entity set. For example, a particular customer entity is described by the set {(name, Harris), (S.I.N., 890-123- 456), (street, North), (city, Georgetown)}.

RELATIONSHIPS AND RELATIONSHIP SETS

Relationships: A relationship is a term which simply describes an association between several entities.

A **relationship set** is a set of relationships of the same type.

The **role** of an entity is the function it plays in a relationship.

Mapping Cardinalities: express the number of entities to which another entity can be associated via a relationship.

- **One-to-one:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
- **One-to-many:** An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.
- **Many-to-one:** An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.
- **Many-to-many:** Entities in A and B are associated with any number from each other.

Existence Dependencies: if the existence of entity X depends on the existence of entity Y, then X is said to be **existence dependent** on Y. (Or we say that Y is the **dominant entity** and X is the **subordinate entity**.)

The **primary key** of a relationship set **R** depends on the **mapping cardinality** and the presence of descriptive attributes.

Design of an E-R Database Scheme

Some of the decisions to be made are:

- Using a ternary relationship versus two binary relationships.
- Whether an entity set or a relationship set best fit a real-world concept.
- Whether to use an attribute or an entity set.
- Use of a strong or weak entity set.
- Appropriateness of generalisation.
- Appropriateness of aggregation.

THE RELATIONAL ALGEBRA

Relational algebra is a procedural query language. It can be defined formally by means of a basic expression consisting of either:

- a relation in the database
- a constant relation

Fundamental Operations

- $\sigma_p(E1)$ select (unary): The **Select** operation is responsible for selecting **tuples** that satisfy a given predicate. e.g $\sigma_{Place = 'Mumbai' \text{ or } Salary \geq 1000000} (Citizen)$
- $\Pi(E1)$ project (unary): **Project** copies its argument relation for the specified attributes only. e.g $\pi_{Name \ Salary} (Employees)$
- $\rho_x(E1)$ rename (unary): The **rename** operation solves naming problems which occur when performing the Cartesian product of a relation with itself. e.g $\rho_{MaleStudent(Sno, Name) \pi_{RollNo, SName}(\sigma_{Condition}(Student))}$
- $E1 \times E2$ cartesian product (binary): The **cartesian product** is a new relation with a tuple for each possible **pairing** of tuples from the set $E1 \times E2$
- $E1 \cup E2$ union (binary): returns the union (set union) of two compatible relations.

For a union operation to be legal, we require that:

- **r** and **s** must have the same number of attributes.
- The domains of the corresponding attributes must be the same.
- **E1 - E2** set-difference (binary): **r - s** results in a relation containing tuples that are in **r** but not in **s**.

SQL AND INTEGRITY CONSTRAINTS

Components of an SQL Expression

An SQL expression consists of select, from and where clauses.

The “SELECT” Clause

The **select** clause lists attributes to be copied. It corresponds to relational algebra project.

The “Where” Clause

The **where** clause corresponds to the selection predicate in a relational algebra. The predicates can involve the following:

- Logical connectives **and**, **or** and **not**.
- Arithmetic expressions on constant or tuple values.
- The **between** operator for ranges of values.

The “From” Clause

The **from** corresponds to Cartesian product, which lists relations to be used.

DATABASE MODIFICATION

Deletion

delete *from* **r** *where* **P**.

In the event that the **where** clause is omitted, all tuples are deleted.

Insertion

insert into account select * from account

Updates

update account set balance=balance * 1.06 where balance > 10,000

INTEGRITY CONSTRAINTS

Domain Constraints

A **domain** constraint specifies the permissible values for a given attribute, while a **key** constraint specifies the attributes that uniquely identify a row in a given table.

Domain/Key Normal Form (DKNF)

Domain/key normal form (DKNF) is a normal form used in database normalisation which requires that the database contains no constraints other than domain constraints and key constraints.

Domain Constraint Guidelines

- Attributes may have the same domain, e.g. cname and employee- name.

- Strong typing of domains enables one to test for values inserted, and whether queries make sense.

The **Check** Clause

The check clause in SQL-92 permits domains to be restricted in ways that most programming language type systems do not permit.

Referential Integrity: ensures that a value appearing in a relation for a given set of attributes also appears for another set of attributes in another relation.

Referential Integrity in SQL

Foreign Keys

A **foreign key** simply refers to the short form for declaring a single column.

primary key clause includes a list of attributes forming the primary key.

unique key clause includes a list of attributes forming a candidate key.

foreign key clause includes a list of attributes forming the foreign key, and the name of the relation referenced by the foreign key.

COMPUTER DATA STORAGE AND LEVEL

Computer data storage, often called **storage** or **memory**, refers to computer components and recording media that retain digital data used for computing for some **interval of time**.

Levels of Storage

There are different forms of storage, divided according to their distance from the central processing unit. The fundamental components of a general-purpose computer are arithmetic and logic unit, control circuitry, storage space, and input/output devices.

Primary Storage

Primary storage (or **main** memory or **internal** memory), commonly referred to as memory, is the only memory directly accessible to the CPU.

Secondary Storage

Secondary storage (also known as **external** memory or **auxiliary** storage) is not directly accessible by the CPU. The computer usually uses its i/o channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage is non-volatile, is two orders of magnitude less expensive than primary storage.

Types of Secondary Storage

- Hard disk drives

- **Rotating optical storage devices** such as CD and DVD drives

Some other examples of secondary storage technologies are: flash memory (e.g. USB flash drives or keys), floppy disks, magnetic tape, paper tape, punched cards, standalone RAM disks, and Iomega Zip drives.

Tertiary Storage

Tertiary storage involves a robotic mechanism which will mount (insert) and dismount removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use.

Off-line Storage

Off-line storage is computer data storage on a medium or a device that is not under the control of a processing unit. The medium is recorded, usually in a secondary or tertiary storage device, and then physically removed or disconnected.

FEATURES OF STORAGE TECHNOLOGIES

1. Volatility

a. **Non-Volatile Memory:** This is a form of memory that retains stored information even if it is not constantly supplied with electric power. It is suitable for long-term storage of information.

b. **Volatile Memory:** This form of memory requires constant power to maintain the stored information.

2. Differentiation

a. **Dynamic Random Access Memory:** A form of volatile memory which also requires the stored information to be periodically re-read and re-written, or refreshed, otherwise it would vanish.

b. **Static Memory:** A form of volatile memory similar to DRAM, that never needs to be refreshed as long as power is applied. (It loses its content if power is removed).

3. Mutability

a. **Read/Write Storage or Mutable Storage:** This form of storage allows information to be overwritten at any time.

b. **Read Only Storage:** Retains the information stored at the time of manufacture, and write once storage (Write Once Read Many) allows the information to be written only once at some point after manufacture.

c. **Slow Write, Fast Read Storage:** Read/write storage which allows information to be overwritten multiple times, but with the write operation being much slower than the read operation. Examples include CD-RW and flash memory.

4. Accessibility

a. **Random Access:** Any location in storage can be accessed at any moment in approximately the same amount of time. Suitable for primary and secondary storage.

b. **Sequential Access:** The accessing of pieces of information will be in a serial order, therefore the time to access a particular piece of information depends upon which piece of information was last accessed. Such characteristic is typical of off-line storage.

5. Addressability

- a. **Location-Addressable:** Each individually accessible unit of information in storage is selected with its numerical memory address.
- b. **File-Addressable:** Information is divided into files of variable length, and a particular file is selected with human-readable directory and file names.
- c. **Content-Addressable:** Each individually accessible unit of information is selected based on the basis of (part of) the contents stored there.

6. Capacity

- a. **Raw Capacity:** This is the total amount of stored information that a storage device or medium can hold. It is expressed as a quantity of bits or bytes (e.g. 10.4 megabytes).
- b. **Memory Storage Density:** This refers to the compactness of stored information. It is the storage capacity of a medium divided with a unit of length, area or volume (e.g. 1.2 megabytes per square inch).

7. Performance

- a. **Latency:** The time it takes to access a particular location in storage. The relevant unit of measurement is typically nanosecond for primary storage, millisecond for secondary storage, and second for tertiary storage.
- b. **Throughput:** The term “throughput” simply refers to the rate at which information can be read from or written to the storage. In computer data storage, throughput is usually expressed in terms of megabytes per second or MB/s, though bit rate may also be used.

8. Energy Use

2.5 inch hard disk drives often consume less power than larger ones. Low capacity solid-state drives have no moving parts and consume less power than hard disks. Also, memory may use more power than hard disks.

COMMON STORAGE TECHNOLOGIES

Semi Conductors: Semi conductor memory uses semi conductor-based integrated circuits to store information. A semiconductor memory chip may contain millions of tiny transistors or capacitors

Magnetic Storage: uses different patterns of magnetisation on a magnetically coated surface to store information. Magnetic storage is non-volatile.

Optical Storage: A typical optical disc, stores information on the surface of a circular disc and reads this information by illuminating the surface with a laser diode and observing the reflection. E.g CD, CD-ROM, DVD, BD-ROM

Magneto-Optical Disc Storage: This is a form of optical disc storage where the magnetic state on a ferromagnetic surface stores information. The information is read optically and written by combining magnetic and optical methods.

Paper Data Storage: typically in the form of paper tape or punched cards.

Information was recorded by punching holes into the paper or cardboard medium and was read mechanically (or later optically) to determine whether a particular location on the medium was solid or contained a hole.

FILE ORGANISATION

Introduction to File Organisation

File organisation: primarily refers to the logical arrangement of data (which can itself be organised in a system of records with correlation between the fields/columns) in a file system.

Two significant components of file organisation are:

- The way the internal file structure is arranged and
- The external file as it is presented to the operating system or program that calls it.

System File Organisation Specifics

1. Rapid access to a record or a number of records which are related to each other.
2. The adding, modification, or deletion of records.
3. Efficiency of storage and retrieval of records.
4. Redundancy, being the method of ensuring data integrity.

File Organisation Technique

a. **Sequential:** A sequential file contains records organised in the order they were entered. The order of the records is fixed. The records are stored and sorted in physical, contiguous blocks within each block the records are in sequence.

b. **Line-sequential:** Line-sequential files are similar to sequential files, except that the records can contain only characters as data.

c. **Indexed-sequential:** Key searches are improved by means of the indexed-sequential file organisation.

d. **Inverted list:** In an inverted list file organisation, the file is indexed on many of the attributes of the data itself.

e. **Direct or hashed access organisation:** In a direct or hashed access file organisation, a portion of the disk space is reserved and a —hashing algorithm computes the record address.

FUNDAMENTALS OF XML

What is XML?

XML is an extensible markup language for documents containing structured information.

XML Development Goals

1. It shall be straightforward to use XML over the Internet
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. XML shall be compatible with SGML.
5. The number of optional features in XML is to be kept to an absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly
8. The design of XML shall be formal and concise.
9. XML documents should be easy to create.
10. Terseness in XML markup is of minimal importance.

XML Pointer Language (XPointer) and XML Linking Language (XLink):

This defines a standard way to represent links between resources. In addition to simple links.

Extensible Style Language (XSL): The Extensible Style Language defines the standard style sheet language for XML.

XML DOCUMENTS

XML Markup and Content

- **Elements:** Elements are the most common form of markup. Delimited by angle brackets, most elements identify the nature of the content they surround.
- **Attributes:** Attributes are name-value pairs that occur inside start-tags after the element name.
- **Entity References:** entities are used to represent these special characters. Entities are also used to refer to often repeated or varying text and to include the content of external files. Entity references begin with the ampersand and end with a semicolon.
- **Comments:** Comments begin with <!-- and end with -->. Comments can contain any data except the literal string --.
- **Processing Instruction:** Processing instructions (PIs) are an escape hatch to provide information to an application.
- In a document, a CDATA section instructs the parser to ignore most markup characters.
- Document Type Declaration

DOCUMENT TYPE DECLARATION

Types of Declaration in XML

- **Element Type Declarations:** Element type declarations identify the names of elements and the nature of their content.

- **Attribute List Declarations:** Attribute list declarations identify which elements may have attributes, what attributes they may have, what values the attributes may hold, and what value is the default.
- Typical Attribute Types**
- CDATA attributes are strings, any text is allowed.
 - The value of an ID attribute must be a name
 - An IDREF attribute value must be that of a single ID attribute on some element in the document.
 - Name token attributes are a restricted form of string attribute. In general, an NMTOKEN attribute must consist of a single word.
 - A LIST OF NAMES: One can specify that the value of an attribute must be taken from a specific list of names.
- **Entity Declarations:** Entity declarations enable one associate a name with some other fragment of content.

Types of Entities

- **Internal Entities:** Internal entities associate a name with a string of literal text.
The XML specification predefines five internal entities:
 - ◆ < produces the left angle bracket, <
 - ◆ > produces the right angle bracket, >
 - ◆ & produces the ampersand, &
 - ◆ ' produces a single quote character (an apostrophe), '
 - ◆ " produces a double quote character, "
- **External Entities:** External entities associate a name with the content of another file.
- **Parameter Entities:** Parameter entities can only occur in the document type declaration. A parameter entity declaration is identified by placing % (percent-space) in front of its name in the declaration.

Instances where the declaration is required:

- **Authoring Environments:** Most authoring environments need to read and process document type declarations in order to understand and enforce the content models of the document.
- **Default Attribute Values:** If an XML document relies on default attribute values, at least part of the declaration must be processed in order to obtain the correct default values.
- **White Space Handling:** The semantics associated with white space in element content differs from the semantics associated with white space in mixed content.

INTRODUCTION TO WEB SERVICES

Advantages of Web Services

- Web Services are platform-independent and language- independent, since they use standard XML languages.

- Most Web Services use HTTP for transmitting messages (such as the service request and response).

Disadvantages of Web Services

- a. **Overhead.** Transmitting all your data in XML is obviously not as efficient as using a proprietary binary code.
- b. **Lack of versatility.** Currently, Web Services are not very versatile, since they only allow for some very basic forms of service invocation.

Web Services Architecture

- **Service Process:** This part of the architecture generally involves more than one Web service. For example, discovery belongs in this part of the architecture, since it allows us to locate one particular service from among a collection of Web services.
- **Service Description:** One of the most interesting features of Web Services is that they are self-describing.
- **Service Invocation:** Invoking a Web Service involves passing messages between the client and the server.
- **Transport:** Finally, all these messages must be transmitted somehow between the server and the client.