UNIT 1: Introductory to DBMS

A. Data & Information

Data: Raw facts which can be manipulated. Refers to what is *actually stored* in the database.

Information: Manipulation of data. Summarization of data in a presentable form. *Meaning* of the data as understood by some users.

- ❑ Data consists of facts, which become information when they are processed and convey some meaning to people.

B. Database

- ❑ Organized collection of facts.
- ❑ Collection of "***persistent***" data.
- ❑ Collection of Information arranged and presented to serve an assigned purpose.
- ❑ Collection of data that contains information relevant to an enterprise.
- ❑ Definition (C.J. Date):
  - ❑ A Database is a collection of persistent data that is used by the application systems of some given enterprise. Ex: Dictionary

Persistent data

- ❑ Once the data is entered in database it can subsequently be removed from the database only by some explicit request to the DBMS.
- ❑ Data which is not transient in nature. Transient data like input data, output data, work queues, software control blocks, SQL statements and intermediate results of processing data.

Databases touch all aspects of our lives.

C. Database Applications

- ❑ *Banking*: customer information, accounts, loans, all transactions.
- ❑ *Airlines*: reservations, schedule information.
- ❑ *Universities*: student information, course registration, grades.
- ❑ *Sales*: customers, products, purchase information.
- ❑ *Online retailers*: sales data + online order tracking, customized recommendations, online product evaluations.
- ❑ *Manufacturing*: supply chain management, tracking production of items, inventories of items & orders for items.
- ❑ *Human resources*: employee records, salaries, tax deductions, benefits, paycheck generation.
- ❑ *Credit card transactions*: purchases, generation of monthly statements.
- ❑ *Finance*: information about holdings, sales & purchases of bonds & stocks, storing real-time market data for online trading & automated trading.
- ❑ *Telecommunication*: keeping records of calls made, generating monthly bills, maintaining balances on pr`epaid cards, information about communication networks.

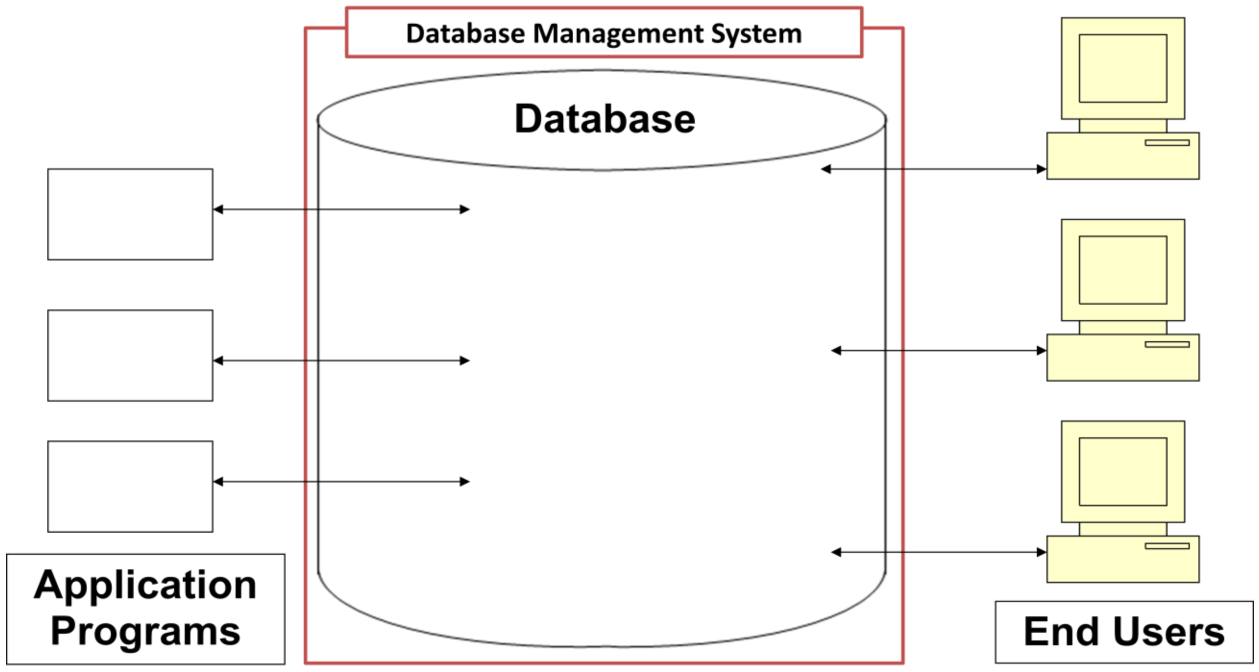
D. What is Database Management System (DBMS)?

DBMS contains information about a particular enterprise

- ❑ Collection of interrelated data & Set of programs to access those data.
- ❑ Primary goal of DBMS: to provide an environment that is both *convenient* and *efficient* to store & retrieve information to & from database.
- ❑ Basically a computerized record-keeping system.
- ❑ Overall purpose is to store information and allow the system users to retrieve and update that information on demand.
- ❑ Designed to manage large bodies of information.

Management of data involves:

- ❑ Defining structures for storage of information.
- ❑ Providing mechanisms for manipulation of information.
- ❑ Ensuring safety of the information stored against system crashes & unauthorized access.



DBMS is

- ❑ Software package: handles all access to the database.
- ❑ Collection of programs that enables users to create and maintain a database.
- ❑ It's a general purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases.
- ❑ Allows data to be effectively stored, retrieved and manipulated from database.
- ❑ Responsible for applying the authorization checks & validation procedures
- ❑ Data contained in a DBMS package can be accessed by multiple application programs & users.

E. Functions of DBMS:

- *Defining Databases*: Specifying the data types, structures and constraints of the data to be stored in the database.
- *Constructing*: Process of storing the data on some storage medium.
- *Manipulating*: Querying to retrieve specific data, updating the database, generating reports, changing database structure, etc.
- *Sharing*: Allows multiple users and programs to access database simultaneously.
- *Optimization and execution* (efficient way of executing a request).
- Provide *data security and integrity*.

- The database management system (DBMS) is the software that:
  - handles all access to the database
  - is responsible for applying the authorization checks and validation procedures
- Conceptually what happens is:
  - A user issues an access request, using some particular DML.
  - The DBMS intercepts the request and interprets it.
  - The DBMS inspects in turn the external schema, the external/conceptual mapping, the conceptual schema, the conceptual internal mapping, and the storage structure definition.
  - The DBMS performs the necessary operations on the stored database.

F. Characteristics of data stored in a database:

The data in a database should have the following features:

- *Shared*: Data stored in a database is shared among different users and application.
- *Persistence*: Data stored in a database exist permanently means that data can exist beyond the scope of the process that created it.
- *Validity/Integrity/Correctness*: Data should be correct with respect to the world entity that they represent.
- *Security*: Data should be protected from unauthorized access.
- *Consistency*: Whenever more than one data element in a database represents related real-world values, the values should be consistent with respect to the relationship.
- *Non-redundancy*: No two data items in a database should represent the same world entity. This means that data should not be duplicated.

A database management system has **four major components**:

1. Data - information held in an integrated, shared database
2. Hardware
3. Software
4. Users

1. Data in database system is mostly Integrated and Shared.

Integrated

- ❑ Unification/Association of several distinct files where any redundancy among those files, partly or wholly, is eliminated. Ex: Student and Class Enrollment files.

Shared

- ❑ Among different users. Many/Different users having access to same data, possibly at the same time (concurrent access).

2. Hardware

- ❑ Consists of secondary storage volumes (magnetic disks) on which the data lies.
- ❑ Hardware processor(s) & associated main memory for execution of database system s/w.
- ❑ Also consists of a processor, control units, etc.
- ❑ Data is assumed to be too big to be held completely in the processor's memory.

3. Software

- ❑ The DBMS (database management system) software allows one or many persons to access the data. DBMS software is also known as database manager / server.
- ❑ Allows the user to deal with data in an abstract (logical) way.
- ❑ Shields the database users from hardware level details.
- ❑ Includes utilities, report management, transaction manager, application development tools etc

4. Users: There are three broad classes of user:

- ❑ ***Application programmer***, responsible for writing programs in some high-level language such as java, etc. Programs can be traditional batch applications, or online applications which allow user to access the database interactively.
- ❑ ***End-User***, who accesses database via a query language.
- ❑ ***Database Administrator (DBA)***, who controls all operations on the database.

G. Users: can also be differentiated by the way they are expected to interact with system.

- ■ **Naive users** or **end users** – invoke one of the permanent application programs that have been written previously E.g. people accessing database over the web, bank tellers, clerical staff
- ■ **Application programmer** – interact with system through DML calls.
- ■ **Sophisticated users** – form requests in a database query language.
- ■ **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework.

Naive users or end users:

- ■ These users who interact with the system by prewritten application. Ex: bank teller who needs to transfer 50 Rs. then just call 'transfer' program.
- ■ User Interface: Forms & Reports → to submit a query user will just fill up the form.

- Users simply read reports generated by the Database.

Application programmer:

- Computer professionals who write application programs. Ex: programmers who create the application which interact with database.
- Creates the forms & reports using some type of tools like RAD (Rapid Application Development) Tool.
- They use predefined function to interact with Database

Sophisticated users:

- These users interact with the system without writing programs. Ex: users who directly work with database through DDL, DML Statements.
- They use queries for getting data by submitting them to query processor.
- They use tools like Online Analytical Processing (OLAP) and Data Mining.

Specialized users:

- Sophisticated users who write specialized database applications. It includes DBA. Ex: people who creates the application to manipulate data directly with database.
- They develop systems that store data with complex data types (e.g. images, audio data).
- Ex: computer-aided design systems, expert systems, environment-modeling

Users of the system can perform variety of operations

- ❑ Creation of new files in database. (Create)
- ❑ Inserting data into existing files. (Insert)
- ❑ Retrieving data from existing files. (Select)
- ❑ Deleting data from existing files. (Delete)
- ❑ Changing data in existing files. (Update)
- ❑ Removing existing files from the database. (Remove)

Database System Types: (depending on number of users)

- ❑ Single-user
  - Only One user can access the database.
  - Likely to be on small machines.
- ❑ Multi-user
  - Many users can access the database at the same time.
  - Likely to be on large systems.

H. Database Administrator

- ❑ A person who has central control over DBMS is called a database administrator (DBA).
- ❑ Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.
- ❑ The function of DBA (or the DBA duties) include:
  - Schema definition: Defining conceptual schema, Conceptual database design. DBA creates original database schema by executing a set of data definition statements in DDL.
  - Storage structure and access method definition: Internal schema definition.

- Schema and physical organization modification: carrying out changes to the schema and physical organization to reflect the changing needs of the organization & improve the performance.
- Granting user authority to access the database: specify security constraints, grant different types of authorities and regulate the user access to the database.
- Specifying integrity constraints, acting as liaison (mediator) with users.
- Perform routine maintenance activities like:
  - Periodically backup the data.
  - Ensuring free disc space for storage & upgrading disc space.
  - Monitoring performance and responding to changes in requirements

I. Purpose of DBMS

In the early days, database applications were built directly on top of file systems. Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems.

Drawbacks of using file-processing systems to store data:

- *Data redundancy and inconsistency*
  - Multiple file formats, duplication of information in different files
- *Difficulty in accessing data*
  - Need to write a new program to carry out each new task
- *Data isolation*
  - multiple files and formats
- *Integrity problems*
  - Integrity constraints (e.g. account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones
- *Atomicity of updates*
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- *Concurrent access by multiple users*
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance and updating it at the same time
- *Security problems*
  - Hard to provide user access to some, but not all, data

Database management systems offer solutions to all the above problems

J. Advantages of DBMS over typical file-processing systems:

- Data can be shared.
- Redundancy can be reduced.
- Inconsistency can be avoided (Out of 2 similar records stored in different places, 1is changed while other doesn't reflect that change. If a given fact is represented by a single entry than inconsistency cannot occur. Or propagate updates).
- Transaction support can be provided (transaction is a logical unit of database work, involving several database operations).
- Integrity can be maintained.
- Security can be enforced.
- Conflicting requirements can be balanced.
- Standards can be enforced.

K. Advantages of DBMS over traditional paper-based methods of record keeping:

- *Compactness*
  - No need of voluminous paper files.
- *Speed*
  - Machines can retrieve and update data faster than humans can.
- *Less hard work*
  - Maintaining files by hand is eliminated.
- *Currency*
  - Accurate, up-to-date information is available on demand at any time.
- *Protection*
  - Data is better protected against unintentional loss & illegal access.

Where do database systems come from?

- Commercial software vendors
  - Oracle, DB2, RDB, Ingres, Focus, etc. for mainframes
  - For Unix, Oracle, Ingres, Informix, etc.
  - Macs: Mainly Oracle. PCs: Access, Paradox, Oracle, etc.
- Is there more than one kind? Why?
  - Yes: flat file systems, network systems, hierarchical systems, relational systems, object-oriented systems.
  - And different brands for each type.
  - Why? History, needs and requirements of each type differ.
- Is there a standard? Yes,
  - Relational DBMS
  - SQL as the access language
    - DDL, DML, etc.

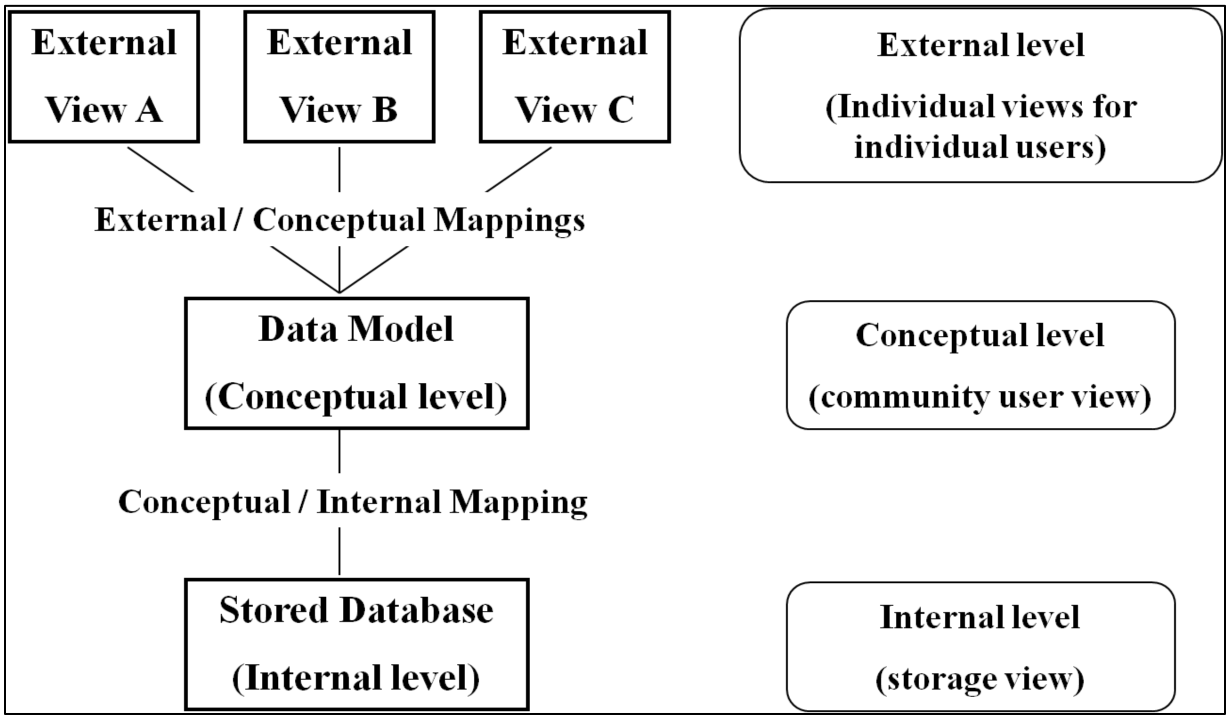
L. ANSI/SPARC Architecture

The **Architecture** for DBMSs is divided into three general levels:

- ❑ **External level (user level):** concerned with the way individual users see the data.
- ❑ **Conceptual level (community logical level):** can be regarded as a community user view - a formal description of data of interest to the organization, independent of any storage considerations.
- ❑ **Internal level (storage level):** concerned with the way in which the data is actually stored.

The **Three-level Architecture** forms the basis of modern database architectures.

- ■ This is in agreement with the ANSI/SPARC study group on DBMS.
- ■ **ANSI/SPARC** is the American National Standards Institute/Standard Planning and Requirement Committee.


External View A
External View B
External View C
External level (Individual views for individual users)
External / Conceptual Mappings
Data Model (Conceptual level)
Conceptual level (community user view)
Conceptual / Internal Mapping
Stored Database (Internal level)
Internal level (storage view)

External Level:

Concerned with the way in which data is viewed by individual users.

- ❑ Closest to End User.
- ❑ Individual users are given different views according to the user's requirement.
- ❑ Same database but – **diff views for diff users**.
- ❑ Insulates users from the conceptual and internal levels.
- ❑ Also known as **View Level**.
- ❑ Each user sees the data in terms of an external view
  - ■ Defined by an external schema, consists of external record descriptions, and understands the mapping between external schema and the conceptual level.

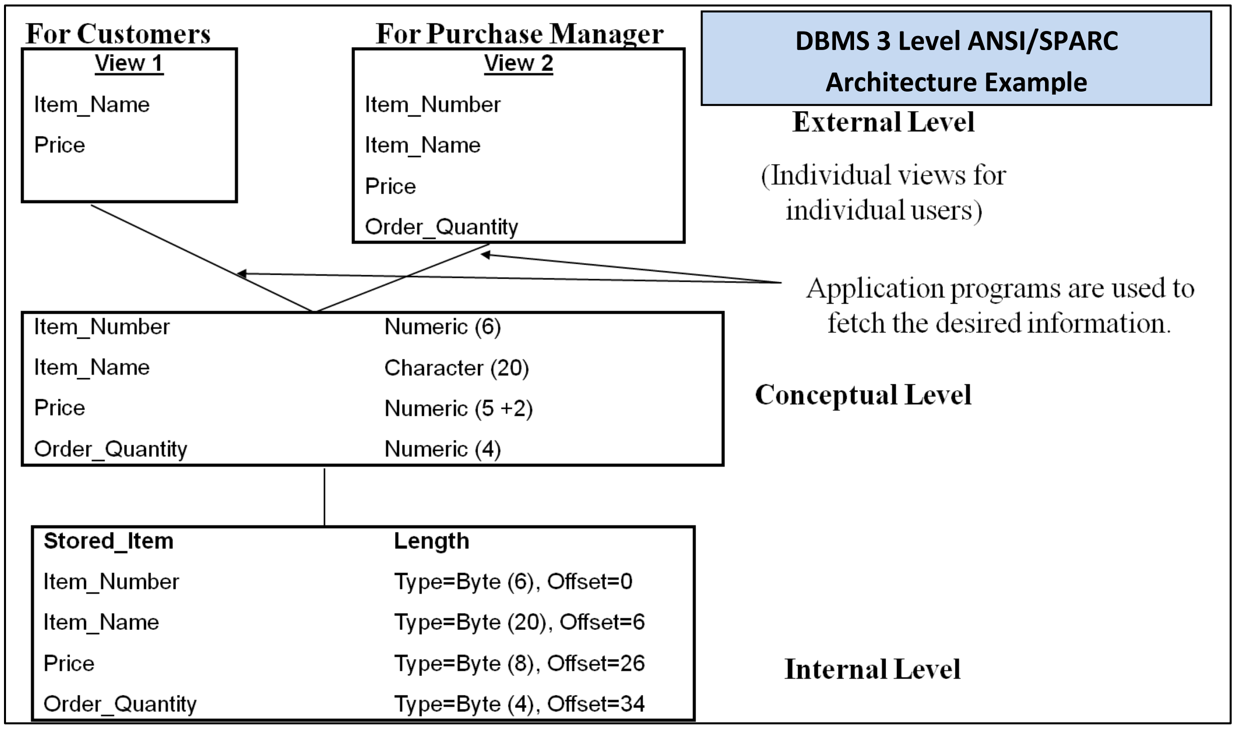
- A user is anyone who needs to access some portion of the data.
  - Access via programming lang. like Java, etc (**Programmer**) or a query language (**Casual user**).
  - All access methods include a **Data Sub-Language (DSL).**
- A DSL is a combination of two languages:
- **Data Definition Language** (**DDL**) –
  - Definition and description. A language in which the storage structure and access methods used by the database system are specified.
- **Data Manipulation Language** (**DML**) –
  - Manipulating data. A Language for accessing and manipulating the data organized by the appropriate data model. Also known as query language.
- Two classes of languages
  - **Procedural** – user specifies what data is required and how to get those data.
  - **Nonprocedural** – user specifies what data is required without specifying how to get those data.

Conceptual Level:

- Concerned with what type of data is stored in database and relationships among data items.
- Users – concerned with the **business logic** of DB.
- Also known as **Logical Level**.
- It is in general a view of the data as it actually is.
- It helps to decide –
  - Data types required for different data stored in DB.
  - How these data items must be related with one another.
  - How the changes made in one data item effects other items.
- The conceptual schema, as well as definitions, contains authorization and validation procedures.

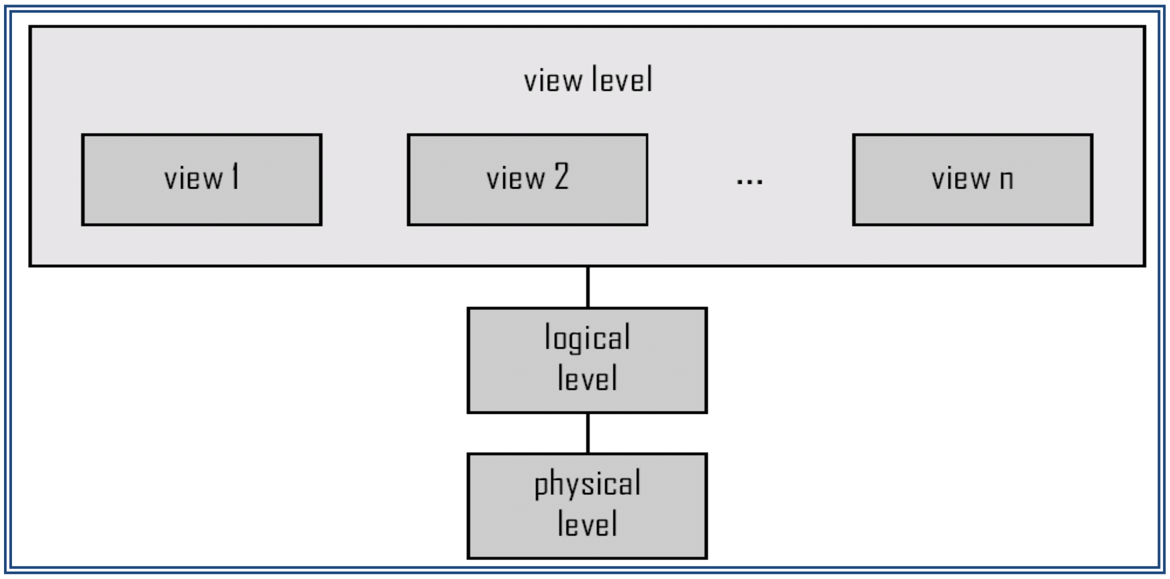
Internal Level:

- This is a very low-level representation of the entire database.
- Closest to the physical aspects of data storage.
- Refers to how data is actually stored on the physical storage medium like Hard discs, etc.
- Also known as the **Physical Level**.
- Internal level is described by the internal schema as:
  - defines the various types of stored records
  - what indices exist
  - how stored fields are represented
  - what physical sequence the stored records are in.
- In effect, the internal schema is the storage definition structure.


For Customers
View 1
Item_Name
Price
For Purchase Manager
View 2
Item_Number
Item_Name
Price
Order_Quantity
DBMS 3 Level ANSI/SPARC Architecture Example
External Level
(Individual views for individual users)
Application programs are used to fetch the desired information.
Item_Number Numeric (6)
Item_Name Character (20)
Price Numeric (5 +2)
Order_Quantity Numeric (4)
Conceptual Level
Stored_Item Length
Item_Number Type=Byte (6), Offset=0
Item_Name Type=Byte (20), Offset=6
Price Type=Byte (8), Offset=26
Order_Quantity Type=Byte (4), Offset=34
Internal Level

M. Views of Data

- ❑ Major purpose of a DBMS →is to provide users with and abstract view of the data.
- ❑ System hides the details of how the data are stored & maintained.
- ❑ Many DBMS users are not computer trained.
- ❑ To simplify users' interactions with DBMS the complexity is hidden from users through several levels of abstraction:



Three levels of data abstraction

- **Physical level:**
  - The lowest level of abstraction
  - Describes *how* the data are actually stored.
  - It describes complex level data-structures in detail.
  - It describes how a record (Ex: record for a customer) is stored in terms of consecutive storage locations for example, words or bytes.
- **Logical level:**
  - Next-higher level of abstraction.
  - Describes *what* data are stored in database & what relationships exist among those data.
  - Programmers and Database administrators work at this level.
  - Programmers use any programming language to access the data from database.
  - For ex: in a Pascal like language we may declare a record as follows:

```
type customer = record
        customer_id       :     string;
        customer_name     :     string;
        customer_street   :     string;
        customer_city : string;
end;
```

- **View level:**
  - Highest level of abstraction.
  - Describes only part of the entire database.
  - Many users do not need to access all the information stored in database, but,
  - Users need only a part of the database.
  - System may provide many views for the same database.
  - Ex: Application programs hide details of data types.
  - Views can also hide information (such as an employee's salary) for security purposes.

N. Schemas & Instances

- Similar to types(Schema) & variables(Instances) in programming languages
- **Instance:** Actual contents of database at a particular point in time.
  - The collection of information stored in the database at a particular moment is called an *Instance* of the database**.**
  - e.g. Value of the variable in the program
- Instances change frequently.
- **Schema:** The overall design of the database is called *Schema*.
  - An outline or plan that describes the records & relationships existing at particular level. The logical structure of a database (e.g., set of customers and accounts and the relationship between them).
  - e.g. Declaration of the variable
- Schema does not change frequently.
- Data views at each of the 3 levels are described by ***schema***.

- External level: defined by means of External Schema.
- Similarly Conceptual level & Internal level both are defined by means of Conceptual and Internal schemas respectively.
- Schema also describes the way in which data elements at one level can be mapped to the corresponding data elements in the next level.
- **Physical schema** => design of the database in physical level is called a physical schema.
- **Logical schema** => design of the database in logical level is known as logical schema
- **View schema** => design of the database in view level is known as view schema. This schema is also known as "subschema".

O. Mappings

- **Conceptual/internal mapping:** defines conceptual and internal view correspondence.
  - specifies mapping from conceptual records to their stored counterparts
  - A change to the storage structure definition means that the conceptual/internal mapping must be changed accordingly, so that the conceptual schema may remain invariant, achieving physical data independence.
- **External/conceptual mapping:** defines external and conceptual view correspondence.
  - A change to the conceptual definition means that the conceptual/external mapping must be changed accordingly, so that the external schema may remain invariant, achieving logical data independence.

P. Data Independence

- Ability to modify a schema definition in one level without affecting a schema definition in the next higher level.
- The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

  Two levels of data independence:
- **Physical data independence**
  - Capability to change the internal schema without having to change the logical schema external schemas, or application schema.
  - User only needs to know what he wants and need not worry about how the data is obtained or stored at physical level.
  - Execution of ad hoc requests and application programs is not affected by changes in the physical data access and storage methods
- **Logical data independence**
  - Logical data independence is the capability to change the logical schema without having to change the external schemas or application programs.
  - E.g. adding an attribute or column to the base table should not disturb the programs.
  - Logical changes in tables & views such as adding/deleting columns or changing field lengths does not require modifications in programs or in the format of ad hoc requests.

Q. Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- A collection of concepts that can be used to describe the structure of a database (description of the organization of a database).
- Provides necessary means to achieve data abstraction.
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

R. Overall Database System Architecture

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. A gigabyte is 1000 megabytes (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes). Main memory of computers cannot store this much information, so the information is stored on disks. Data are moved between disk storage and main memory by storage manager as and when needed.

The query processor is important because it helps the database system simplify and facilitate access to data. High-level views help to achieve this goal. Users of the system do not have to remember unnecessarily the physical details of the implementation of the system. It is the job of the database system to translate updates and queries written in a nonprocedural language into an efficient sequence of operations.

- **Storage manager:**
  1. A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
  2. It is responsible for:
     - interaction with the file manager
     - efficient storing, retrieving and updating of data

  – Translating various DML statements into low-level file-system commands

Storage manager components include

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.
-

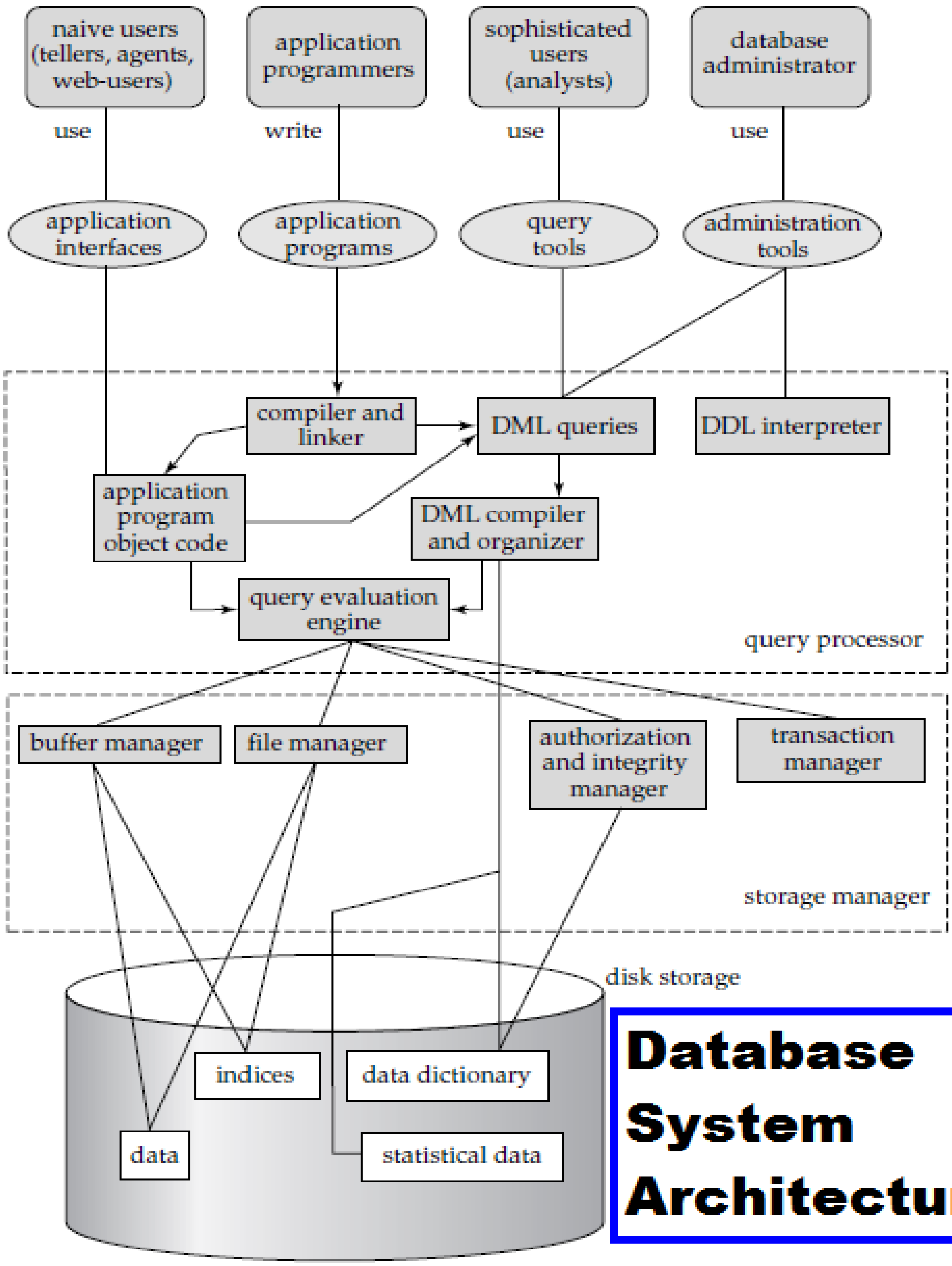
The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which provide fast access to data items that hold particular values.

❑ **The Query Processor:**

The query processor components include

- **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.
- **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.



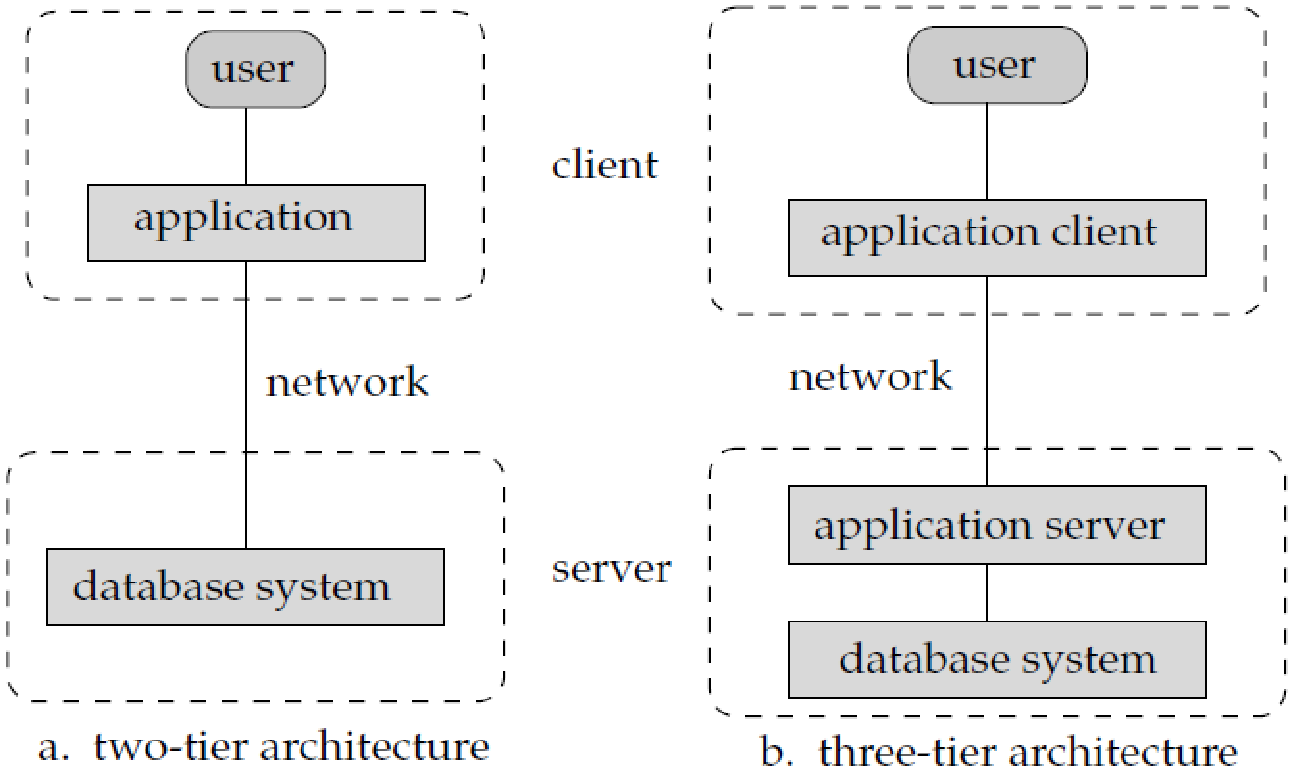
Database System Architecture

DBMS Application Architectures

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between **client** machines, on which remote database users work, and **server** machines, on which the database system runs.

Database applications are usually partitioned into two or three parts:

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



In contrast, in a **three-tier architecture**, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an **application server**, usually through a forms interface. The application server in turn communicates with a database system to access data. The **business logic** of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.