Database Management System

UNIT – I

Suggested Reading:

- Database Concepts by Korth, Silbertz, Sudarsharn
- Fundamentals of Database Systems by Elmasri, Navathe
- An Introduction to Database System by Date. C.J

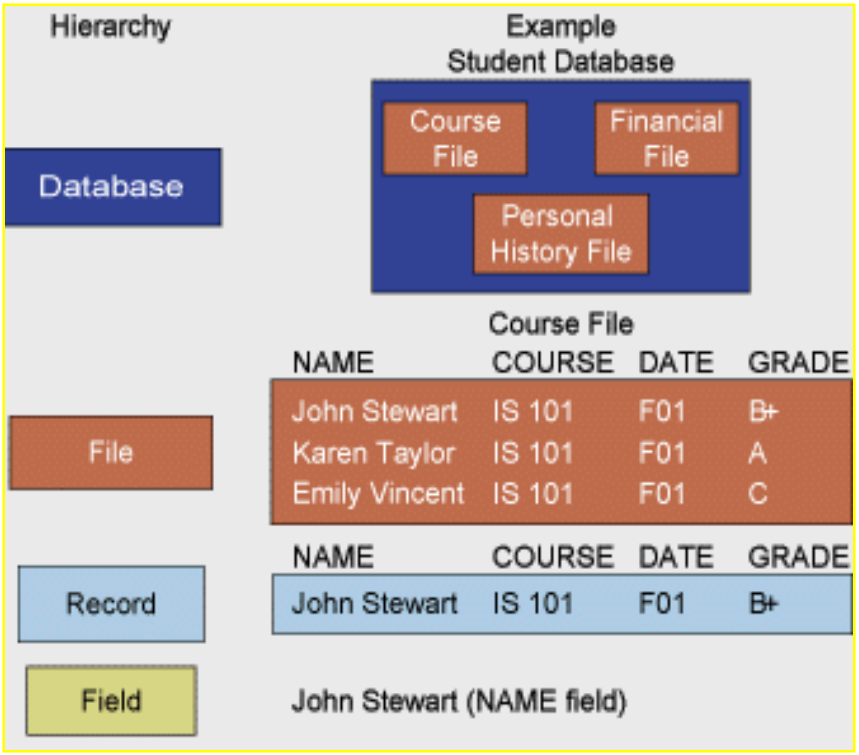
Note: Go through NPTEL videos for DBMS lectures

What is a Database System?

- **Database**: A collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Database Management System (DBMS)**: A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**: The DBMS software together with the data itself. Sometimes, the applications are also included.

Database Organization – Basic Terms

- **Database:** Group of related files
- **File:** Group of records of same type
- **Record:** Group of related fields
- **Field:** Group of words or a complete number
- **Byte:** Group of bits that represents a single character
- **Bit:** Smallest unit of data; binary digit (0,1)



- **Database Applications:**
  - Banking: all transactions
  - Ticket Reservation System (Airlines/Railways/Bus/Cabs)
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
  - Social networking sites

and many more……

File Processing System

- File processing systems was an early attempt to computerize the manual filing system. A file system is a method for storing and organizing computer files and the data they contain to make it easy to find and access them.
- File systems may use a storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files.
- The manual filing system works well when the number of items to be stored is small. It even works quite adequately when there are large numbers of items and we have only to store and retrieve them.

Characteristics of File Processing System

- It is a group of files storing data of an organization.
- Each file is independent from one another.
- Each file contained and processed information for one specific function, such as accounting or inventory.
- Files are designed by using programs written in programming languages such as COBOL, C, C++.
- The physical implementation and access procedures are written into database application; therefore, physical changes resulted in intensive rework on the part of the programmer.
- As systems became more complex, file processing systems offered little flexibility, presented many limitations, and were difficult to maintain.

Drawbacks of using file 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

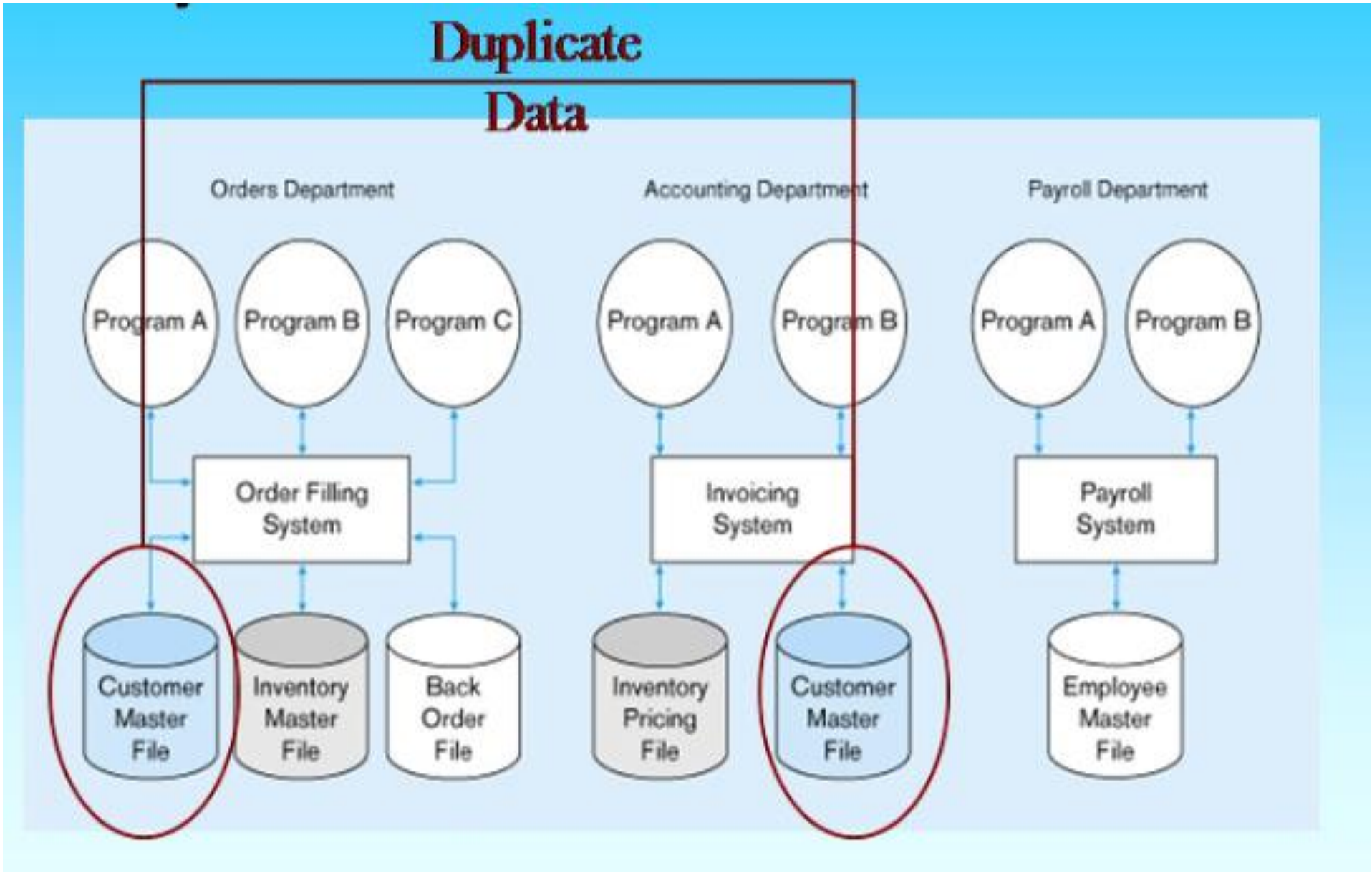
Limitations of File Organization Method

- ✓Hard to add new constraints or change existing ones
- ✓Lack of flexibility
- ✓Poor security

Disadvantages of File Processing

- **Program-Data Dependence**
  - All programs maintain metadata for each file they use
- **Duplication of Data**
  - Different systems/programs have separate copies of the same data
- **Limited Data Sharing**
  - No centralized control of data
- **Lengthy Development Times**
  - Programmers must design their own file formats
- **Excessive Program Maintenance**
  - 80% of of information systems budget

Fig: File Processing System at Furniture Company



Problems with Data Redundancy

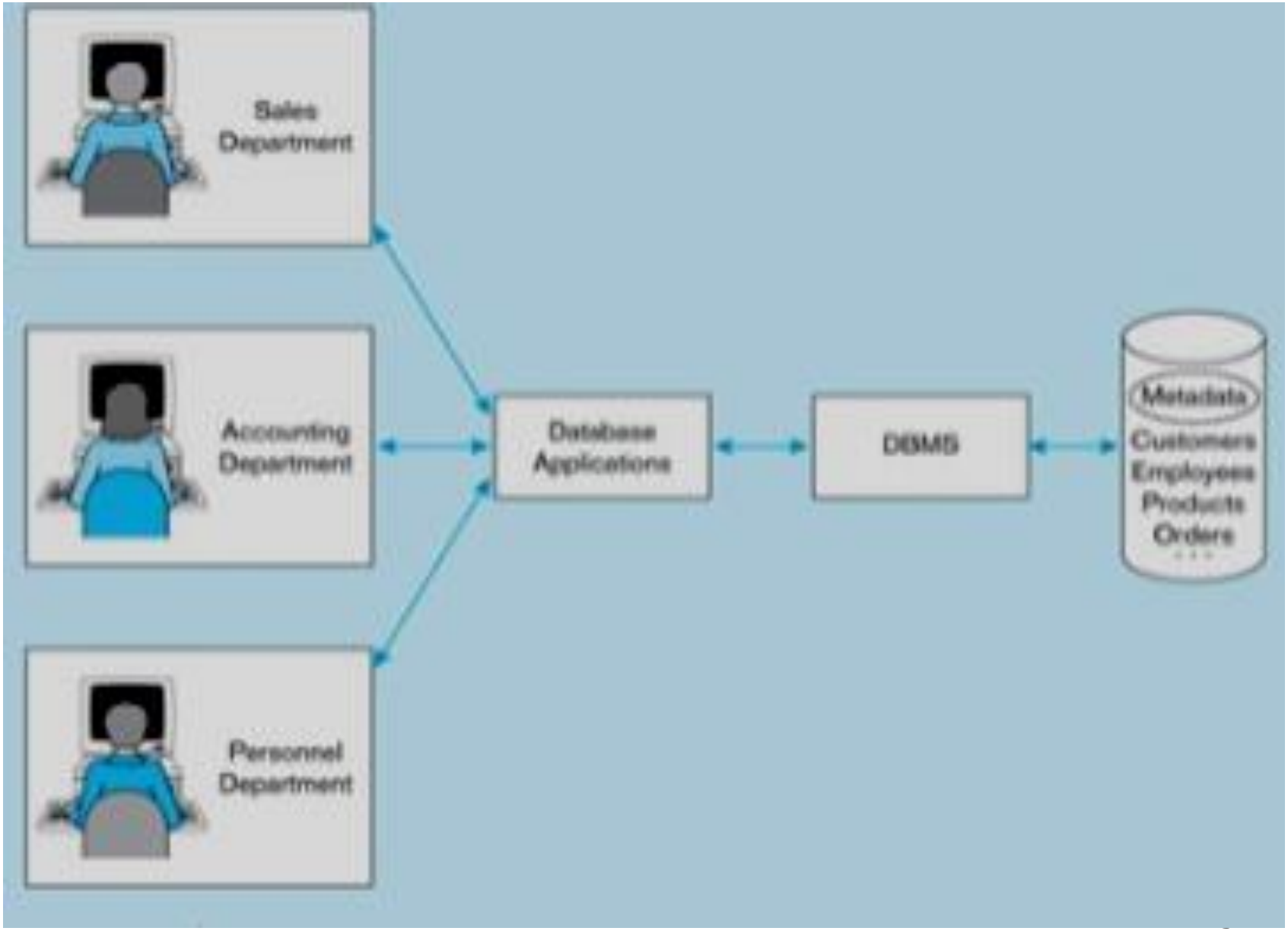
- Waste of space to have duplicate data
- Causes more maintenance headaches
- The biggest problem:
  - **When data changes in one file, could cause inconsistencies**
  - Compromises ***data integrity***

SOLUTION:
The DATABASE Approach

- Central repository of shared data
- Data is managed by a controlling agent
- Stored in a standardized, convenient form

Requires a Database Management System (DBMS)

Database Management System



Definition of DBMS

- **A Database Management System** is a framework that enables users to create and maintain a database. It is a general purpose software system that facilitates processes of defining, constructing and manipulating databases for various applications.

Advantages of Database approach:

- **Controlling Redundancy**
- **Restricting Unauthorized access**
- **Providing persistent storage for program objects and data structures**
- **Providing multiple user interface**
- **Representing complex relationships among data**
- **Enforcing integrity constraints and providing backup and recovery**

Database System Concepts & Architecture

Architecture of Database Systems

The architecture of a DBMS can be examined from several angles:

✓the functional architecture that identifies the main components of a DBMS,

✓the application architecture that focuses on application uses of DBMSs, and

✓the logical architecture that describes various levels of data abstractions.

Functionally, a DBMS contains several main components like:

- **The memory manager**
- **The query processor**
- **The transaction manager**

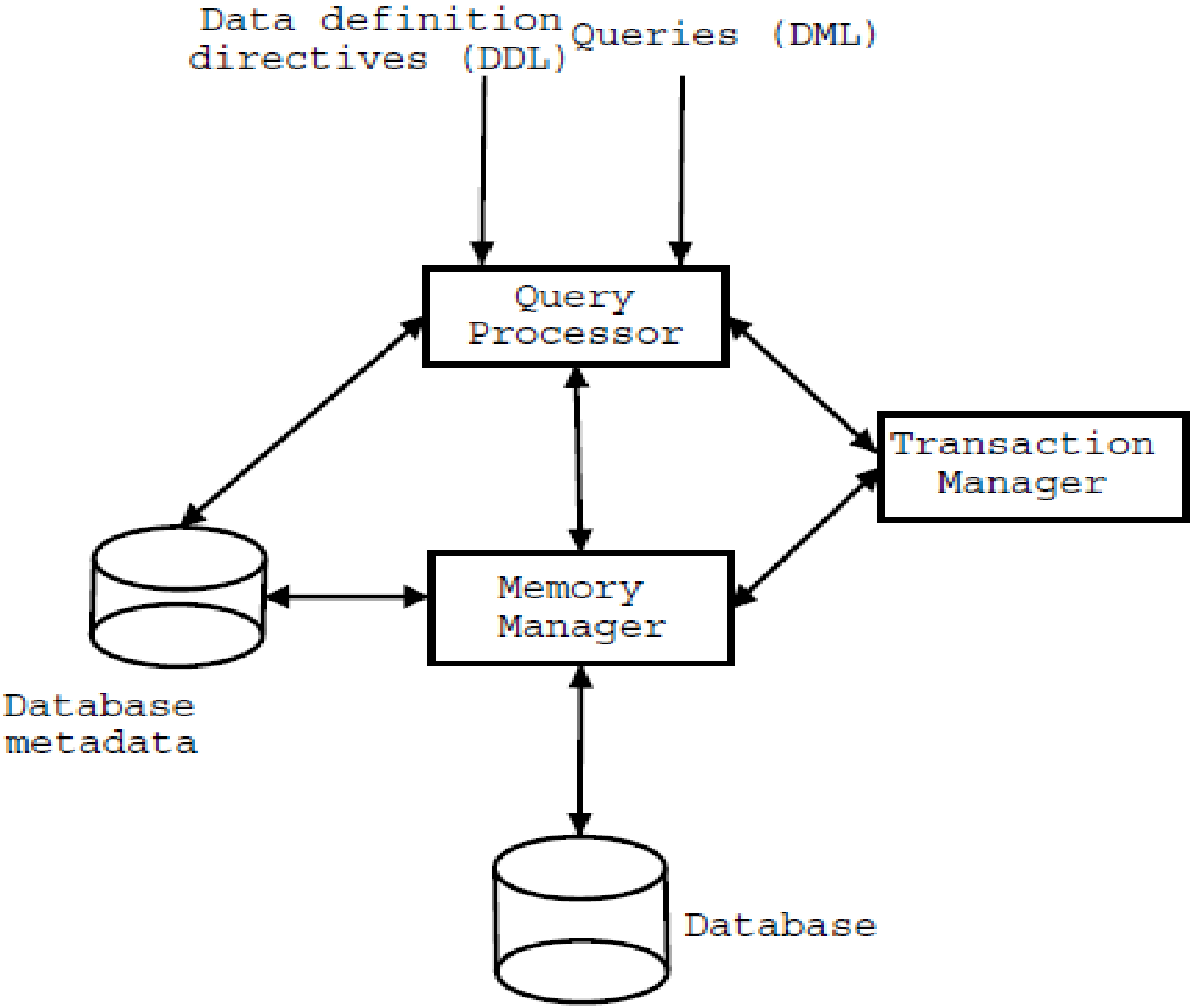


Figure 1.1: Functional Architecture of DBMSs

Three schema architecture of Database

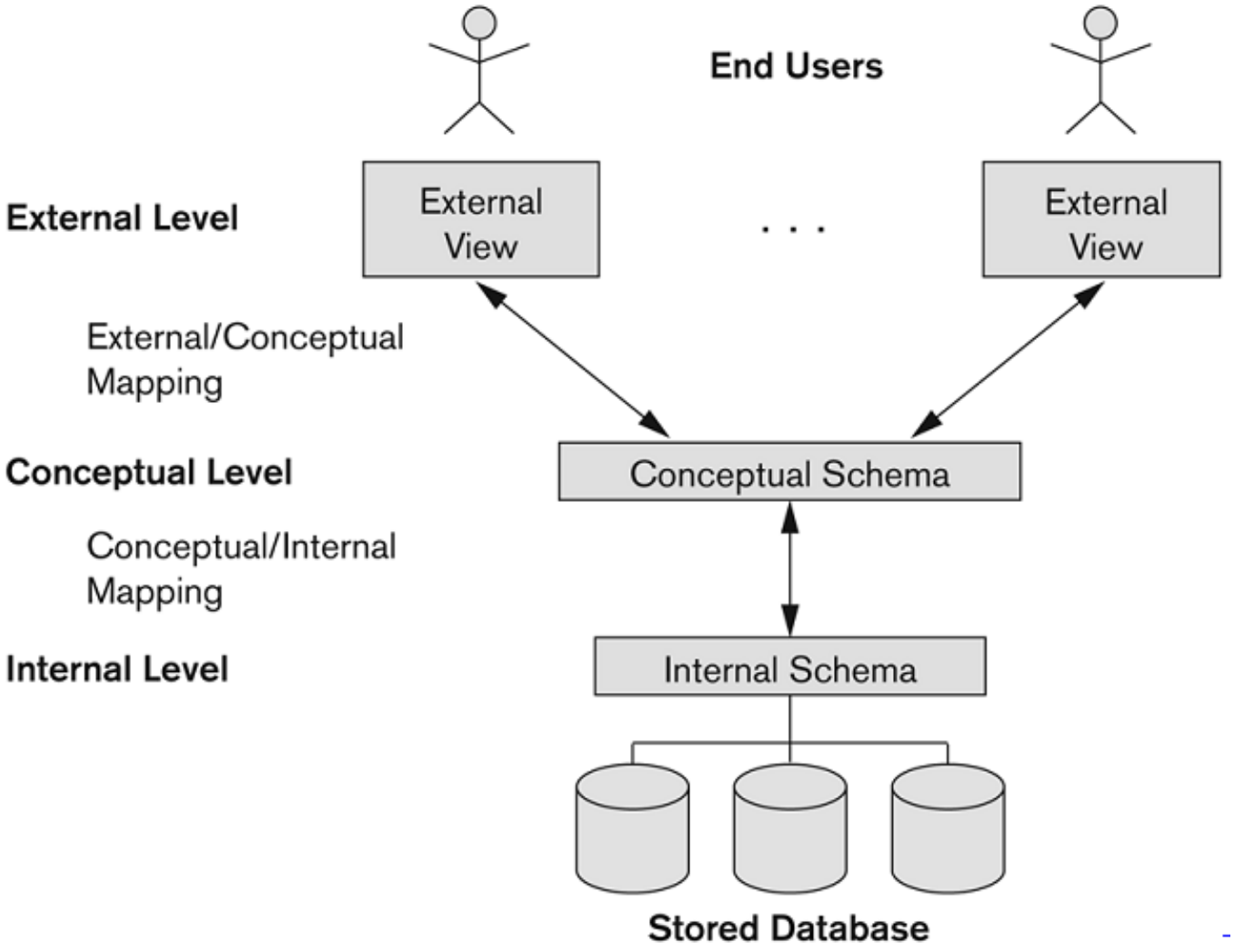
The goal of Three-Schema architecture is to separate the user applications and physical database. In this architecture, schemas can be defined at the following three levels:

•**The internal or physical level**:- The internal level has an internal schema which describes the physical storage structure of the database.

•**The conceptual or logical level**:-The conceptual level has a conceptual schema, it describes the entities, data types, relationships, user operations, and constraints.

The external level or view level:- The external or view level includes a number of external schemas or user views. It describes the part of the database that a particular user group

The three-schema architecture



Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database- provides the necessary means to achieve the abstraction, and certain *constraints* that the database should obey.

- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model. Operations on the data model may include *basic operations* and *user-defined operations*.

Categories of data models

- **Conceptual** (**high-level**) data models: Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.). This model uses concepts such as entities, attributes, and relationships

- **Physical** (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored in the computer.

- **Implementation** (**representational**) data models: Provide concepts that may be understood by end users . ***This fall between the above two***. Representational data model ***hide some details*** of data storage. These models are used frequently in traditional commercial DBMSs. They use legacy data models- network and hierarchical models. This model represent data by using record structures and some times called ***record-based models***.

Categories of data models

- In software engineering, an **entity-relationship model** (**ERM**) is an abstract and conceptual representation of data.

- Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system.

- It is drawn in top-down fashion.

History of Data Models

A database model defines the manner in which the various files of a database are linked together. The commonly used database model are:

1. Hierarchical Database.
2. Network Database.
3. Relational Database.
4. Object Oriented Database.

HIERARCHICAL DATABASE MODEL

The data element are linked in form of inverted tree structure.

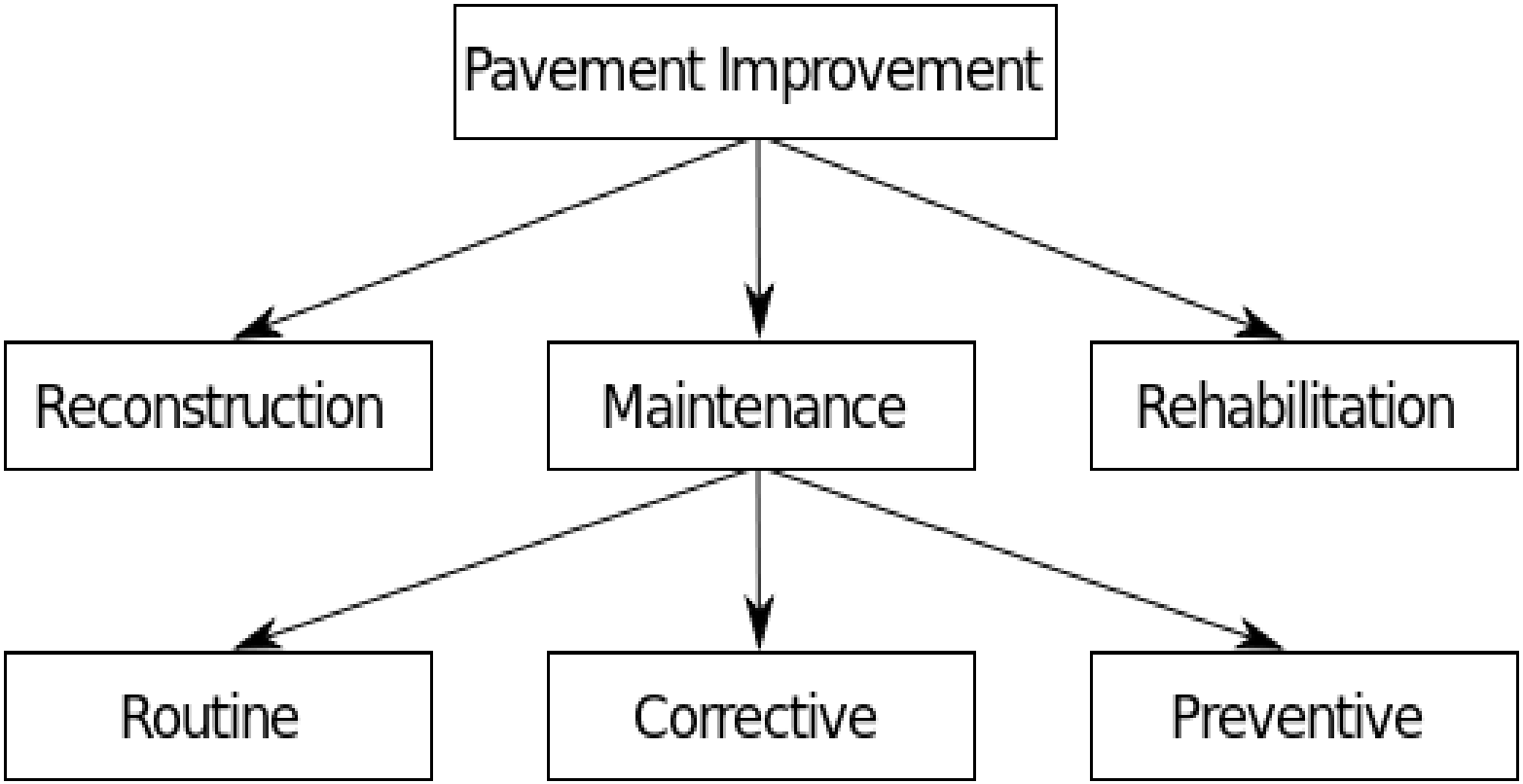
- Root at the top.
- Parent-child relationship
- Parent data element is one and can have one or more subordinate or child element.
- There may be many child but only one parent data element.

HIERARCHICAL DATABASE MODEL

The data element of many applications can be neatly organised with this model

- The main limitation is, it does not support flexible data access because the data can only be accessed by following the tree structure.
- Hence the mapping of data and their relationship in tree structure is very important when the database is first design.

Hierarchical Model

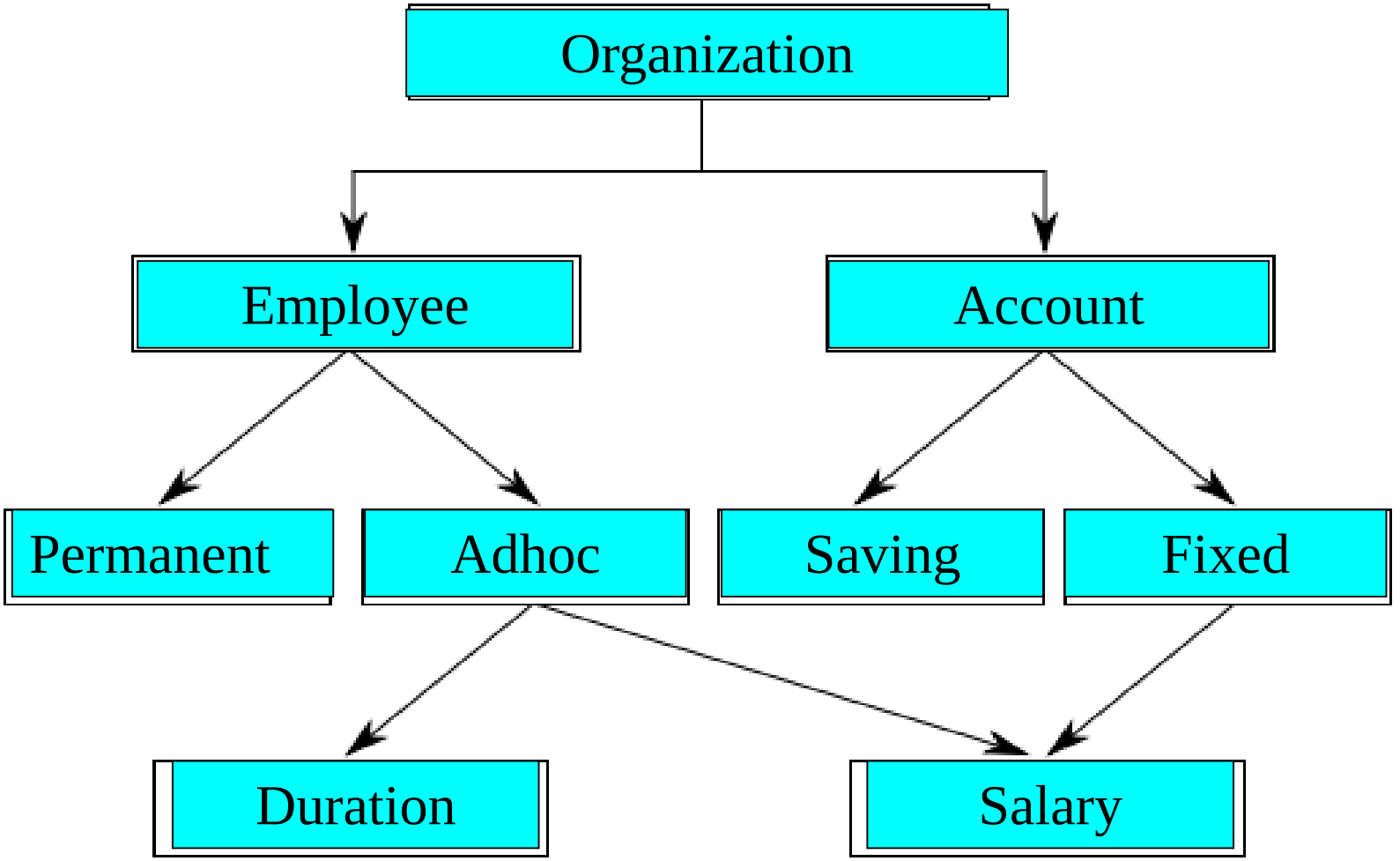


NETWORK DATABASE MODEL

It is extension of Hierarchical Database model.

- It follows parent-child relationship.
- the mapping of data and their relationship is very important when the database is first design.
- Here the child data can have more than one parent and can have no parent at all.
- Here the extraction of information can be from any data element in database structure instead of root data element.

Network Model



RELATIONAL DATABASE MODEL

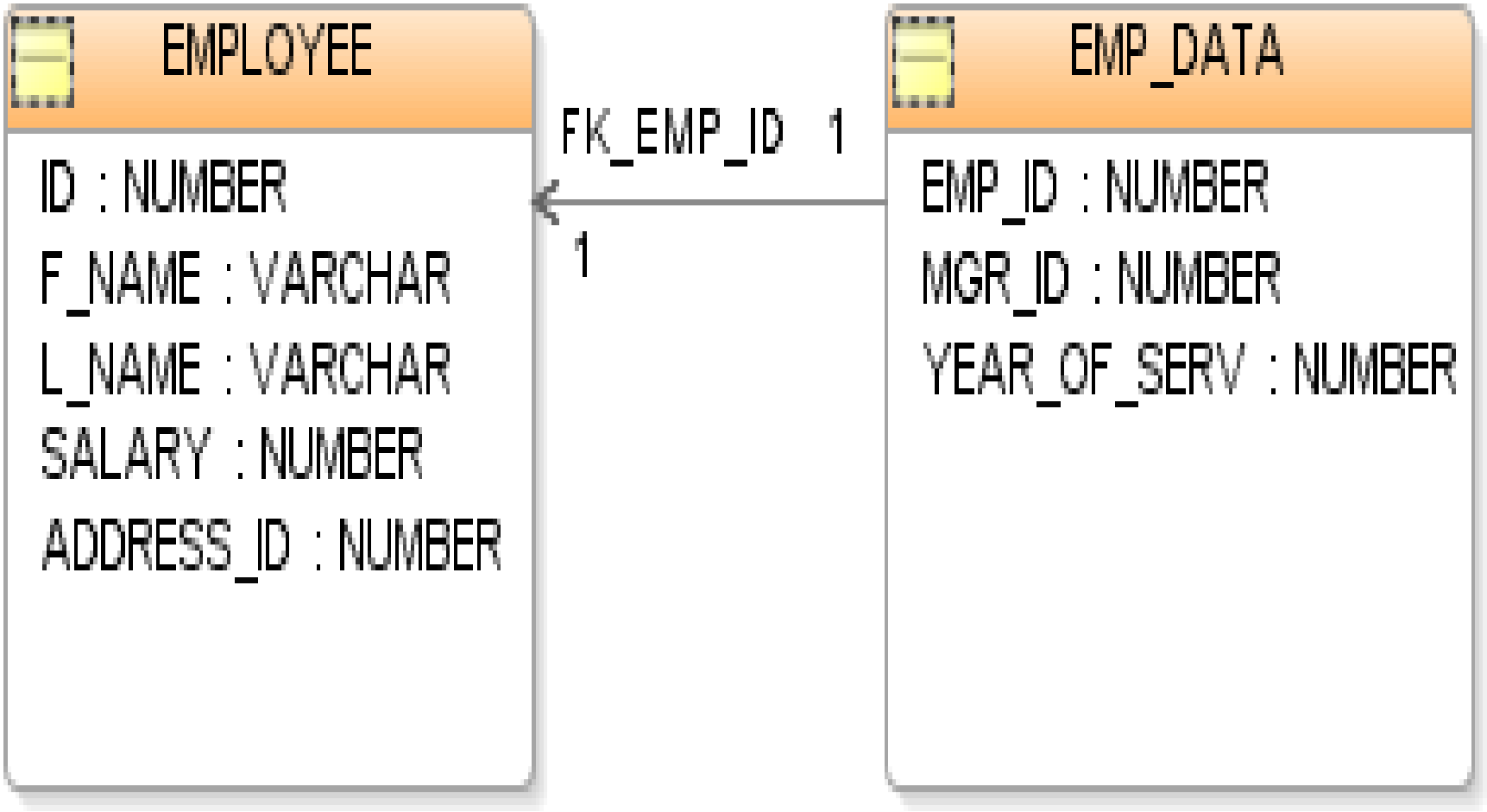
Here the data element are organised in form of multiple tables with rows and columns.

- Each table represents separate file.
- Each table column are represent as field.
- Each table row are represent as data record.
- The data in one table are related to data in another table by common field.

It provide grater flexibility in data organisation and future enhancement.

- If new data is to be added then it is not necessary to redesign the database rather new table can be easily added.

Two Tables with Relationship

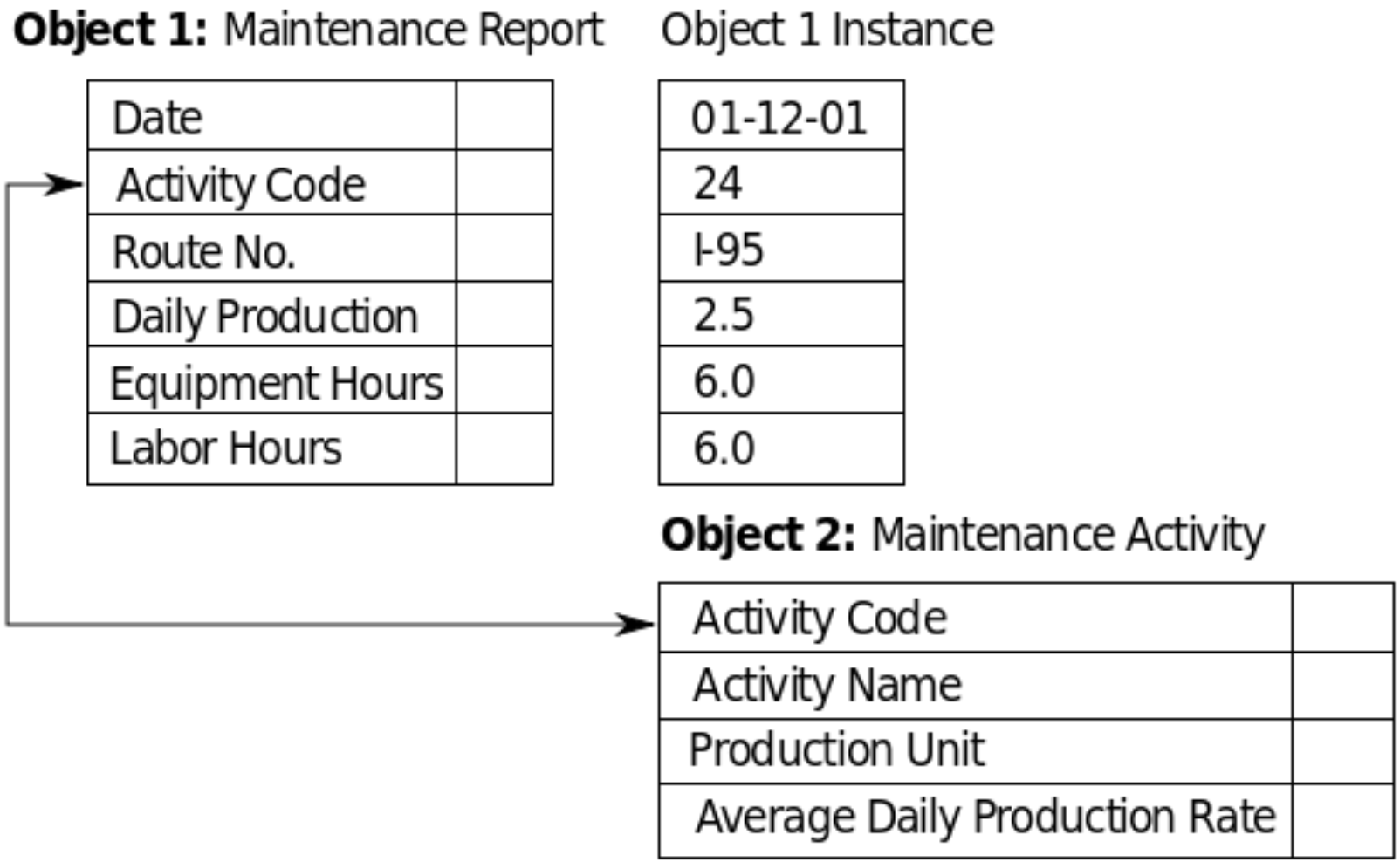


Object-Oriented DATABASE MODEL

It was introduced to overcome the previous shortcoming.

- IT is a collection of object whose behavior, state and relationship is defined according to object oriented concept.

Object-Oriented Model



Hierarchical Model

ADVANTAGES:

- Hierarchical Model is simple to construct and operate on
- Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
- Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

DISADVANTAGES:

- Navigational and procedural nature of processing
- Database is visualized as a linear arrangement of records
- Little scope for "query optimization"

Network Model

ADVANTAGES:

- Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
- Can handle most situations for modeling using record types and relationship types.
- Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.

DISADVANTAGES:

- Navigational and procedural nature of processing
- Database contains a complex array of pointers that thread through a set of records. Little scope for automated "query optimization”

Schemas versus Instances

- **Database Schema**: The *description* of a database is called the database schema, which is specified during database design and is not expected to change frequently. Most data models have certain conventions for displaying schemas as diagrams. A display schema is called a schema diagram. It includes descriptions of the database structure and the constraints that should hold on the database.
- **Schema Diagram**: A diagrammatic display of a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

Database Schema Vs. Database State

- **Database State:** the data in a database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database.
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.*
  - **Schema** is also called **intension**, whereas **state** is called **extension**.

Example of a Database Schema

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Example of a database state

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Data Models

- Entity-Relationship model
- Relational model
- Other models:
  - object-oriented model
  - semi-structured data models
  - Older models: network model and hierarchical model

Entity-Relationship Model

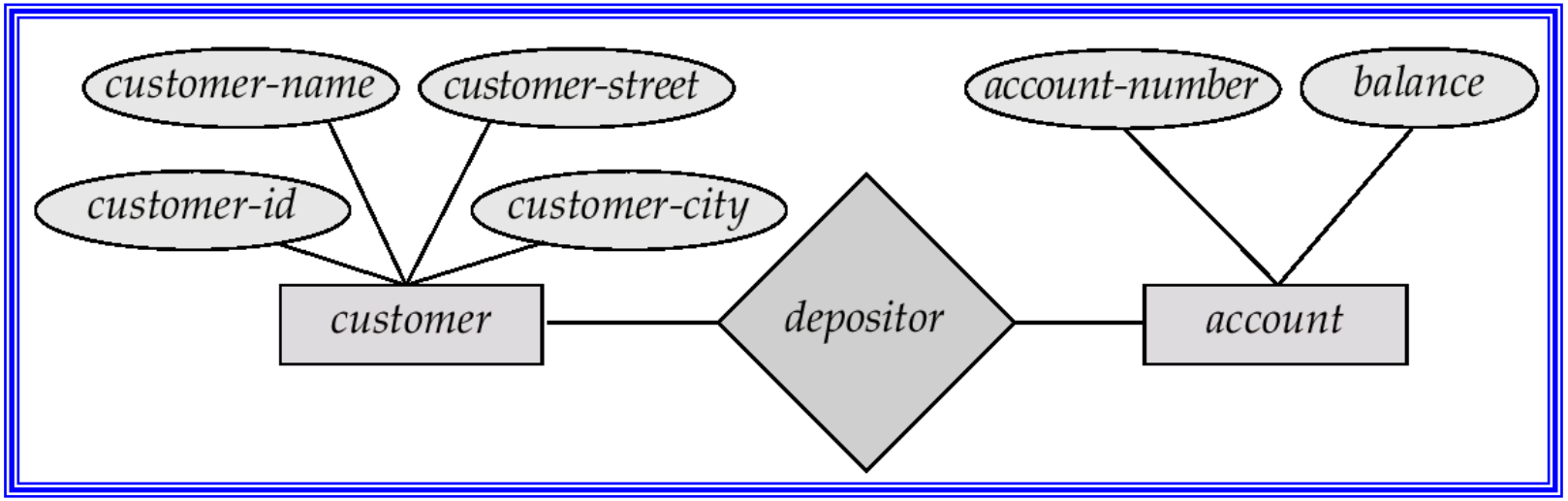


Fig: Example of schema in the Entity-relationship model

Entity Relationship Model (Cont.)

- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
  - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing

Relational Model

- Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
- Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
- Several free open source implementations, e.g. MySQL, PostgreSQL
- Currently most dominant for developing database applications.
- SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, …

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence**
  - Support of **multiple views** of the data

Mappings among schema levels are needed to transform requests and data. Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Three-Schema Architecture

- Defines DBMS schemas at *three levels*:
  - **Internal schema** at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

External View

- A user is anyone who needs to access some portion of the data.
  - Access via a 3GL,COBOL, etc (programmer) or a query language (causal user).
  - All access methods include a data sub-language (DSL).
- A DSL is a combination of two languages:
  - A data definition language (DDL) - definition and description
  - A data manipulation language (DML) - manipulating data
- 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.

Conceptual View

- An abstract representation of the entire information content of the database.
- It is in general a view of the data as it actually is.
- It consists of multiple occurrences of multiple types of conceptual record
- To achieve data independence, the definitions of conceptual records must involve information content only.
- The conceptual schema, as well as definitions, contains authorization and validation procedures.

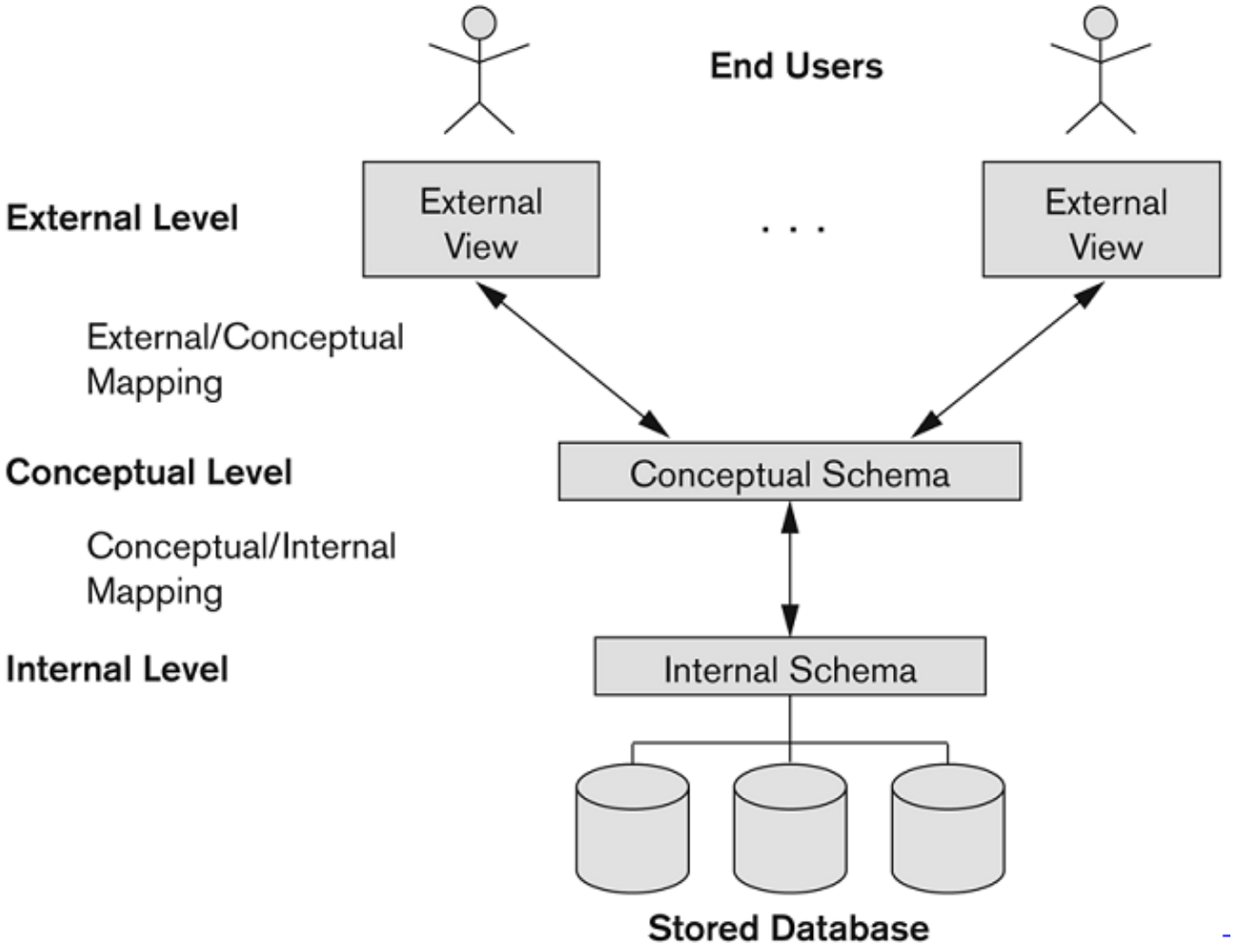
Internal View

- This is a very low-level representation of the entire database
- It is at one remove from the physical level
- The internal view is described by the internal schema:
  - defines the various types of stored record
  - 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.

Mappings

- The conceptual/internal mapping:
  - defines conceptual and internal view correspondence
  - specifies mapping from conceptual records to their stored counterparts
- An external/conceptual mapping:
  - defines a particular external and conceptual view correspondence
- 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.
- 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.

The three-schema architecture



Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.

- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.

Data Independence

When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are *unchanged*. Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

DBMS Languages

- **Data Definition Language** (**DDL**): Used by the DBA and database designers to specify the *conceptual schema* of a database.

- In many DBMSs, the DDL is also used to define internal and external schemas (views).

- In some DBMSs, separate **storage definition language** (**SDL**) and **view definition language** (**VDL**) are used to define internal and external schemas.

DBMS Languages Contd..

- **Data Manipulation Language** (**DML**): Used to specify database retrievals and updates.
  - DML commands (**data sublanguage**) can be *embedded* in a general-purpose programming language.
  - Alternatively, *stand-alone* DML commands can be applied directly (**query language**).

DBMS Languages Contd..

- **High Level** or **Non-procedural Languages:** e.g., SQL, specify what data to retrieve than how to retrieve. Also called *declarative* languages.

- **Low Level** or **Procedural Languages:** record-at-a-time; they specify *how* to retrieve data and include constructs such as looping.

DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Natural language: requests in written English
  - Combinations of the above

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC for other programming languages
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

Database System Utilities

- To perform certain functions such as:
  - *Loading* data stored in files into a database. Includes data conversion tools.
  - *Backing up* the database periodically on tape.
  - *Reorganizing* database file structures.
  - *Report generation* utilities.
  - *Performance monitoring* utilities.
  - Other functions, such as *sorting*, *user monitoring*, *data compression*, etc.

Database Users

Users are differentiated by the way they expect to interact with the system

- **Application programmers** – 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
- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - –Examples, people accessing database over the web, bank tellers, clerical staff

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - –Schema definition
  - –Storage structure and access method definition
  - –Schema and physical organization modification
  - –Granting user authority to access the database
  - –Specifying integrity constraints
  - –Monitoring performance and responding to changes in requirements

Other Tools

- **Data dictionary / repository**:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
  - *Active* data dictionary is accessed by DBMS software and users/DBA.
  - *Passive* data dictionary is accessed by users/DBA only.

Centralized and Client-Server Architectures

- **Centralized DBMS:** combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

- **Basic Client-Server Architectures**
  - **Specialized Servers with Specialized functions**
  - **Clients**
  - **DBMS Server**

- **DBMS SERVER**
- Provides database query and transaction services to the clients
- Sometimes called query and transaction servers

- **CLIENT**
- Provide appropriate interfaces and a client-version of the system to access and utilize the server resources.
- Clients maybe diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network. (LAN: local area network, wireless network, etc.)

Two Tier Client-Server Architecture

- **User Interface Programs and Application Programs** run on the client side
- Interface called **ODBC (Open Database Connectivity)** provides an Application program interface (API) allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.
- A client program may connect to several DBMSs.
- Other variations of clients are possible: e.g., in some DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc. In such situations the server may be called the **Data Server**.

Three Tier Client-Server Architecture

- Common for **Web applications**
- Intermediate Layer called **Application Server** or **Web Server:**
  - stores the web connectivity software and **the rules and business logic (constraints)** part of the application used to access the right amount of data from the database server
  - acts like a conduit for sending partially processed data between the database server and the client.
- **Additional Features- Security:**
  - encrypt the data at the server before transmission
  - decrypt data at the client

Classification of DBMSs

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.

- **Other classifications:**
  - Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)
  - Distributed Database Systems have now come to be known as client server based database systems_because they do not support a totally distributed environment, but rather a set of database servers supporting a set of clients**.**

Typical DBMS Component Modules