Chapter 1: Introduction

Outline

- The Need for Databases
- Users of Database system
- Data Models
- Schemas and Instances
- Three Schema Architecture
- Data Independence
- DBA & its role

Basic Definitions

- **Database**: A collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Mini-world**: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **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.

Definitions of Database

- ***Def 1:*** Database is an *organized collection* of *logically related data*
- ***Def 2:*** A database is a *shared* collection of *logically related data* that is stored to meet the requirements of *different users* of an organization
- ***Def 3:*** A database is a *self-describing* collection of integrated records
- ***Def 4:*** A database models a *particular real world system* in the computer in the form of data

Definitions

- ***Data***: stored representations of meaningful objects and events or
- Referred to facts concerning objects and events that could be recorded and stored on computer media
  - Structured: numbers, text, dates
  - Unstructured: images, video, documents
- ***Information***: data processed to increase knowledge in the person using the data
- ***Metadata:*** data that describes the properties and context of user data

What is a Database

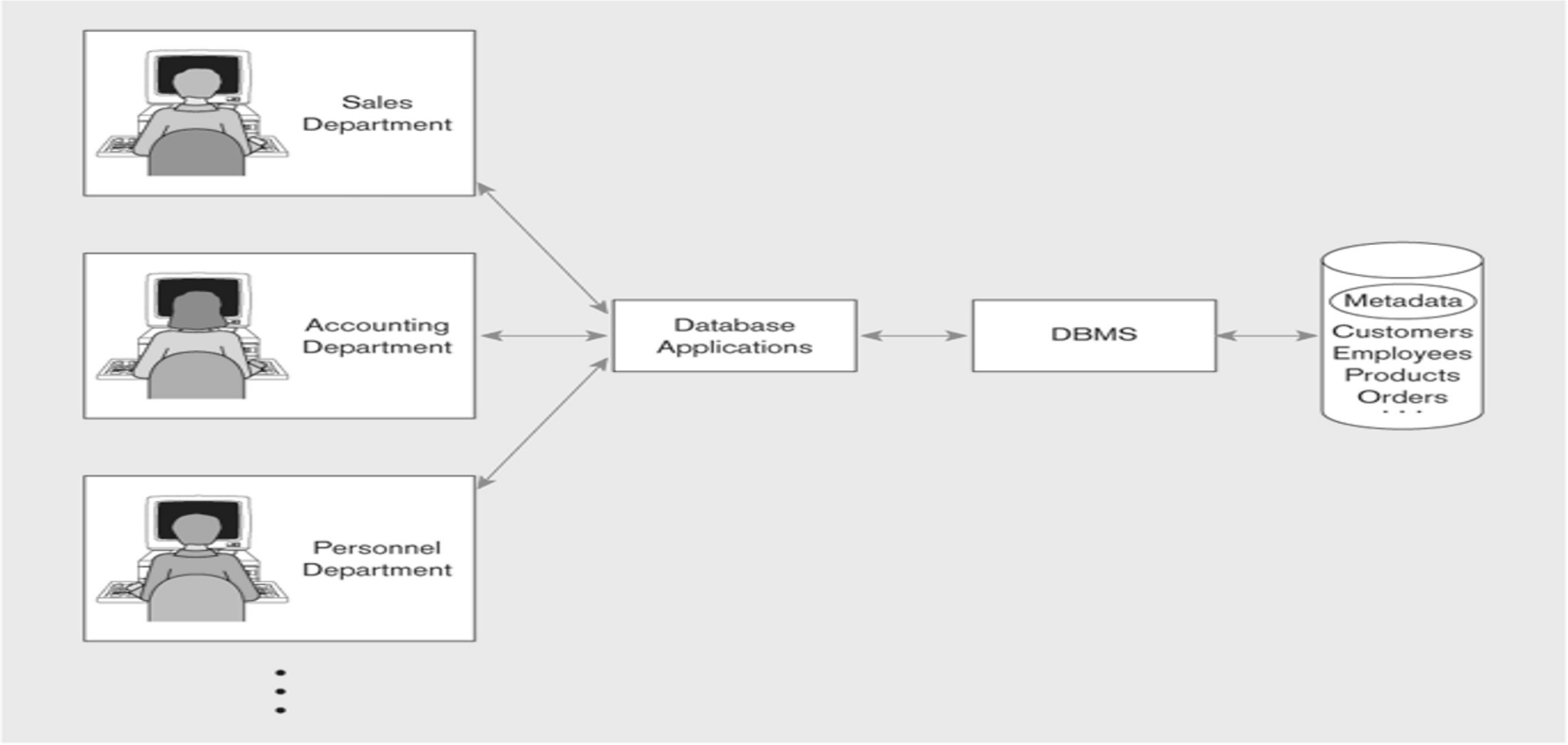
- Shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization.
- System catalog (metadata) provides description of data to enable program–data independence.
- Logically related data comprises entities, attributes, and relationships of an organization's information.

Database Management System

- A software system that is used to create, maintain, and provide controlled access to users of a database
- (Database) application program: A computer program that interacts with database by issuing an appropriate request (SQL statement) to the DBMS

Database Management System

Figure 1-3 Database approach at Pine Valley Furniture Company



DBMS manages data resources like an operating system manages hardware resources

Typical DBMS Functionality

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent

Typical DBMS Functionality

Other features:

- Protection or Security measures to prevent unauthorized access
- “Active” processing to take internal actions on data
- Presentation and Visualization of data

Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example**: Part of a UNIVERSITY environment.
- **Some mini-world *entities***:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

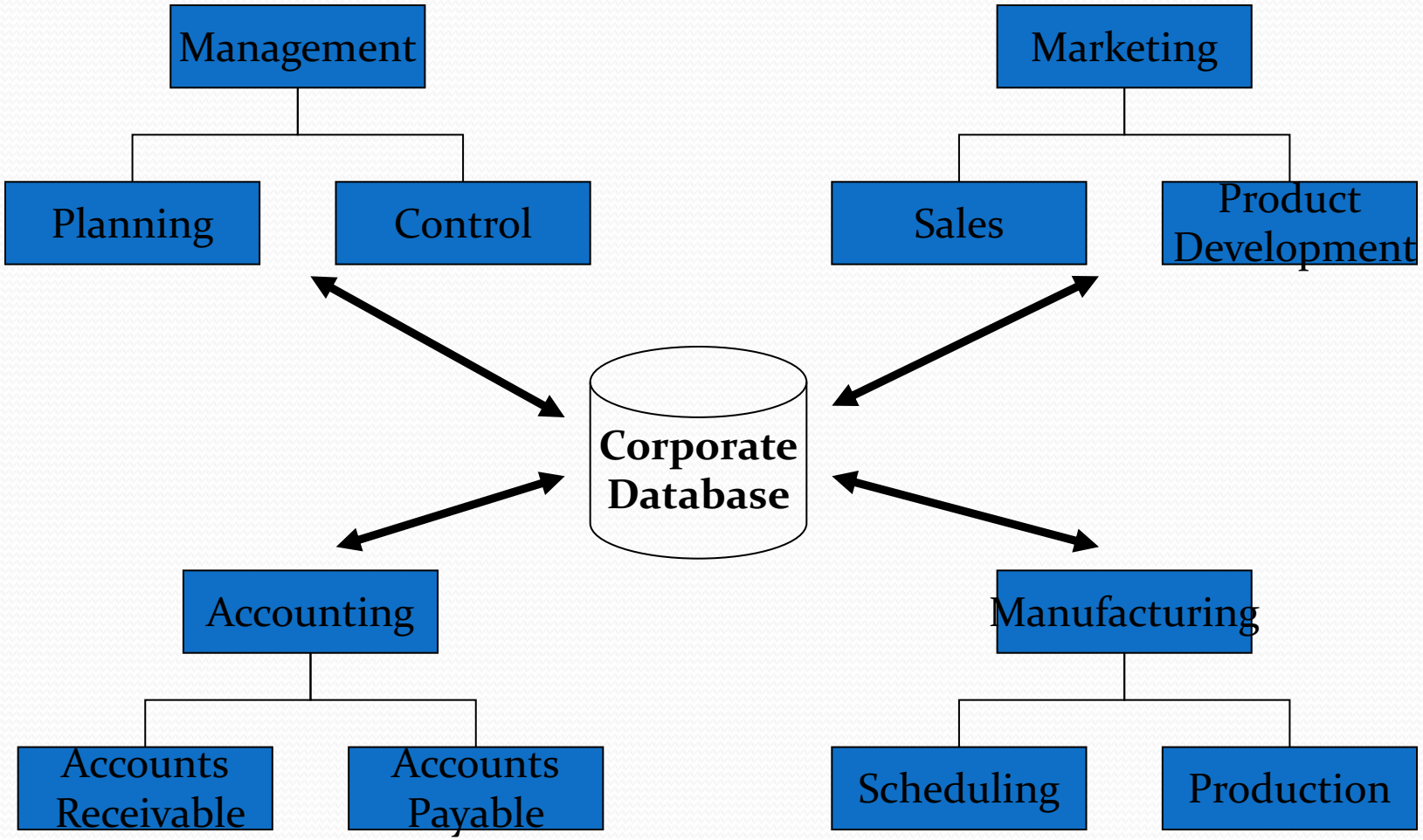
Note: The above could be expressed in the ENTITY-RELATIONSHIP data model.

Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships***:
  - SECTIONs *are of* specific COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have* prerequisite COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

Note: The above could be expressed in the *ENTITY-RELATIONSHIP* data model.

The concept of a shared organizational database



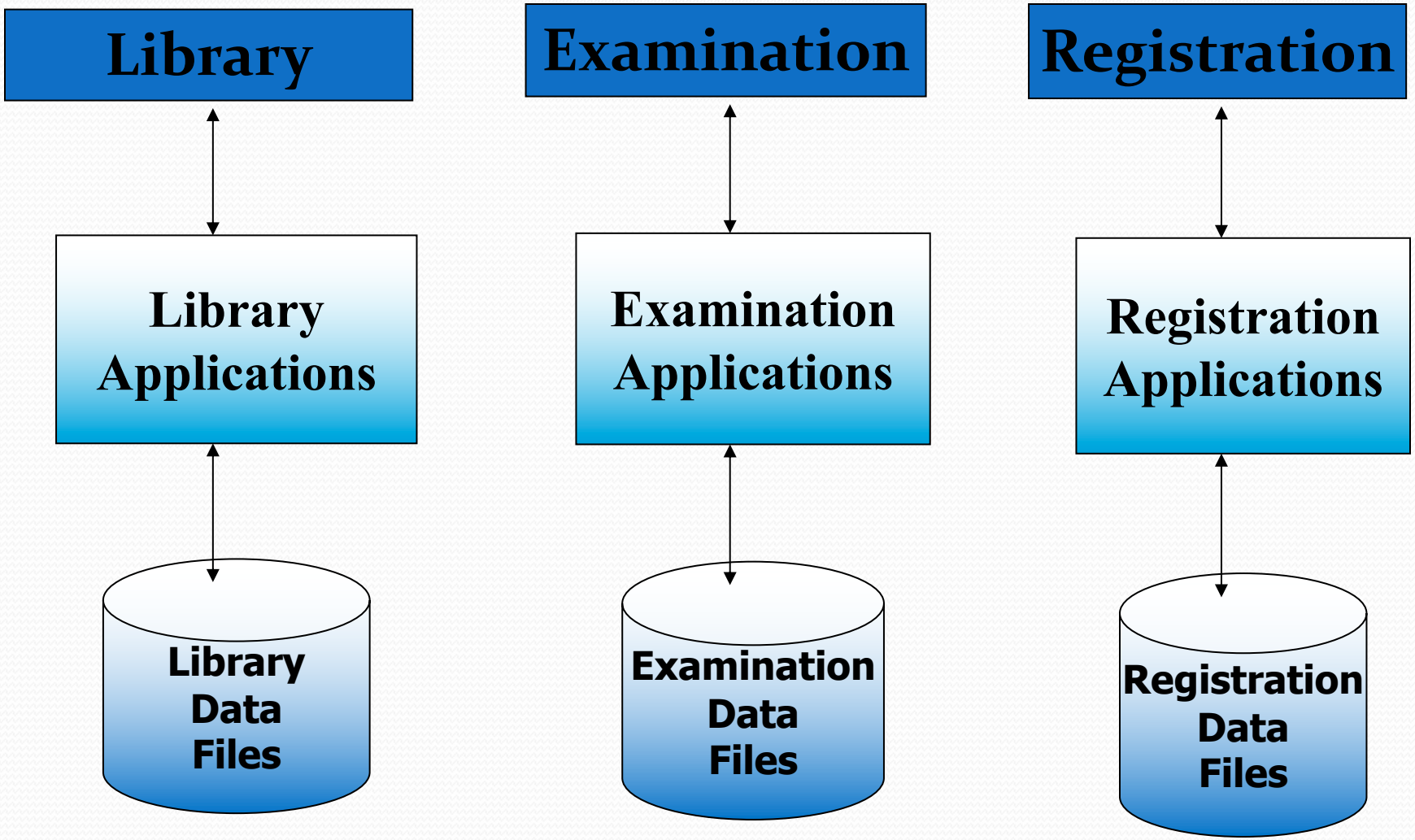
A bit of History

- Computer initially used for computational/ engineering purposes
- Commercial applications introduced File Processing System

File Processing System

- A collection of application programs that perform services for the end-users such as production of reports
- Each program defines and manages its own data

File Processing Systems



Program and Data Interdependence

File Processing Systems

Library
Reg_Number
Name
Father Name
Books Issued
Fine

Examination
Reg_Number
Name
Address
Class
Semester
Grade

Registration
Reg_Number
Name
Father Name
Phone
Address
Class

Files Based Processing

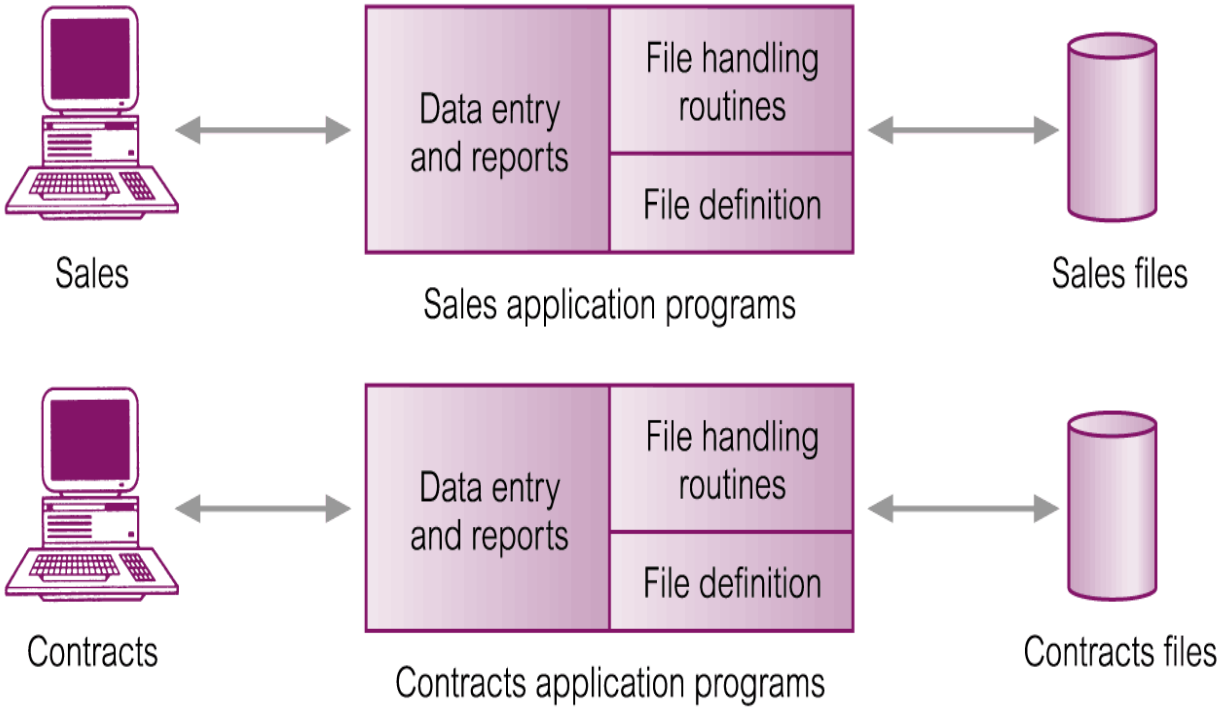


Figure 1.5
File-based processing.

Disadvantages of File Processing

- **Program-Data Dependence**
  - File structure is defined in the program code.
  - All programs maintain metadata for each file they use
- **Duplication of Data (Data Redundancy)**
  - Different systems/programs have separate copies of the same data
  - Same data is held by different programs.
  - Wasted space and potentially different values and/or different formats for the same item.
- **Limited Data Sharing**
  - No centralized control of data
  - Programs are written in different languages, and so cannot easily access each other's files.

Disadvantages of File Processing

- **Lengthy Development Times**
  - Programmers must design their own file formats
- **Excessive Program Maintenance**
  - 80% of of information systems budget
- **Vulnerable to Inconsistency**
  - Change in one table need changes in corresponding tables as well otherwise data will be inconsistent

Problems with Data Dependency

- Each application programmer must maintain their own data
- Each application program needs to include code for the metadata of each file
- Each application program must have its own processing routines for reading, inserting, updating and deleting data
- Lack of coordination and central control
- Non-standard file formats

Problems with Data Redundancy

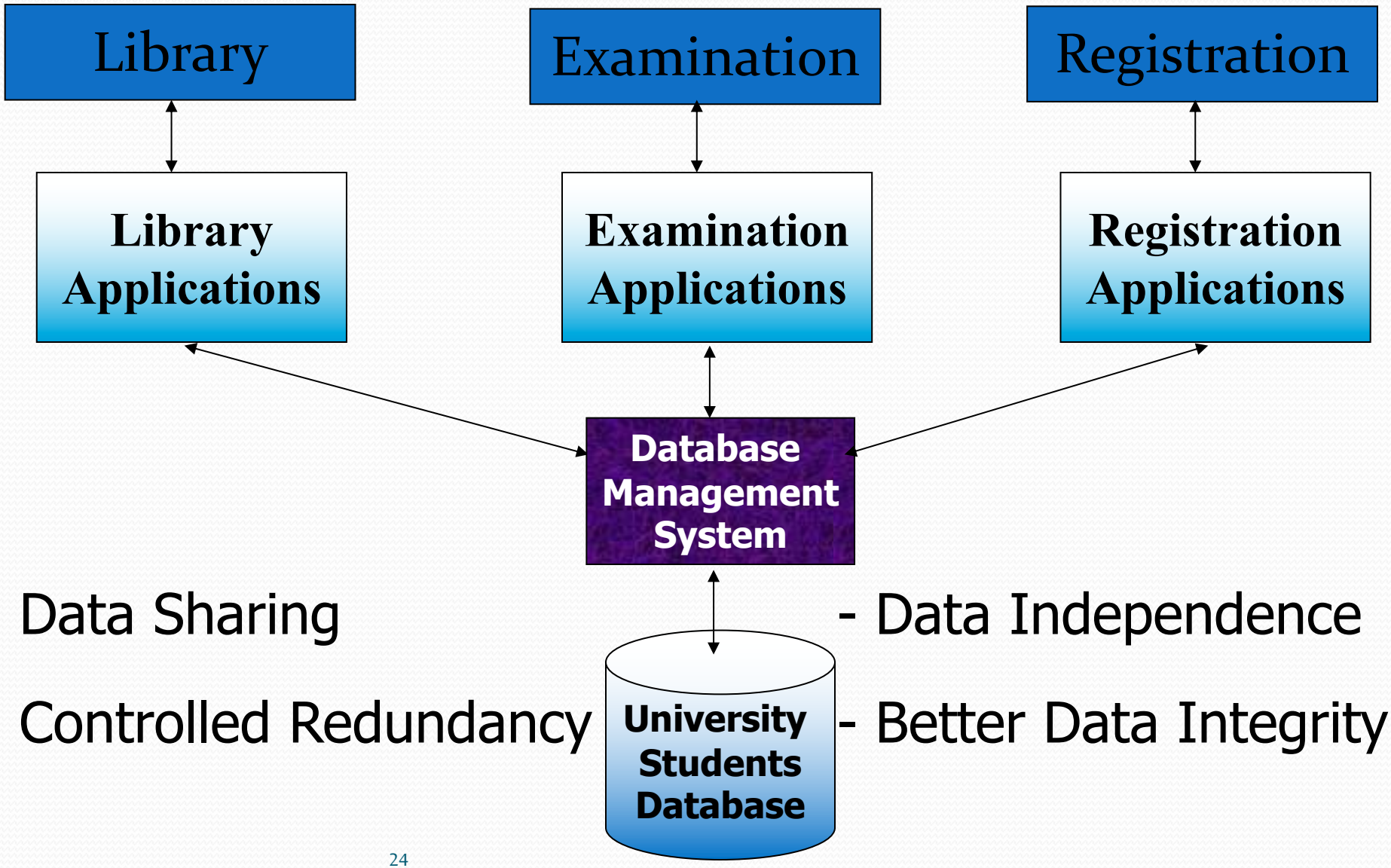
- Waste of space to have duplicate data
- Causes more maintenance headaches
- The biggest problem:
  - **When data changes in one file, could cause inconsistencies (<u>Vulnerable to Inconsistency</u>)**
  - Compromises ***data integrity (data reliability)***

SOLUTION:
The DATABASE Approach

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

This requires a
Database and Database Management System (DBMS)

Advantages of Database Approach



- Data Sharing
- Data Independence
- Controlled Redundancy
- Better Data Integrity

Levels of Abstraction

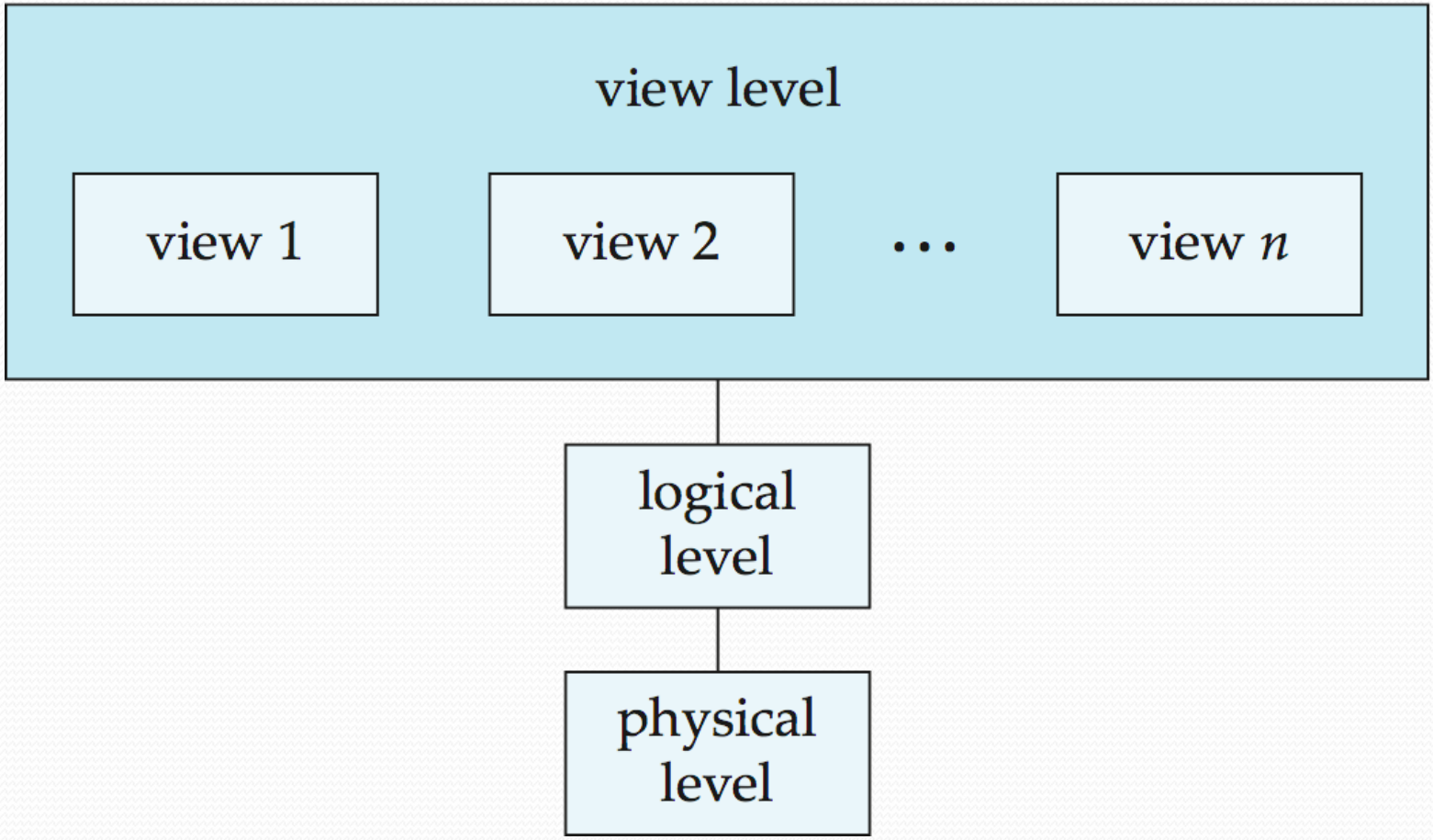
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



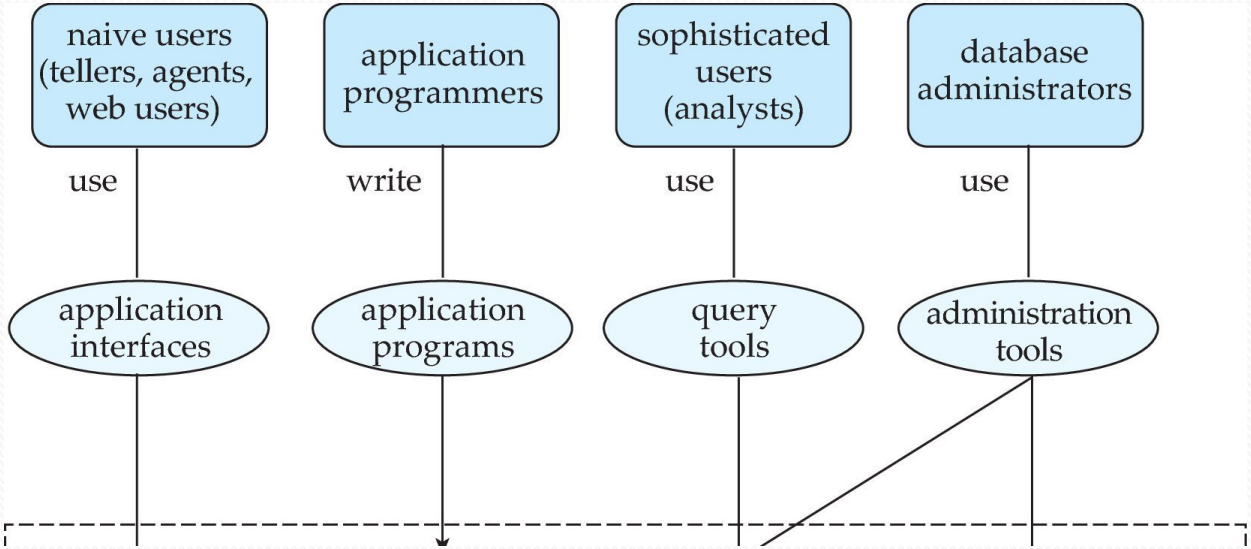
Instances and Schemas

- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Database Users

Users may be divided into those who actually use and control the content (called “Actors on the Scene”) and those who enable the database to be developed and the DBMS software to be designed and implemented (called “Workers Behind the Scene”).

Database Users and Administrators



Database Users

Actors on the scene

- **Database administrators:** responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database Designers:** responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.
- **End-users:** they use the data for queries, reports and some of them actually update the database content

Database Users

- **Application Programmers -** They are the developers who interact with the database by means of DML queries. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.

- **Sophisticated Users -** They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.

Database Users

- **Specialized Users** - These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.
- **Naive Users** - these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.

Database Administrator

- A Database Administrator, Database Analyst or Database Developer is the person responsible for managing the information within an organization.
- As most companies continue to experience inevitable growth of their databases, these positions are probably the most solid within the IT industry.

Role of Database Administrator

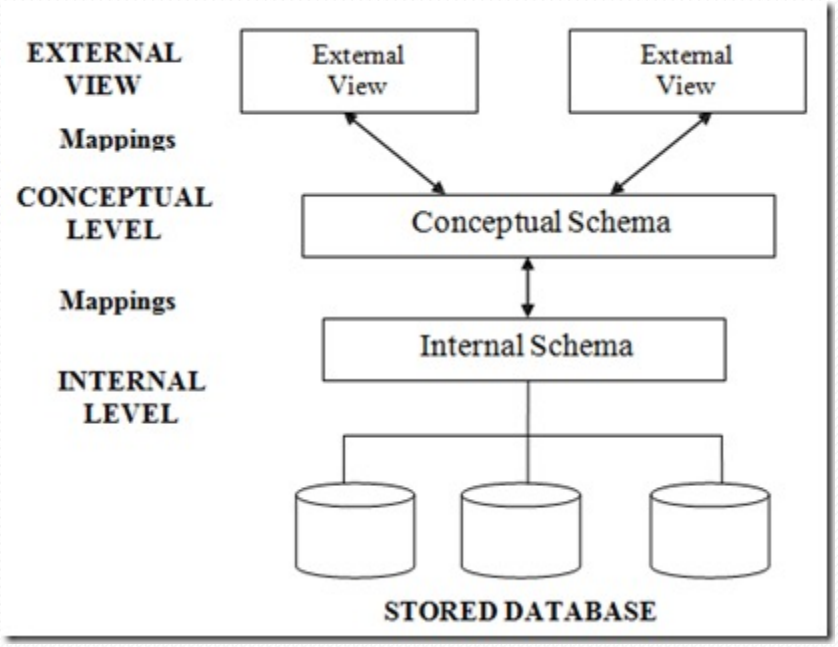
- *1. Schema Definition:*
  - The DBA definition the logical Schema of the database. A Schema refers to the overall logical structure of the database.
  - According to this schema, database will be developed to store required data for an organization.
- *2. Storage Structure and Access Method Definition:*
  - The DBA decides how the data is to be represented in the stored database.
- *3. Assisting Application Programmers:*
  - The DBA provides assistance to application programmers to develop application programs.

Role of Database Administrator

- *4. Physical Organization Modification:*
  - The DBA modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.
- *5. Approving Data Access:*
  - The DBA determines which user needs access to which part of the database.
  - According to this, various types of authorizations are granted to different users.
- *6. Monitoring Performance:*
  - The DBA monitors performance of the system.The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.
- *7. Backup and Recovery:*
  - Database should not be lost or damaged.
  - The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
  - In case of failure, such as virus attack database is recovered from this backup.

Three-Schema Architecture

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



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.

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.

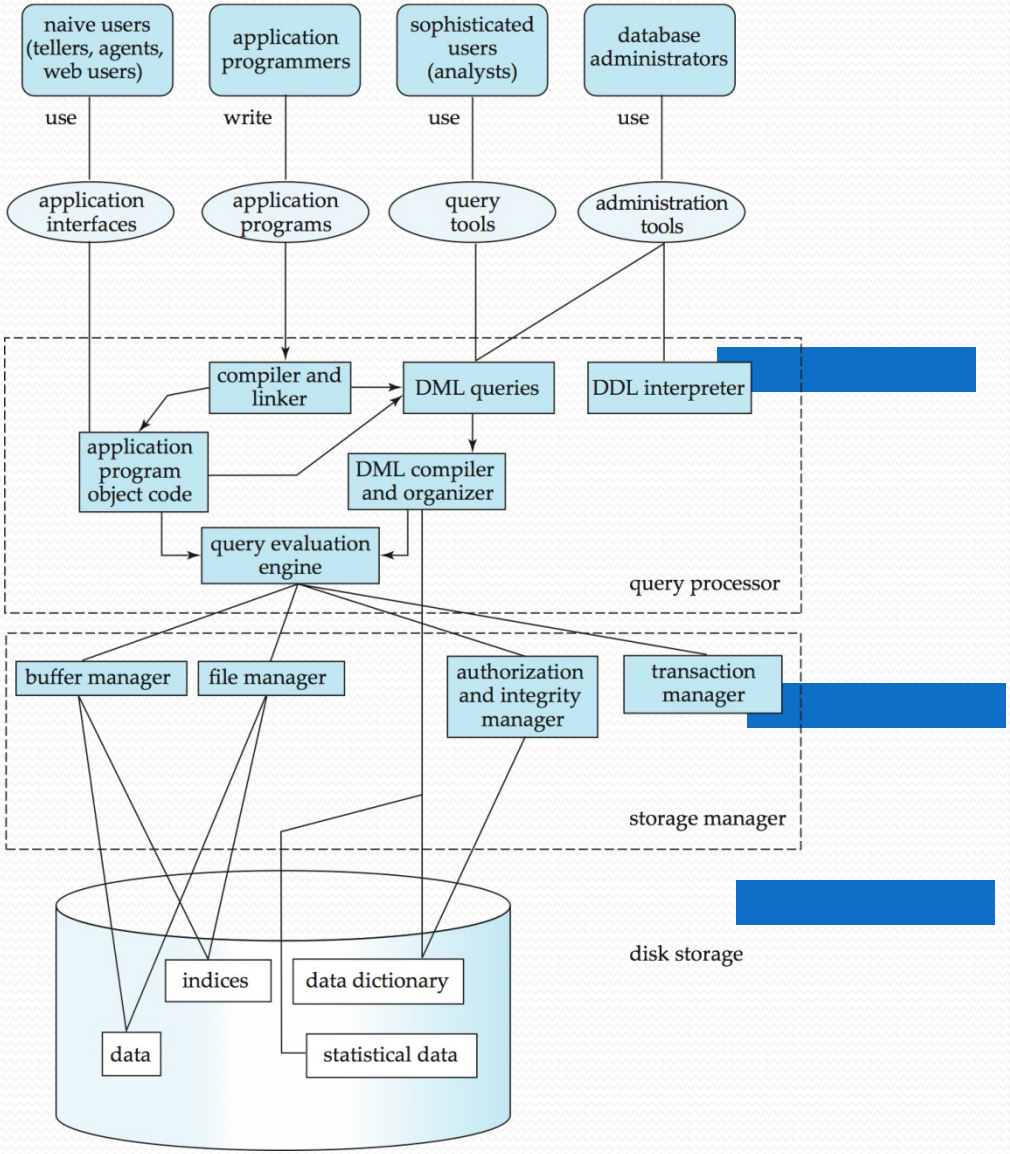
Data Independence

- **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.

Database System Internals



Database Engine

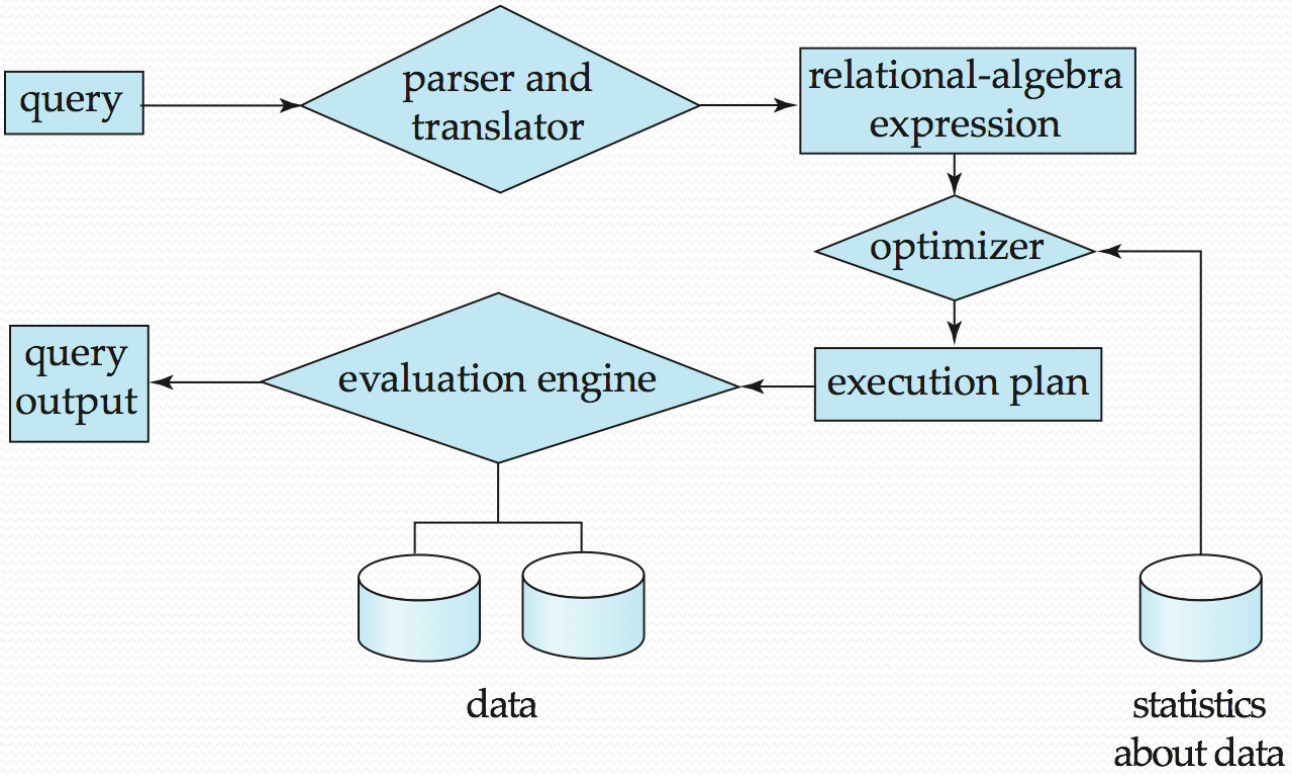
- Storage manager
- Query processing
- Transaction manager

Storage Management

- **Storage manager** is 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.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Architecture

The architecture of a database systems is greatly influenced by

the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

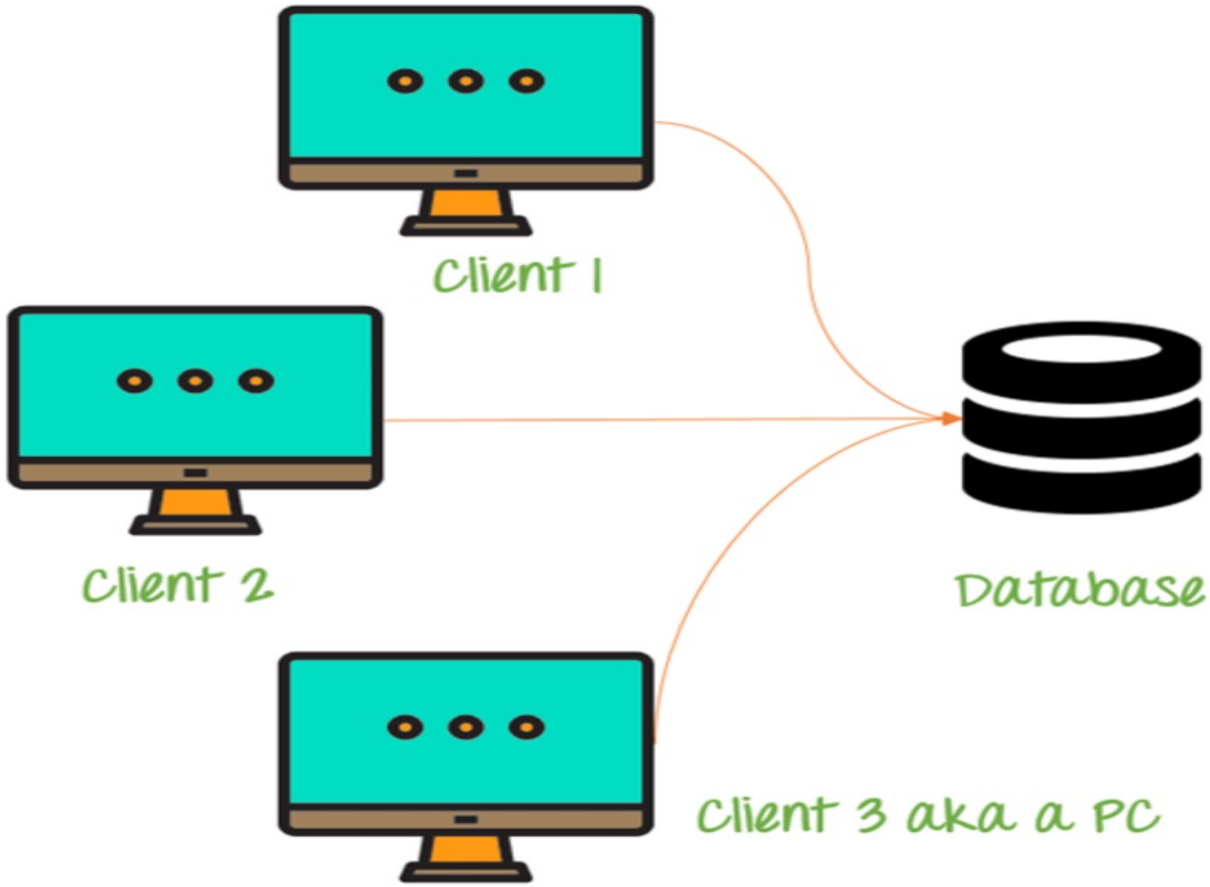
1-Tier Architecture

1 Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

1 Tier Architecture Diagram

2-Tier Architecture

A 2 Tier Architecture in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



2 Tier Architecture Diagram

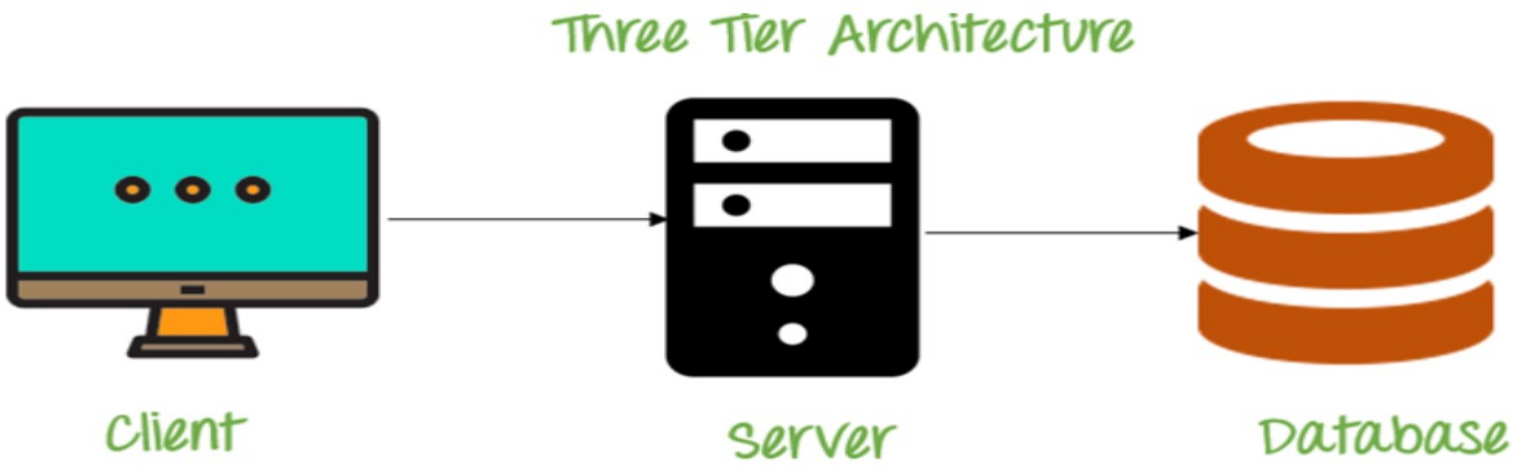
In the above 2 Tier client-server architecture of database management system, we can see that one server is connected with clients 1, 2, and 3.

3-Tier Architecture

A 3 Tier Architecture in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

3-Tier database Architecture design is an extension of the 2-tier client-server architecture. A 3-tier architecture has the following layers:

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server



3 Tier Architecture Diagram

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.

Summary

- An Architecture of DBMS helps in design, development, implementation, and maintenance of a database
- The simplest database system architecture is 1 tier where the Client, Server, and Database all reside on the same machine
- A two-tier architecture is a database architecture in DBMS where presentation layer runs on a client and data is stored on a server
- Three-tier client-server architecture consists of the Presentation layer (PC, Tablet, Mobile, etc.), Application layer (server) and Database Server

Main Characteristics of the Database Approach

- Self-describing nature of a database system: A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Support of multiple views of the data: Each user may see a different view of the database, which describes *only* the data of interest to that user.

Main Characteristics of the Database Approach

- Sharing of data and multiuser transaction processing : allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted. OLTP (Online Transaction Processing) is a major part of database applications.

Table 1-5
Advantages of the Database Approach

Program-data independence
Minimal data redundancy
Improved data consistency
Improved data sharing
Increased productivity of application development
Enforcement of standards
Improved data quality
Improved data accessibility and responsiveness
Reduced program maintenance
Improved decision support

Table 1-6

Costs and Risks of the Database Approach

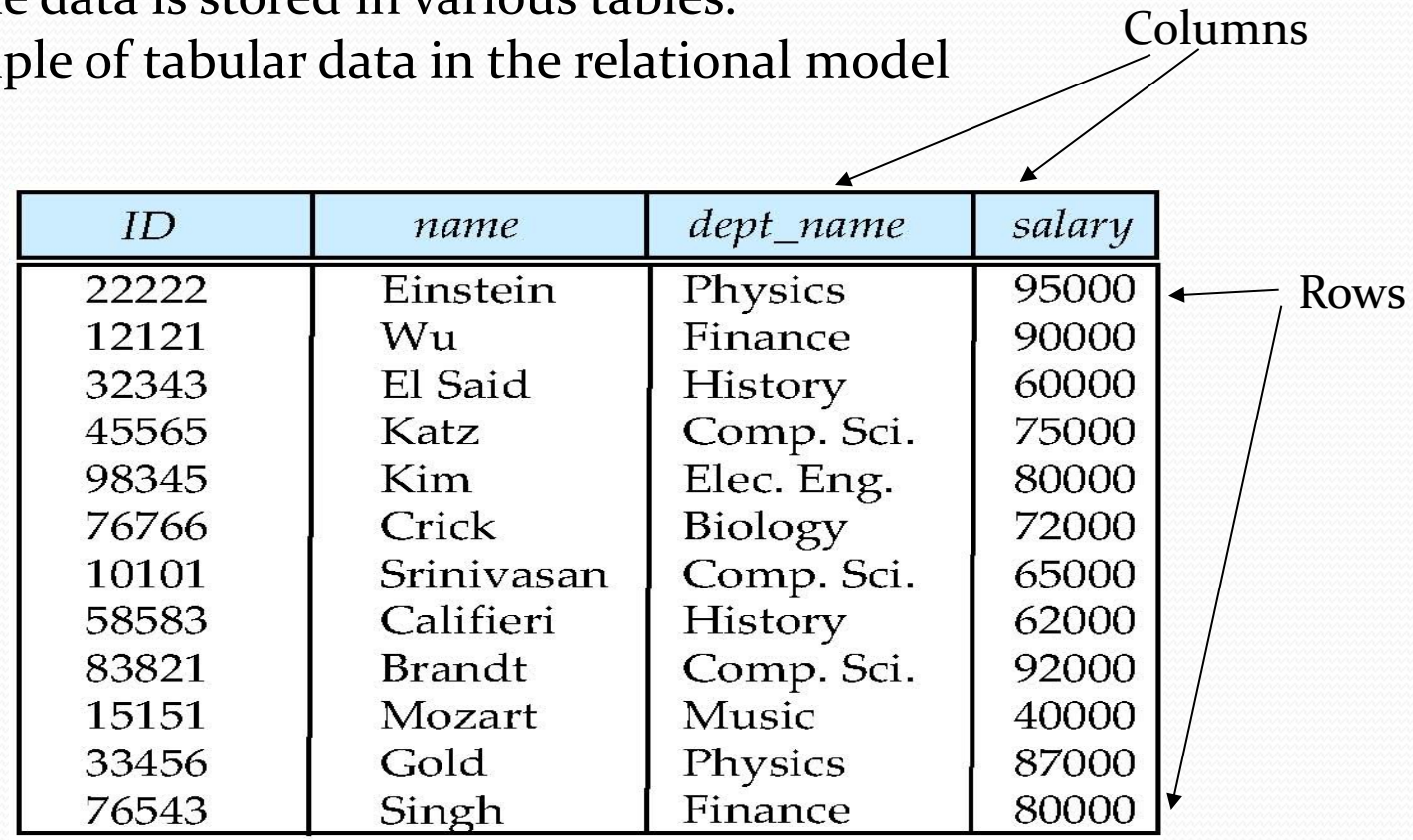
New, specialized personnel
Installation and management cost and complexity
Conversion costs
Need for explicit backup and recovery
Organizational conflict

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- 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

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is "good"
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

End of Chapter 1