

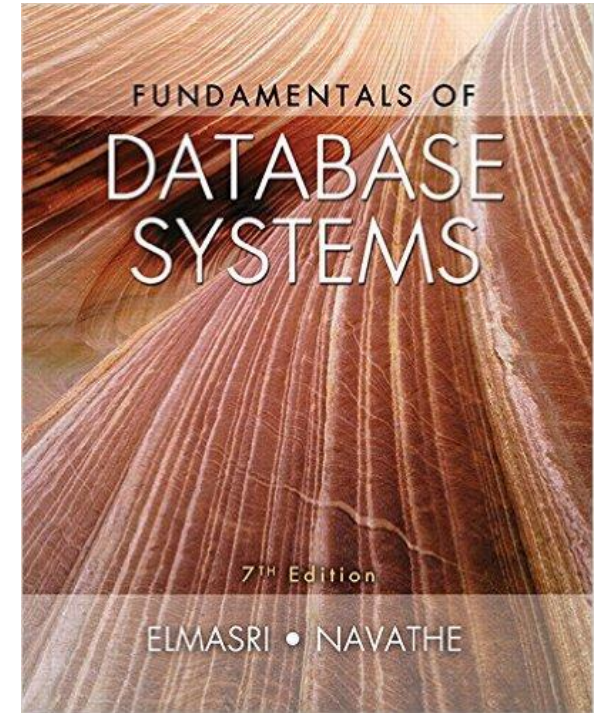
# INTRODUCTION

---

Week 1

# Introduction

- Data, Information and Data Cycle
- Purpose of Database Systems
- View of Data
- Data Models
- Data Definition Language
- Data Manipulation Language
- Database Types
- Database Administrator
- Database Users



## Recommended Book:

- Fundamentals of database systems (7<sup>th</sup> Edition) by Ramez Elmasri and Shamkant B. Navathe
- Database Management Systems (3<sup>rd</sup> Edition) by Raghu Ramakrishnan and Johannes Gehrke

# What is Data?

- Data can be defined as a representation of facts, concepts or instructions in a formalized manner which should be suitable for communication, interpretation, or processing by human or electronic machine.
- Data
  - is a set of discrete, objective facts about events
  - provides no judgment or interpretation
  - gives no sustainable basis for action
  - cannot tell you what to do
  - says nothing about its own importance or irrelevance



# What is Information?

- Information is: data that is organised, classified or presented to make them meaningful or useful value for the receiver.
- Information is the processed data on which decisions and actions are based.
- Information provides context for data.
- For the decision to be meaningful, the processed data must qualify for the following characteristics:
  - **Timely** - Information should be available when required.
  - **Accuracy** - Information should be accurate.
  - **Completeness** - Information should be complete.

# Information is...

- Contextualised: we know for what purpose the data was gathered
- Categorized: we know the units of analysis of key components of the data
- Calculated: the data may have been analysed mathematically or statistically
- Corrected: errors have been removed from the data
- Condensed: the data may have been summarized in a more concise form



# Data and Information Examples

- The history of temperature readings all over the world for the past 100 years is data. If this data is organized and analysed to find that global temperature is rising, then that is information.
- The number of visitors to a website by country is an example of data. Finding out that traffic from the U.S. is increasing while that from Australia is decreasing is meaningful information.

# Data Processing Cycle

- Data processing is the re-structuring or re-ordering of data by people or machine to increase their usefulness and add values for particular purpose.
- Data processing consists of basic steps input, processing and output.



# Data Processing Cycle

- **Input** - In this step the input data is prepared in some convenient form for processing. The form will depend on the processing machine.
  - For example, when electronic computers are used, the input data could be recorded on any one of several types of input medium, such as magnetic disks, tapes and so on.
- **Processing** - In this step input data is changed to produce data in a more useful form.
  - For example, pay-checks may be calculated from the time cards, or a summary of sales for the month may be calculated from the sales orders.
- **Output** - Here the result of the proceeding processing step are collected. The particular form of the output data depends on the use of the data.
  - For example, output data may be pay-checks for employees.



**“ONE PERSON'S  
INFORMATION IS  
ANOTHER  
PERSON'S DATA”**

# What is a Database?

- General Definition:  
A database is a **collection** of related data.
- Database houses a collection of
  - End user data – raw facts of interests to the end user
  - Metadata – data about data
    - Provides a description of the data characteristics and the set of relationships that link the data found within the database
- A database can be of any size and of varying complexity
- A database may be generated and maintained manually or it may be computerized (set of application programs or by DBMS).



# Database Properties

1. A database represents some aspects of the real world, sometimes called the **miniworld** or the **universe of discourse (UoD)**. Changes to the miniworld are reflected in the database.
2. A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot be referred to as a database.
3. A database is designed, built, and populated with data for a specific purpose.
  - It has an intended group of users and some preconceived applications in which these users are interested.

## Restrictive definition:

A database is a persistent, logically coherent collection of inherently meaningful data, relevant to some aspects of the real world.

# Database Management System (DBMS)

- DBMS is a collection of programs that enables users to create and maintain a database.
- It also helps to control access to the database by various users.
- DBMS provides an environment that is both *convenient* and *efficient* to use.
- We will call the database and DBMS software together a database system

# Database Management System (DBMS)

- The DBMS is a general-purpose software system that facilitates the processes of **defining, constructing, and manipulating** databases for various applications
- **Defining:** a database involves specifying the data types, structures, and constraints for the data to be stored in the database.
- **Constructing:** the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.
- **Manipulating:** a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

# Example

UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment

- Define: file (records), data elements, data type (for each data element)
- Construct: store data in the appropriate files (note that records may be related between files)
- Manipulation: querying, updating
  - Informal queries and updates must be specified precisely in the database system language before they can be processed.

# DBMS Applications

- It contains information about a particular enterprise
- Databases touch all aspects of our lives
- Database Applications:
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions

# Purpose of Database System

- In the early days, database applications were built on top of file systems
- 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
    - Integrity constraints (e.g. account balance  $> 0$ ) become part of program code
    - Hard to add new constraints or change existing ones



# Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. 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
      - E.g. two people reading a balance and updating it at the same time
  - Security problems
- Database systems offer solutions to all the above problems

# Benefits of database approach

- Potential for Enforcing Standards
  - Defined for names and formats of data elements, display formats, report structures, terminology, etc..
- Reduced Application Development Time
  - A prime selling feature of the database approach is that developing a new application such as the retrieval of certain data from the database for printing a new report takes very little time
  - Development time using a DBMS is estimated to be one-sixth to one-fourth of that for a traditional file system.
- Flexibility
  - It may be necessary to change the structure of a database as requirements change

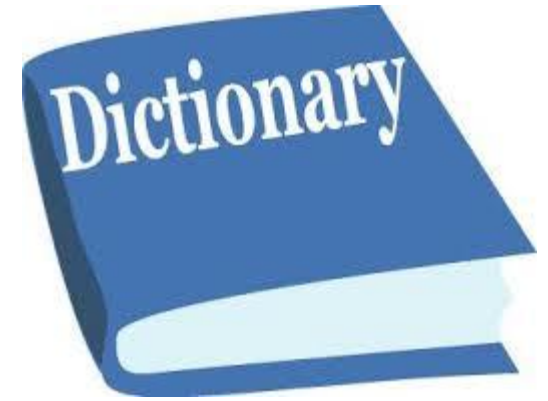
## Benefits of database approach (Cont.)

- Availability of Up-to-Date Information
- Economies of Scale
  - The DBMS approach permits consolidation of data and applications, thus reducing the amount of wasteful overlap between activities of data-processing personnel in different projects or departments.

# When not to use a DBMS

- Overhead costs of using DBMS :
  - High initial investment in hardware, software, and training.
  - Generality that a DBMS provides for defining and processing data.
  - Overhead for providing security, concurrency control, recovery, and integrity functions.
- Additional costs
  - If the database designers and DBA do not properly design the database
  - If database systems applications are not implemented properly
- Use regular files under the following circumstances:
  - The database and applications are simple, well defined, and not expected to change.
  - There are stringent real-time requirements for some programs that may not be met because of DBMS overhead.
  - Multiple-user access to data is not required

# Data Dictionary



- It contains data about data
  - That is, it holds information about the database
- The data that it stores is called **Meta Data**
  - Central repository of meta data
- All well designed database will include a data dictionary
- DBMS checks the data dictionary every time the database is accessed.
- Format
  - No standard for creating a data dictionary
  - Meta data differs from table to table
- It should be stored at convenient location accessible to all users

# Active/Passive Data Dictionary

- Active - integrated data dictionary
  - It is managed automatically by the DBMS
  - It is consistent with the current structure and definition of the database.
  - It is in most of the relational database management systems
- Passive - non-integrated data dictionary
  - It is used only for documentation purposes.
  - It is a self-contained application.
  - It is managed by the users of the system and is modified whenever the structure of the database is changed.
  - Modification to data dictionary is performed manually by the user, thus, it is possible that the data dictionary may not be current with the current structure of the database.
  - This may be maintained as a separate database.

# Meta Data

- Meta Data (meta content) is defined as data providing information about one or more aspects about the data
- Data about data
- E.g. Card catalogs of libraries
  - Allows books (resources) to be found by relevant criteria
- In DBMS, it contains information about each data element
  - Name (of the data element)
  - Type
  - Range of values
  - Source
  - Access authorization
  - Indicates which application programs use the data
    - If there is a change in the data structure, all the affected programs will be known

# Advantages of Data Dictionary

- A database can be clearly defined with name of data elements, their descriptions and data structures.
- When a new user is introduced to the system, identifying table structure and types becomes simple



# Levels of Abstraction

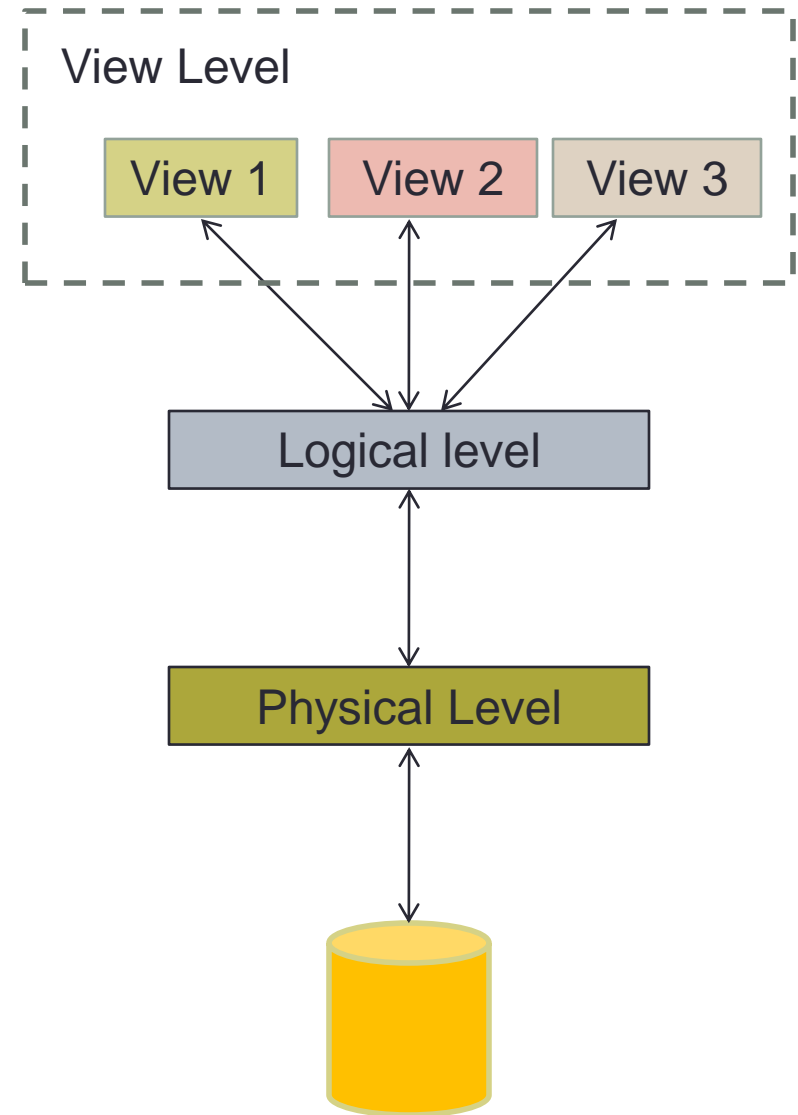
- Physical level describes how a record (e.g., customer) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record  
    name : string;  
    street : string;  
    city : integer;  
end;
```

- View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.

# View of Data

- An architecture for a database system
- Also known as ANSI-SPARC Model

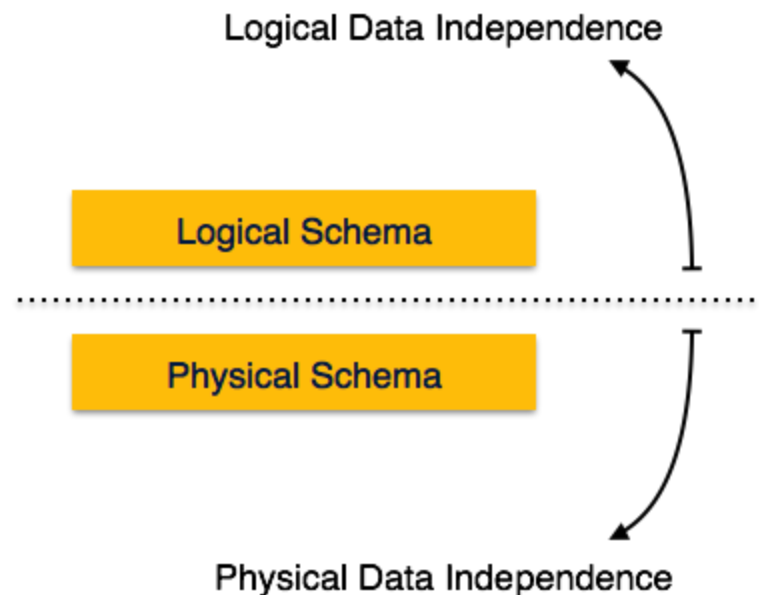


# Instances and Schemas

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

# Data Independence

- Logical data independence: Protection from changes in logical structure of data
- Physical data independence: Protection from changes in physical structure of data



# Logical Data Independence

- Logical data independence: Protection from changes in logical structure of data
  - Logical data is data about database.
  - That is, it stores information about how data is managed inside.
  - For example, a table (relation) stored in the database and all its constraints, applied on that relation.
  - If we do some changes on table format, it should not change the data residing on the disk.

# Physical Data Independence

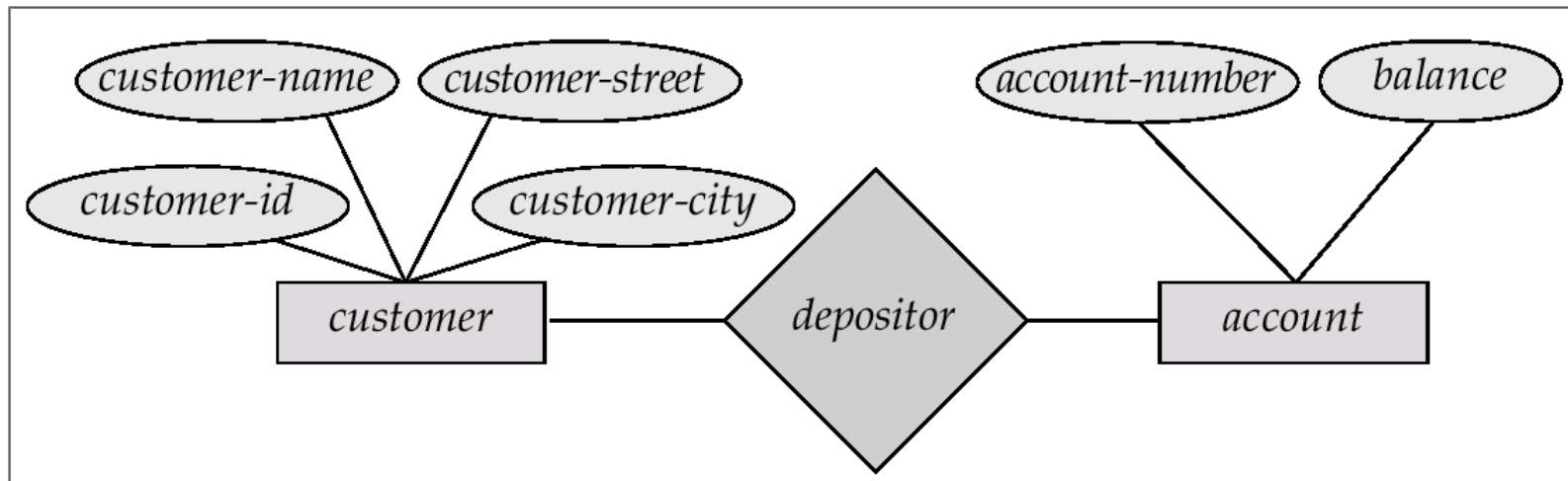
- Physical data independence: Protection from changes in physical structure of data
  - Actual data is stored in bit format on the disk.
  - Physical data independence is the power to change the physical data without impacting the schema or logical data.
  - E.g., in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

# Data Models

- A collection of tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model
- Relational model
- Other models:
  - object-oriented model
  - semi-structured data models
  - Older models: network model and hierarchical model

# Entity-Relationship Model

Example of schema in the entity-relationship model



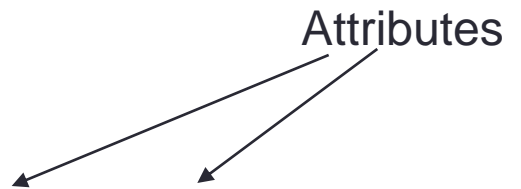


# Entity Relationship Model (Cont.)

- 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

- Example of tabular data in the relational model



<i>Customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>	<i>account-number</i>
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

# A Sample Relational Database

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

<i>account-number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

<i>customer-id</i>	<i>account-number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

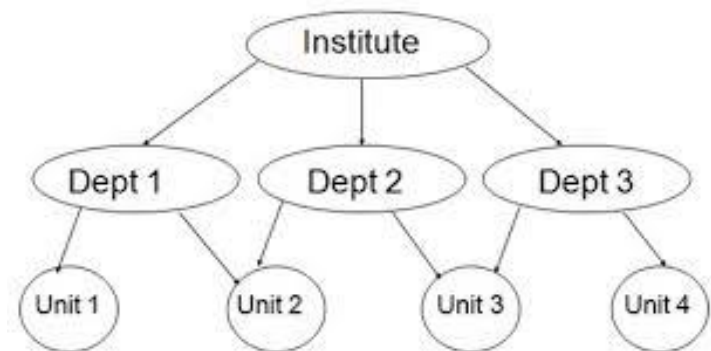
(c) The *depositor* table

# Hierarchical Databases

- Data is organized into a tree-like structure
- The hierarchical structure was developed by IBM in the 1960s, and used in early mainframe DBMS.
- This structure is simple
- However, it is inflexible as the relationship is confined to a one-to-many (1:N) relationship.
- There is a hierarchy of parent and child data segments.

# Network Databases

- Represent one-to-one and many-to-many (more than one parent per child) relationships in data
- In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model.
  - CODASYL is a consortium to guide the development of a standard programming language to be used many computers



# Data Definition Language (DDL)

- Specification notation for defining the database schema
  - E.g.  

```
create table account (  
    account-number char(10),  
    balance integer)
```
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - Data *storage and definition* language
    - language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language

# 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
  - 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
- SQL is the most widely used query language

# SQL

- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465

```
select  customer.customer-name
from    customer
where   customer.customer-id = '192-83-7465'
```
  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465

```
select  account.balance
from    depositor, account
where   depositor.customer-id = '192-83-7465' and
         depositor.account-number = account.account-number
```



# 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
  - E.g. 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
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

# Types of DBMS (based on number of users)

- Single-user DBMS

- Database resides on one computer and is only accessed by one user at a time.
- This one user may design, maintain, and write database programs.

- Multi-user DBMS

- Due to large amount of data management most systems are multi-user.
- A database is integrated when the same information is not recorded in two places.
- E.g. both the Library department and the Account department of the college database may need student addresses.

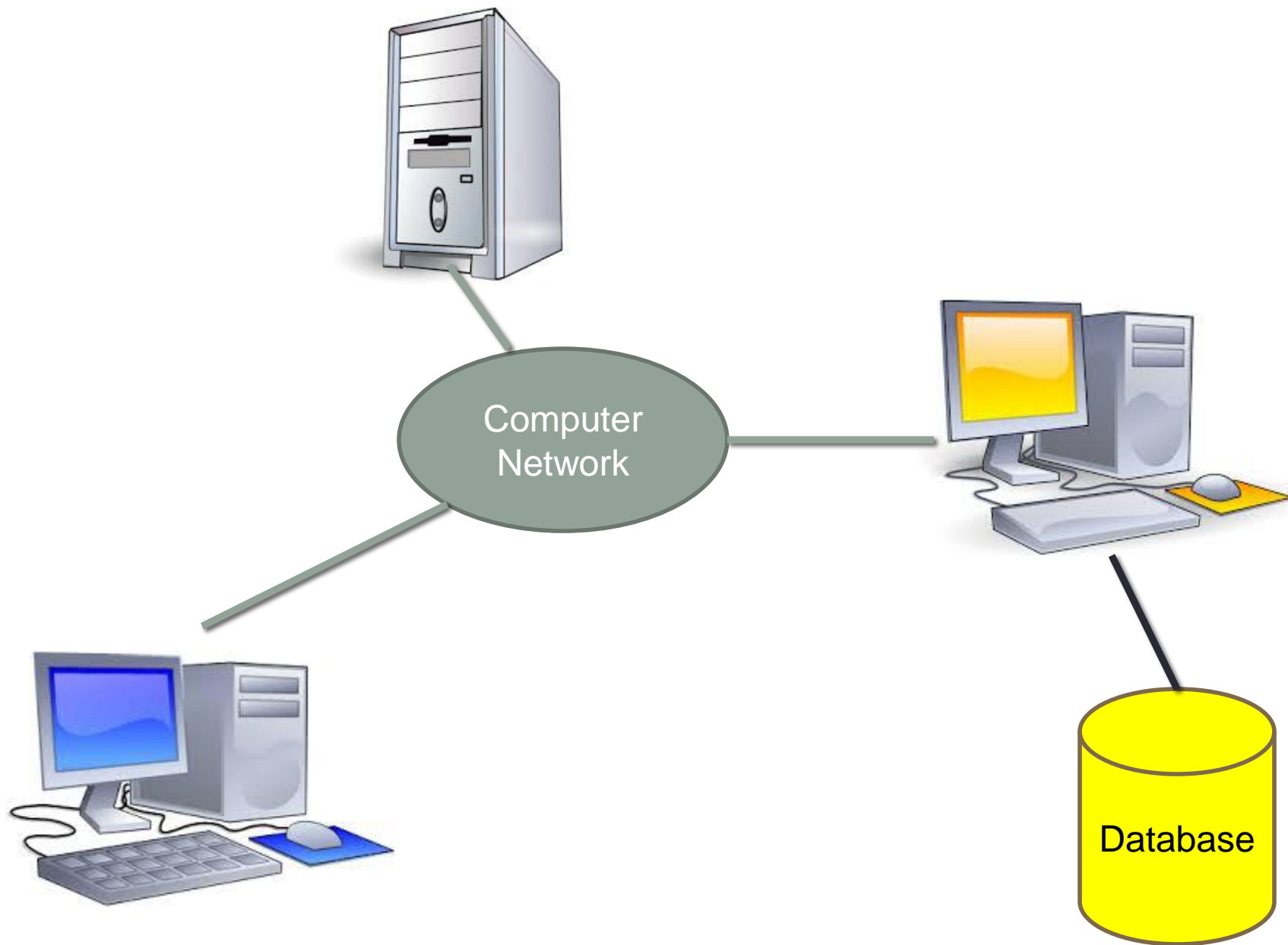
Even though both departments may access different portions of the database, the students' addresses should only reside in one place.

# Types of DBMS (based on location)

- Centralized DBMS
- Distributed DBMS
- Client/Server DBMS

# Centralized DBMS Architecture

- All the DBMS functionality, application program execution, and user interface processing were carried out on one machine
  - E.g. desktop or server CPU, or a mainframe computer.
- A database that is located, stored, and maintained in a single location.
- Access via a communications network
  - LAN
  - WAN
- E.g.
  - Some major banks do all their processing on a mainframe
  - Airline reservation systems need to be centralised to avoid double bookings.



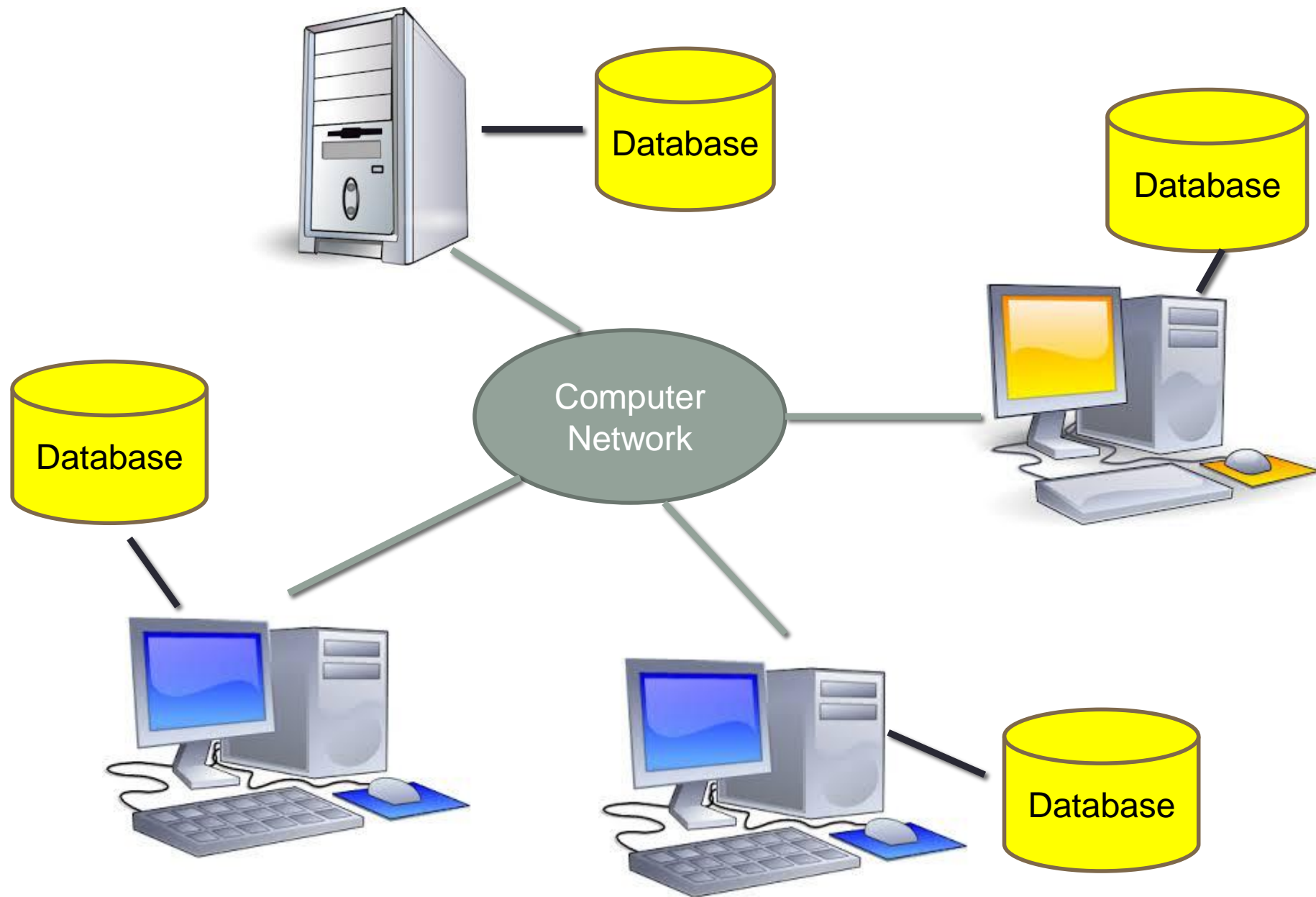
# Advantages/Disadvantage

- Advantages
  - **Reduced redundancy** good planning can allow duplicate or similar data stored in different files for different applications to be combined and stored only once.
  - **Improved availability** information may be made available to any application program through the use of the DBM
  - **Reduced inconsistency** if the same data is stored in more than one place, then updating in one place and not everywhere can lead to inconsistencies in the database.
  - **Enforced data security** authorization to use information can be centralized.
- Disadvantage:
  - When the central site computer or database system goes down, then every one (users) is blocked from using the system until the system comes back.
  - Communication costs from the terminals to the central site can be expensive

# Distributed Database

- A single logical database that is spread physically across computers in multiple locations that are connected by a data communications link.
- In distributed databases, most of the processing is done locally
- It need to consider local ownership of data (e.g. region/country specific data protection laws)
- This requires data sharing
- Users think they are working with a single corporate database
- Example
  - Chain Stores has their individual databases and may be updates monthly/annually with the central system
  - Google: Use a DBMS called Bigtable. (Note it is not a Relational Database).





# Advantages/Disadvantages

- Advantages:

- Distributed database architecture provides greater efficiency and better performance.
- As data volumes and transaction rates increase, users can grow the system incrementally.
- It causes less impact on ongoing operations when adding new locations.
- Distributed database system provides local autonomy.

- Disadvantage:

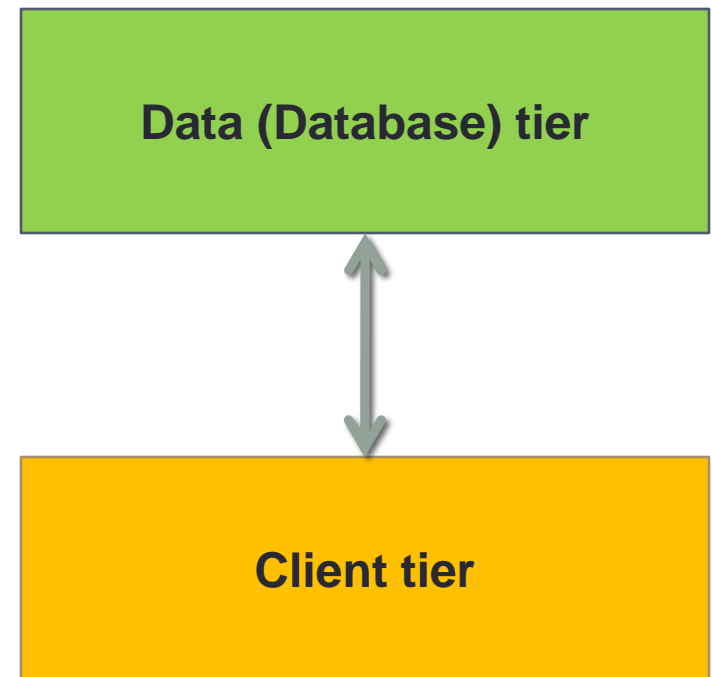
- Recovery from failure is more complex in distributed database systems than in centralized systems.

# Basic Client /Server Architecture

- Two logical components as client and server
- Server and client computers are connected into a network.
- Clients (front end)
  - Generally personal computers or workstations.
  - Applications and tools of DBMS run on one or more client platforms and make requests for DBMS services
- Server (back end)
  - Large workstations, mini range computer system or a mainframe computer system.
  - DBMS software reside on the server
  - The DBMS, in turn, processes these requests and returns the results to the client(s)

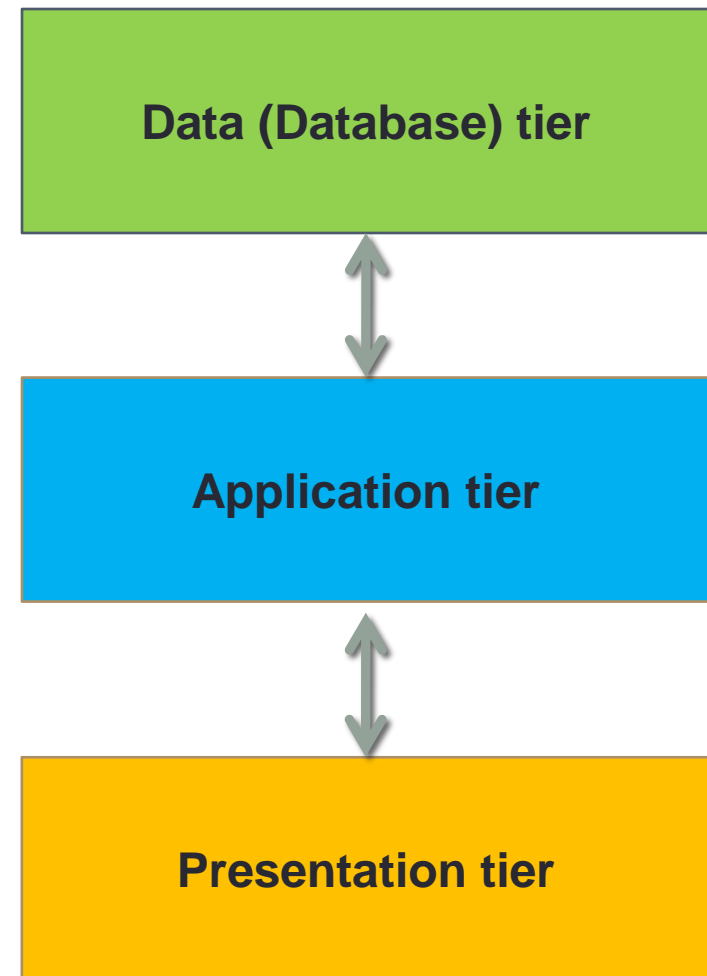
# Two-Tier Client/Server Architecture for DBMS

- The direct communication takes place between client and server.
- There is no intermediate between client and server.

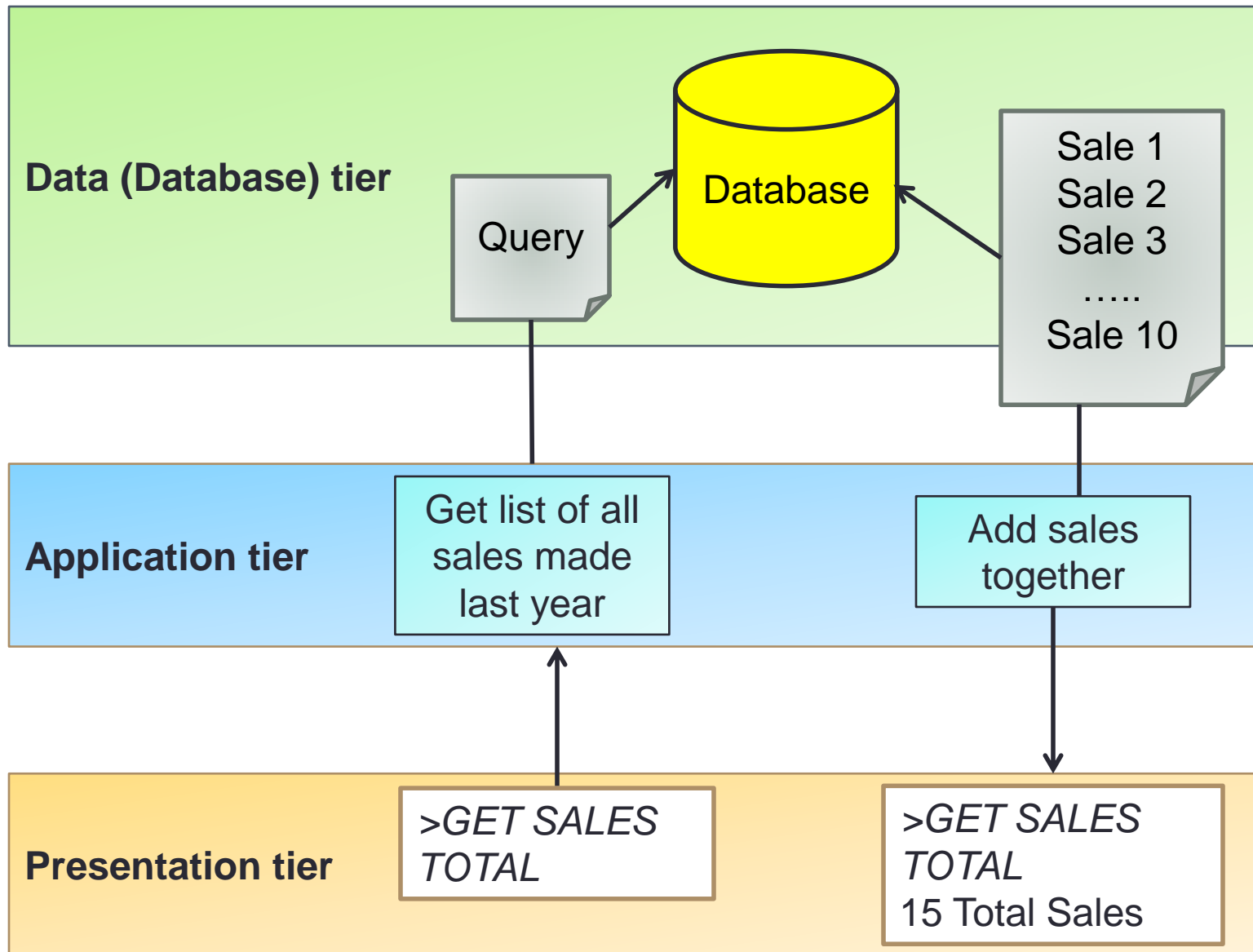


# Three-tier client/Server Architecture for Web Applications

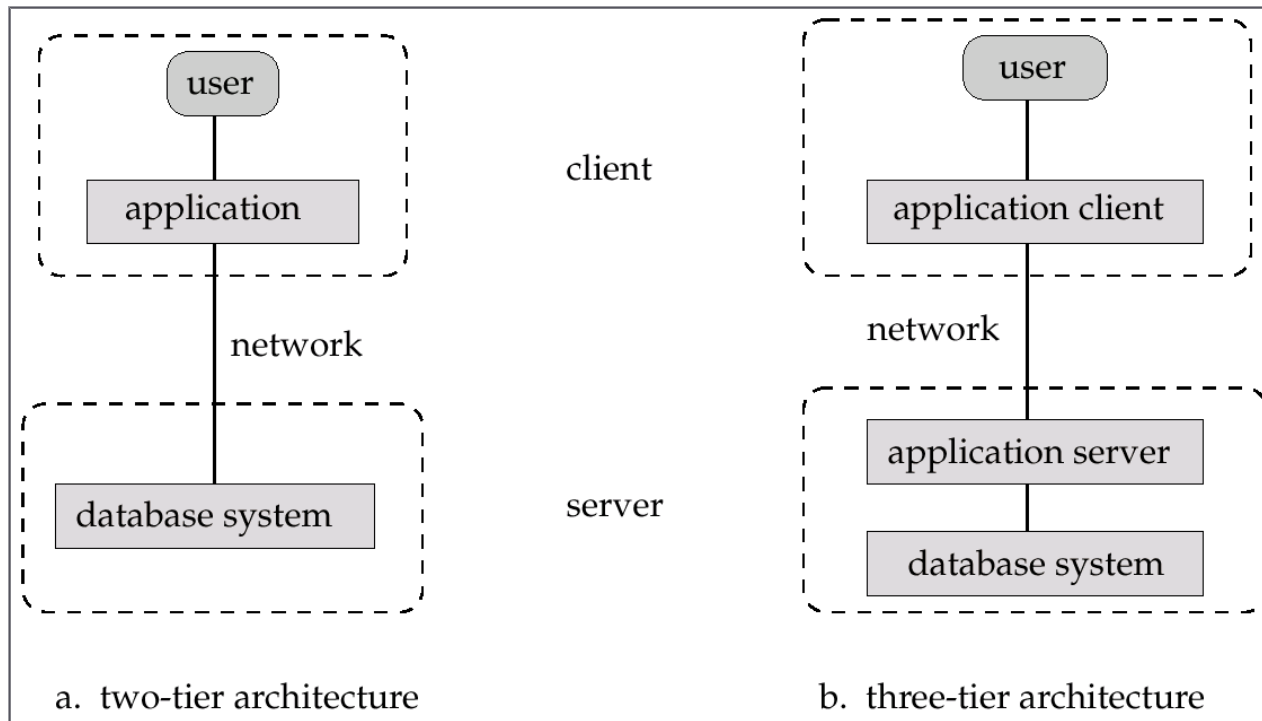
- Separated into 3 tiers based on the complexity of the users and how they use the data present in the database.
- Database (Data) Tier
  - Data resides in this tier
- Application (Middle) Tier
  - Application server and the programs that access the database resides here.
  - This is pulled out from user (presentation) tier
- User (Presentation) Tier
  - End-users operate on this tier



- **User (Presentation) Tier** – End-users operate on this tier
  - At this layer, multiple views of the database can be provided by the application.
  - All views are generated by applications that reside in the application tier.
  - *E.g. User asks for “get total sales” made last year through the interface and in return get the results “total sales” for last year*
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database.
  - End-users are unaware of any existence of the database beyond the application.
  - Also, the database tier is not aware of any other user beyond the application tier.
  - This tier coordinates requests and perform detailed processing
  - This tier could exist in a server machine
  - Importantly this is not tied to a specific client as such could be used by many views
  - *E.g. For user request ‘get list of all sales made last year’ and then add them together and send it to presentation tier*
- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages.
  - Data storage and retrieval takes place
  - *E.g. Query the database and then all sales data relevant to last year is sent to the application tier to process (to get total sales)*



# Application Architectures



- **Two-tier architecture:** E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture:** E.g. web-based applications, and applications built using “middleware”