CHAPTER ONE

“Introduction to Database System

Overview

Data is the cornerstone of any modern software application, and databases are the most common way to store and manage data used by applications.

Data Vs Information

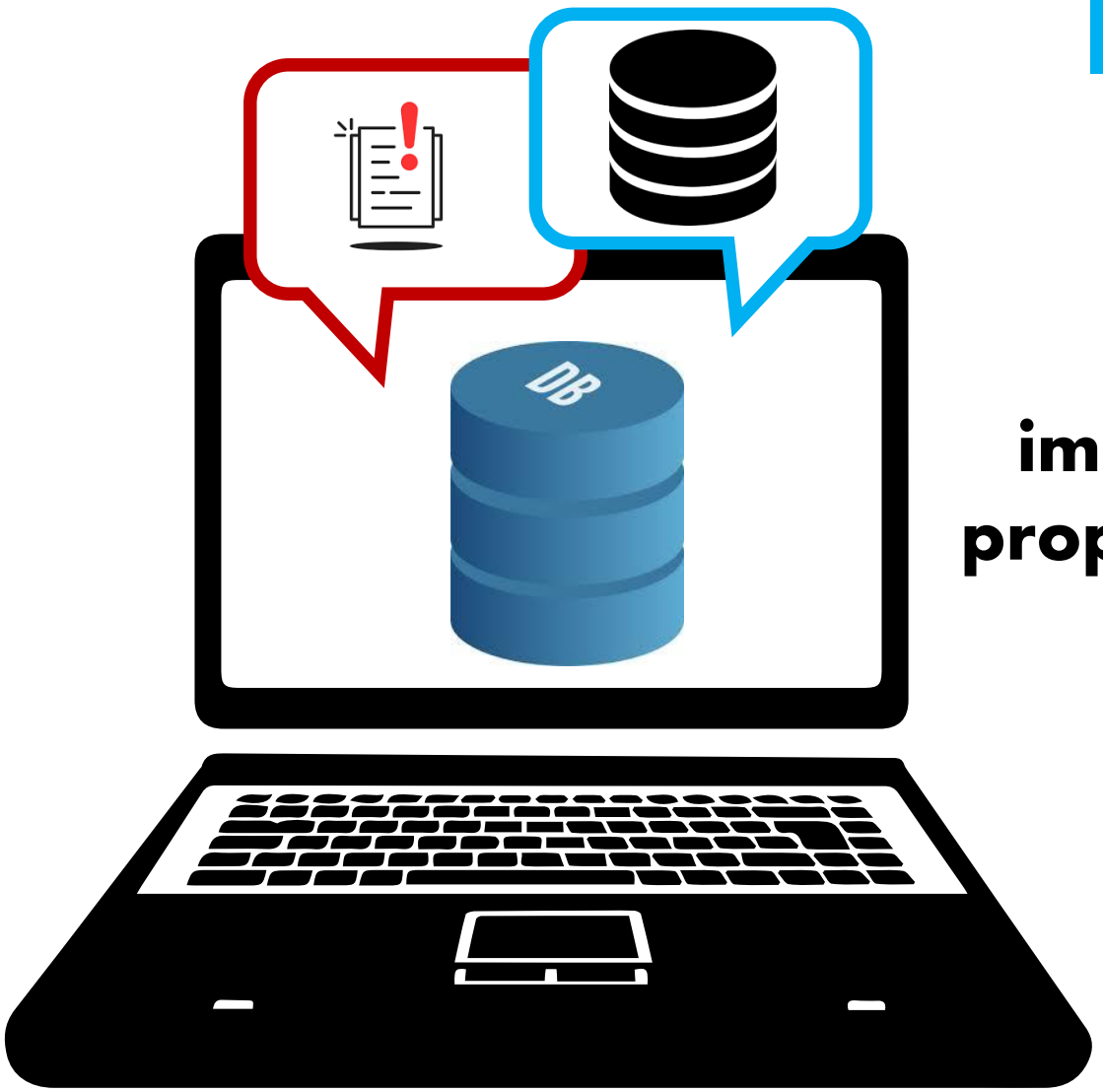
Data

- Raw fact

Information

- Processed form of data

- In order to convert data into useful information a set of software tools are need



Database

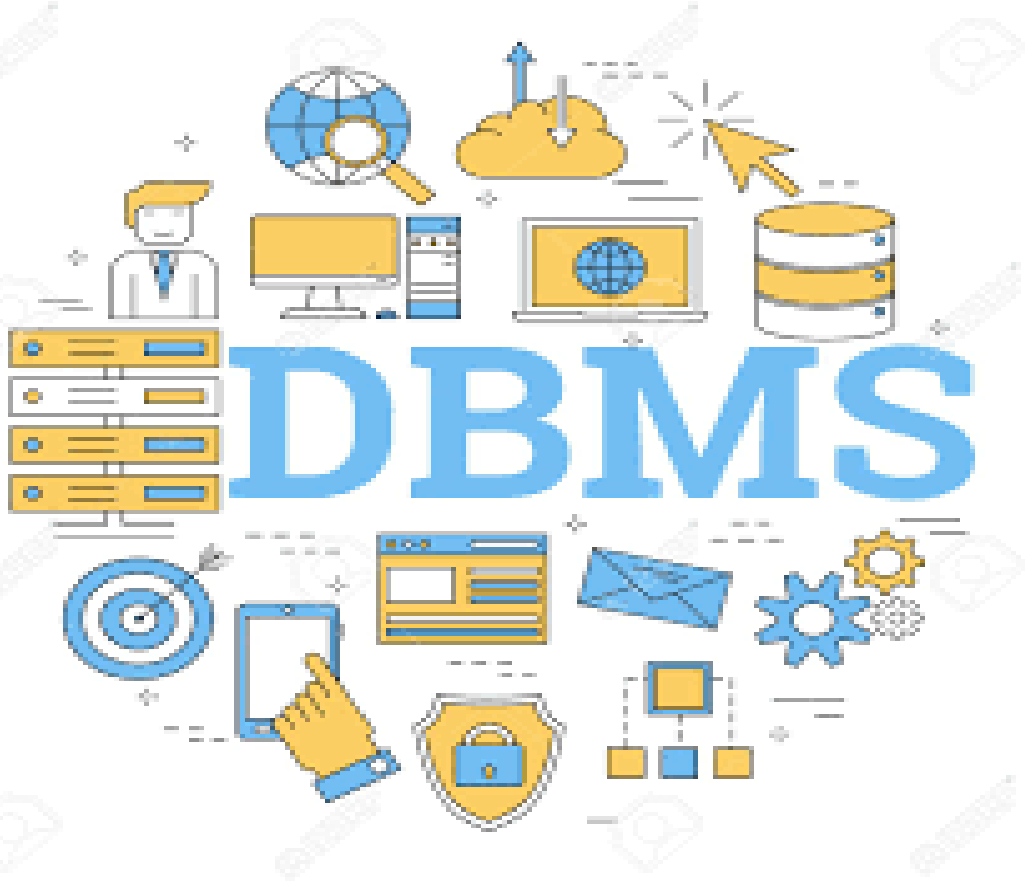
❖ Collection of related data

implicit properties

- ✓ *represents some aspect of the real world*
- ✓ *is a logically coherent collection of data*
- ✓ *is designed, built, and populated with data for a specific purpose*

Database system

Database management system



DBMS also provides the service of controlling ***data access***, enforcing ***data integrity***, managing ***concurrency control***, and ***recovery***.

Defining

- ❖Data type
- ❖Structure
- ❖Constraint

Constructing

- ❖ Process of storing the data

Manipulation

- ❖Querying
- ❖Updating
- ❖Generating report

Data Management Levels

Data storage and retrieval will be performed using human labour

- File
- Cabinet
- Index

Limitations:

- Prone to error
- Difficult to update, retrieve, integrate
- Difficult to compile the information
- Limited to small size information

Data Management Levels

Decentralized

Limited data sharing

❖**Limitations:**

❖Data Redundancy

❖Separation or Isolation of Data

❖Lengthy dev’t and maintenance time

❖Incompatible file formats

❖“update anomalies”

Data Management Levels

Database

Centralized

Advantage:

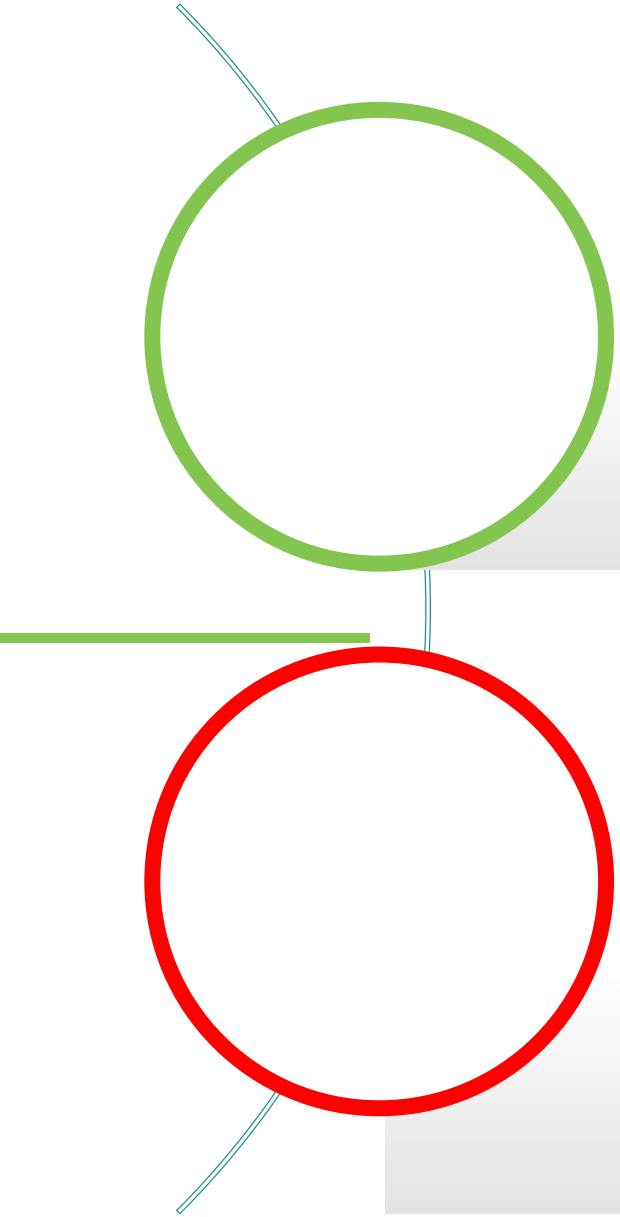
- ❖Shared collection
- ❖Collection of logically related data
- ❖Redundancy can be reduced
- ❖Inconsistency can be avoided

Characteristics of database Approach

Catalog: contains information such as the structure of each file, the type and storage format, and various constraints **meta-data**

- **Self-Describing Nature of a Database System**
- **Insulation between Programs and Data**
- **Support of Multiple Views of the Data**
- **Sharing of Data and Multiuser Transaction Processing**

Actors on the Scene



Database Administrators

- Manage Resource
- Authorization
- Accountable for poor system response
- Acquire resource

Database Designers

- Identify data
- Choose structure
- Communicate with stakeholders

Actors on the Scene

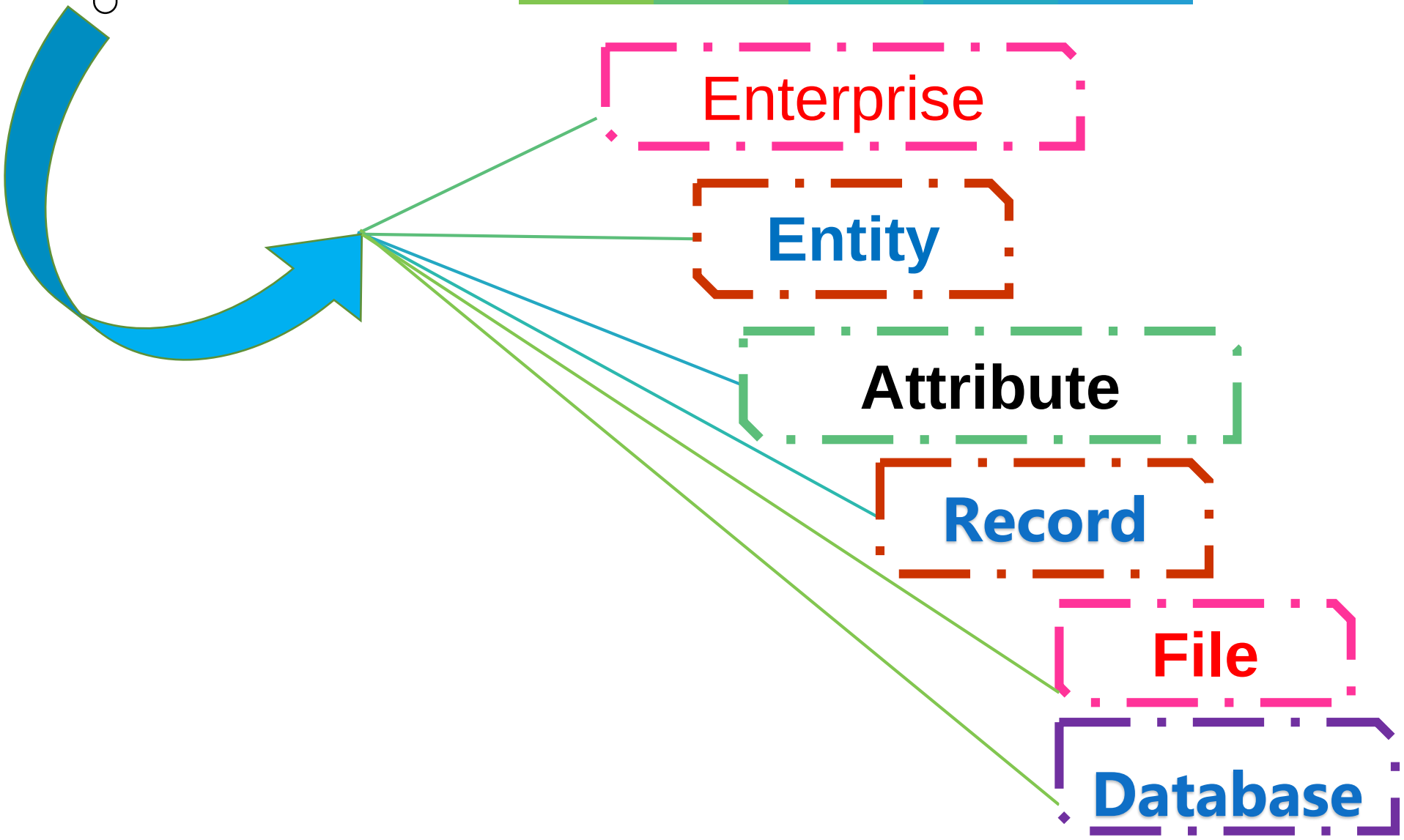
Software Engineers

- System analyst
  - Identify requirement
  - Develop specification
- Programmer
  - Convert specification to code

End users

- **Casual:** occasional users
- **naive:** use the database often
- **Sophisticated:** capable of solving their problem using DBMS
- **Stand alone:** use ready made package

Basic Database Terminologies



CHAPTER Two

“

Database System concepts & Architecture

Database System Architecture

01 Client-server Architecture

02 Data Models, Schemas, and Instances

03 Three-schema Architecture

04 Entity, Attribute and Relationship

Client-server Architecture

System functionality is distributed between two types of modules

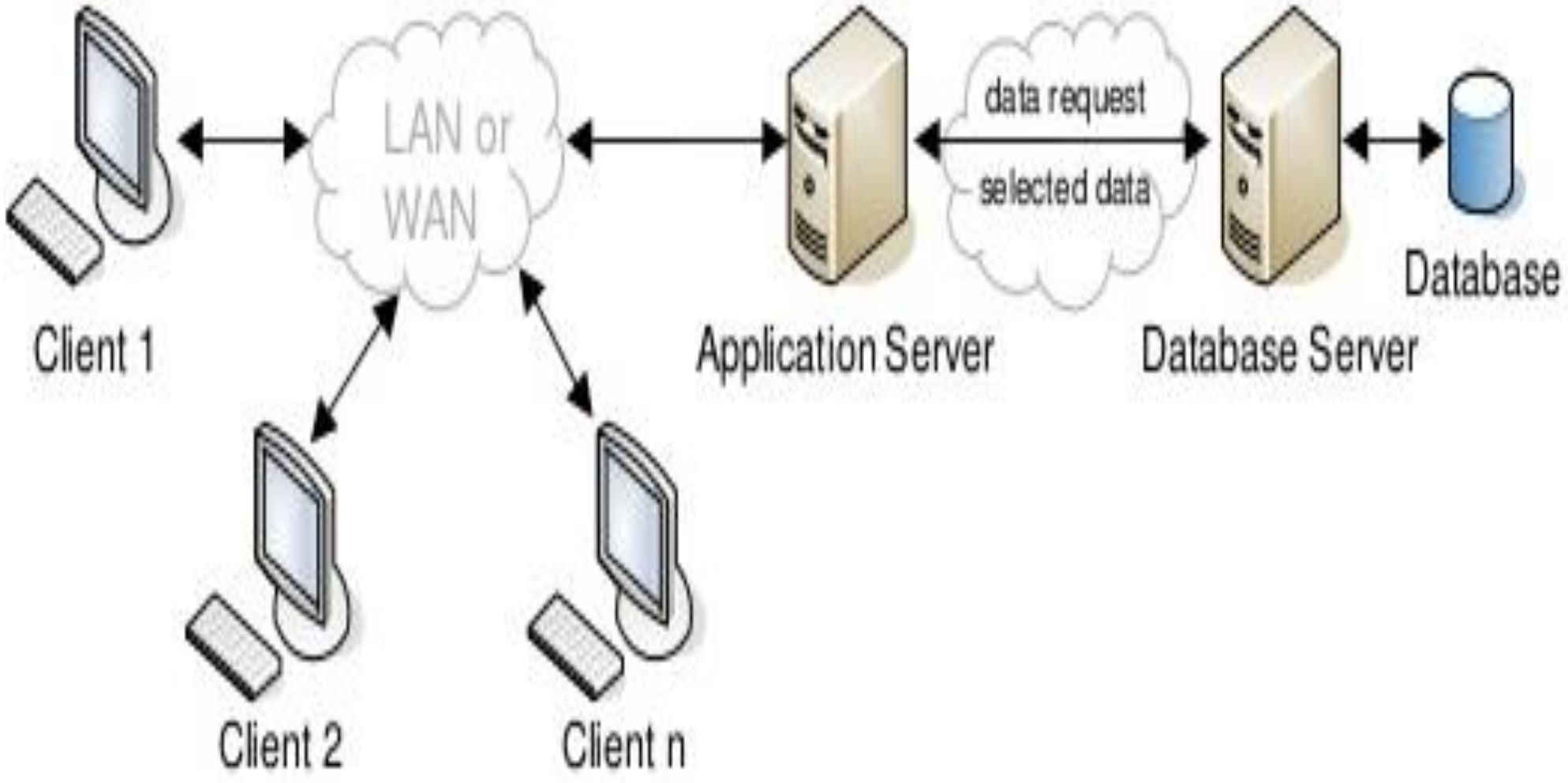
Client Module

- ✓UI
- ✓Application program

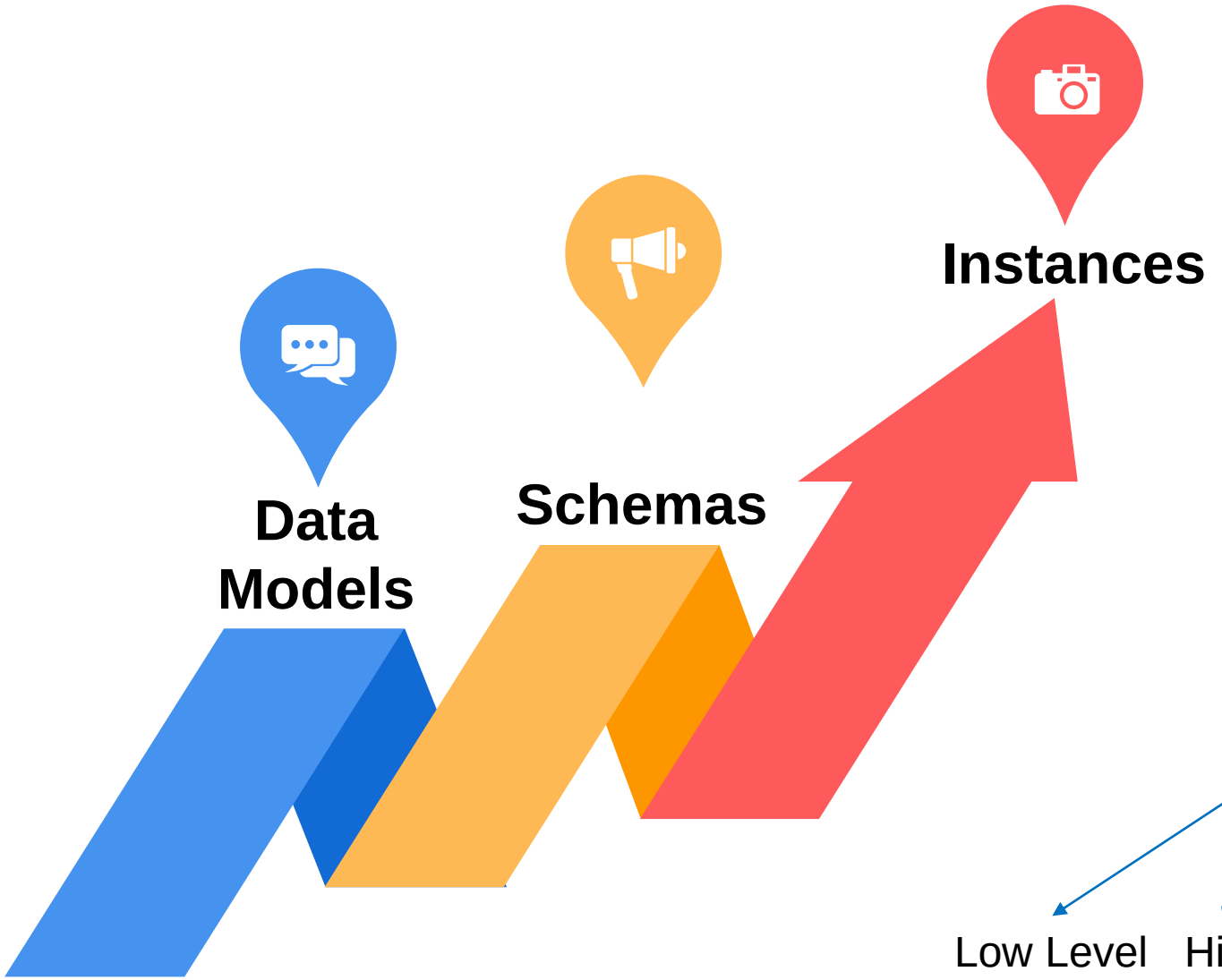
Server Module

- ✓Handles data storage, access…

Cont…


LAN or
WAN
data request
selected data
Client 1
Application Server
Database Server
Database
Client 2
Client n

Data Models, Schemas, and Instances



Instances

✓ Data at a particular moment

Schemas

✓ not expected to change frequently

Data Models

✓ describe the structure of a database

Three-schema Architecture

- separate the user **applications** and the **physical database**

Internal level

- details of data storage and access paths for the database

Conceptual level

- describes the structure of the whole database

External level

- describes the part of the database

Data Independence

capacity to change the schema at one level of a database system without affecting the next higher level

Logical data independence

Change Conceptual Level

Why we need to change this schema?

❖ to expand the database or to reduce the database

Physical data independence

Change Internal Level

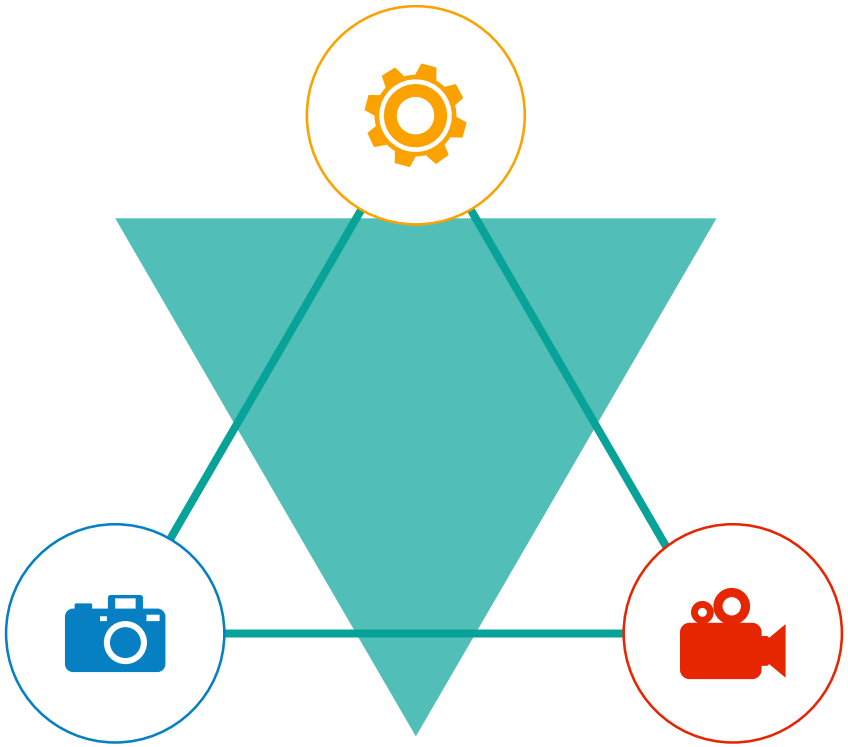
Why we need to change this schema?

❖ to improve the performance of retrieval or update

Entity, Attribute and Relationship

Entity

- Strong
- Weak



Attribute

- Simple vs Composite
- Single-value vs Multi-valued
- Stored vs Derived
- Null Values

Relationship

- Degree
- Cardinality Ratios
- Participation Constraints

Cardinality ratio: specifies the number of relationship instances that an entity can participate in

One-to-one

each tuple in one database table is linked to 1 and only 1 other tuple in another table.

One-to-Many

one tuple can be associated with many other tuples, but not the reverse.

Many-to-Many

one tuple is associated with many other tuples and from the other side

Participation constraint: existence dependency

Total Participation

"the total set"

Partial Participation

some or "part of the set of"

Integrity Rules and Keys

Keys

Primary key
values that uniquely identify each row in a table

Candidate key
unique identifier for a record within a table that can be chosen as the primary key

Foreign key
refers to the unique data values -- often the primary key data -- in another table

Cont…

Integrity Rule

Entity integrity rule

Primary key should not be null

Referential integrity rule

Foreign key should be either null or value that match with primary key

Domain integrity rule

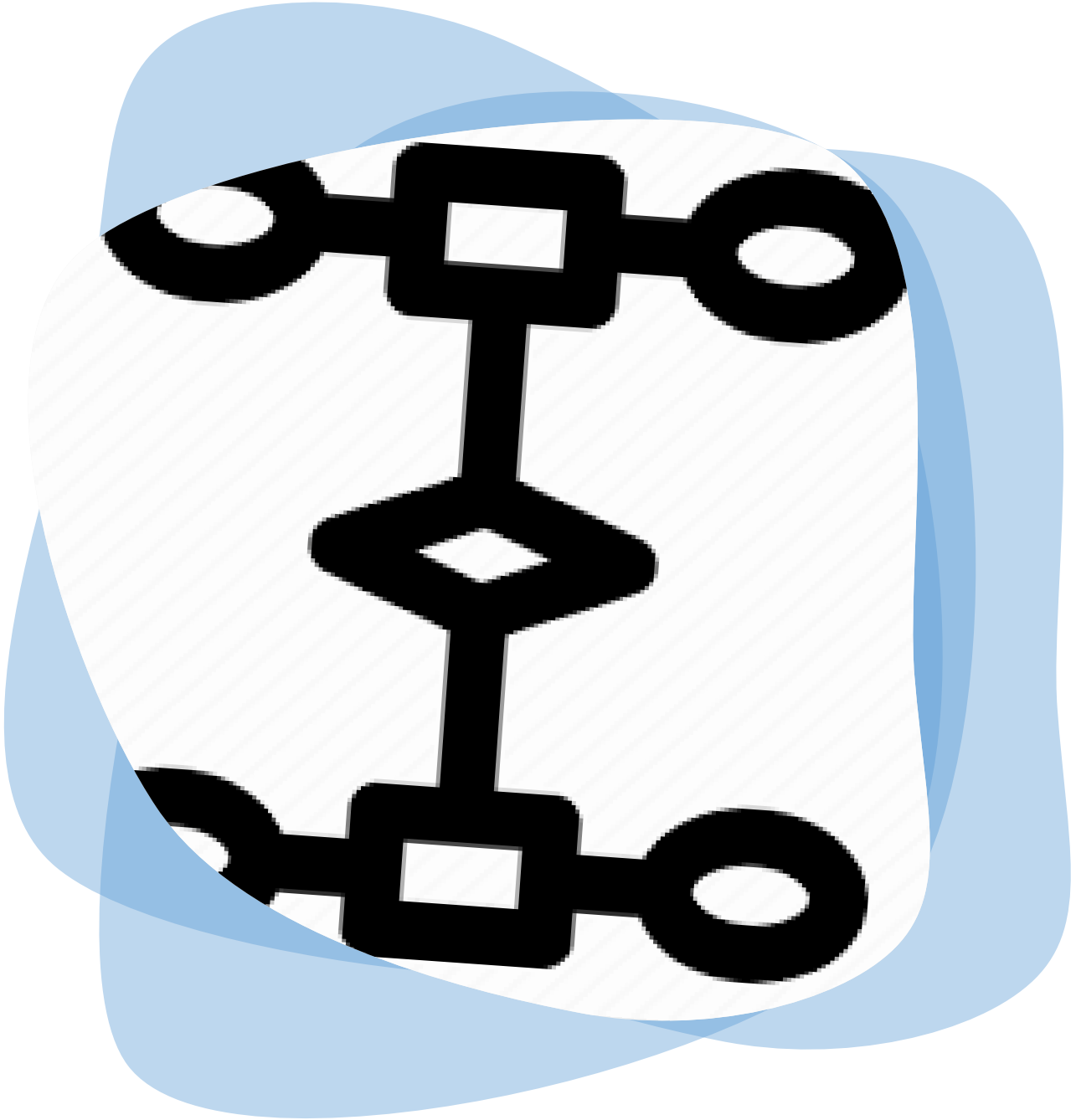
Value should not be beyond limit

Enterprise integrity rule

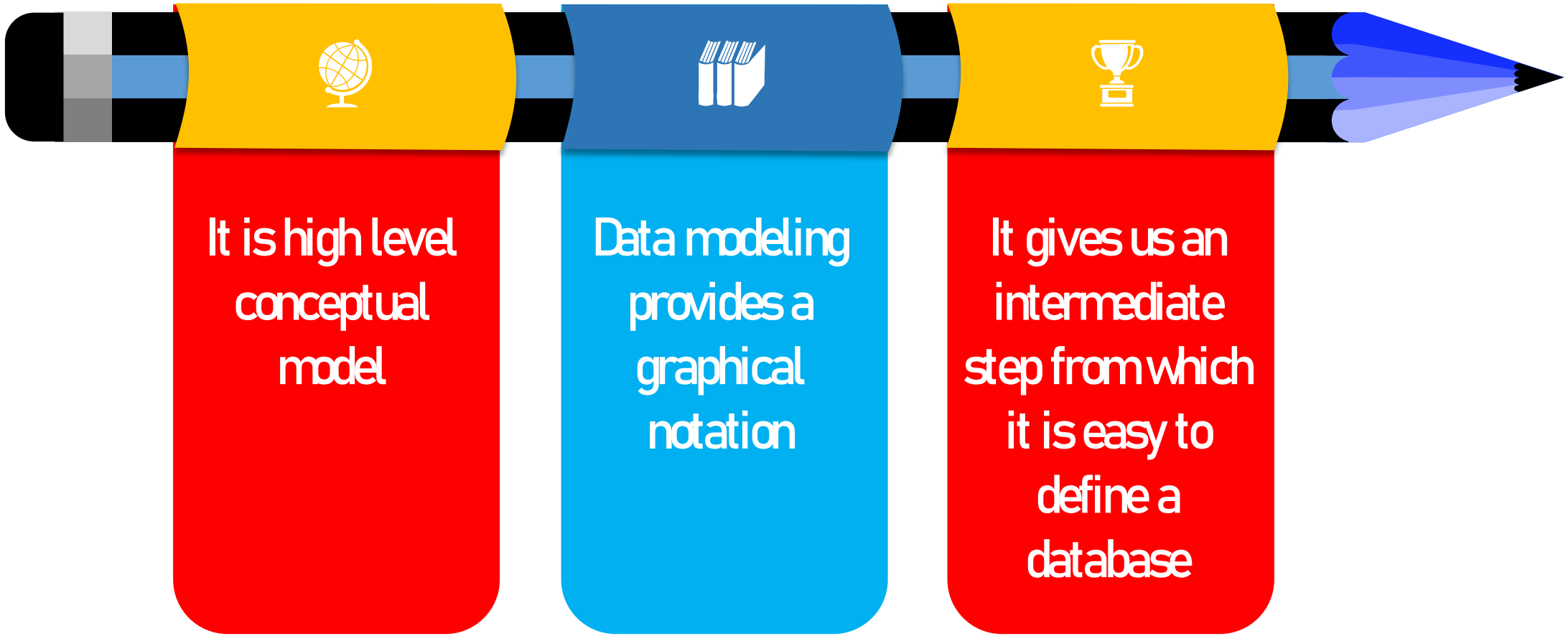
Any additional rule by user or DBA

CHAPTER THREE

“Data modeling using ER-Diagram



ER Modeling



Main components of ER Modeling are:

❖ *Entity* ❖ *Attribute* ❖ *Relationship*

One has to know and answer the following basic questions.

02 What information about these entities and relationships should we store in the database?

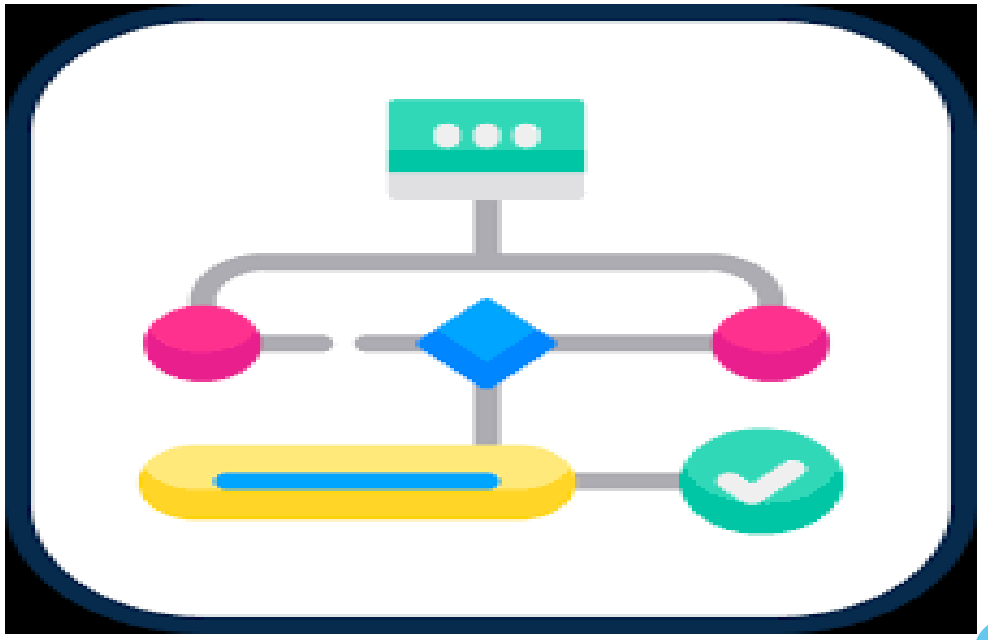
03 What are the integrity constraints that hold?

01 What are the entities and relationships in the enterprise?

04 Represent this information pictorially in **ER** diagrams

Steps to

Draw an ER Diagram



01

Identify Entities

- ✓Classify them as strong or weak
- ✓Diagramming entities

02

Identify the attributes

- ✓Decide types of attributes
- ✓Diagramming attributes

03

Establish relationships

- ✓ Diagramming relationships

Draw final ER-Diagram

ER Diagram

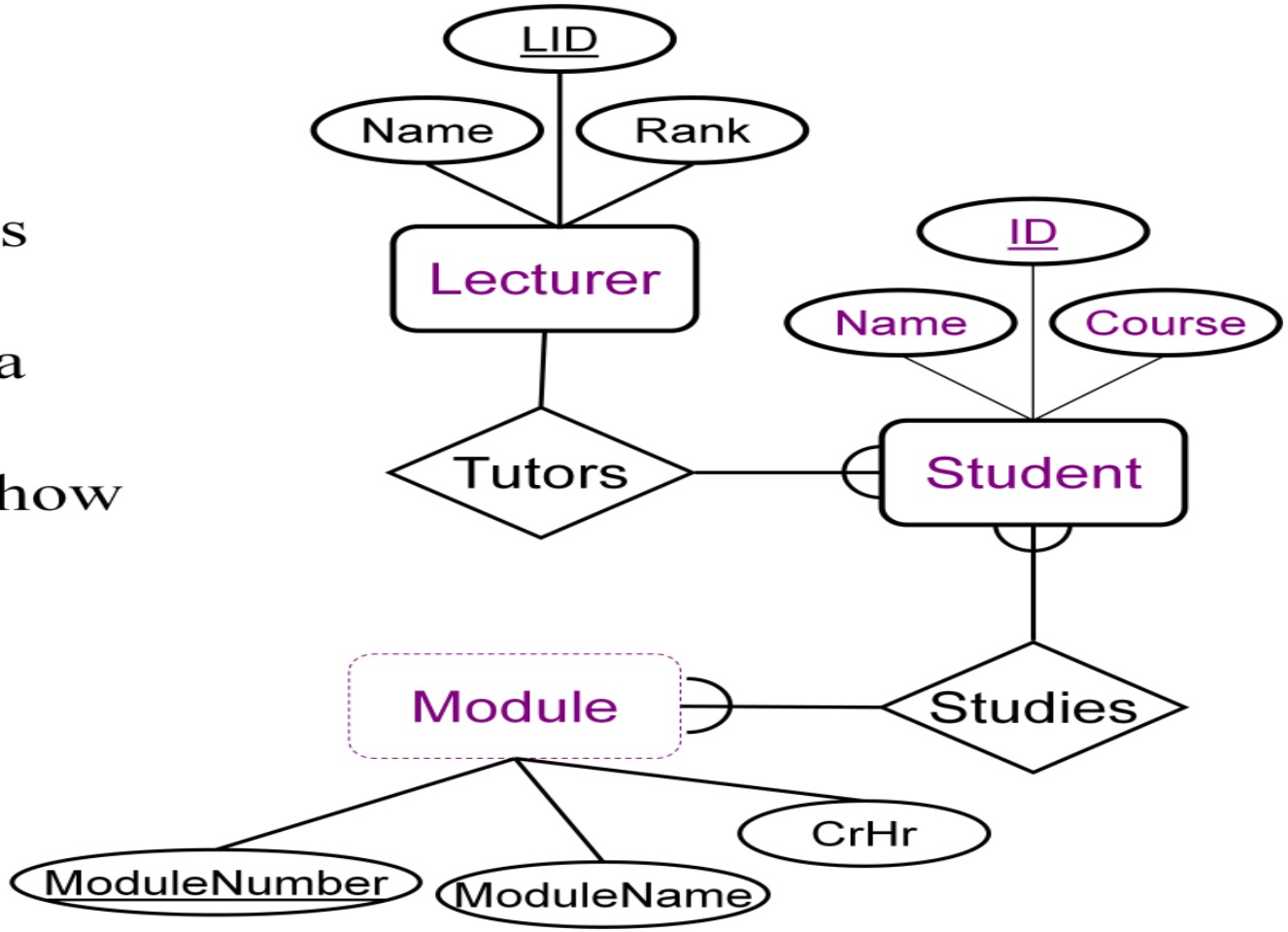
Requirements of the university

Oversimplified for illustrative purposes

In a University database we might have entities for **Students**, **Modules** and **Lecturers**. Students might have attributes such as their ID, Name, and Course whereas Lecturers might have attributes such as LID, Name, Rank and Modules might have attributes such as ModuleNumber, ModuleName, CrHr and could have relationships (studies) with Student and Modules and (*tutor*) with Lecturers.

ER Diagram for Company

- Relationships are links between two entities
- The name is given in a diamond box
- The ends of the link show cardinality



Requirements of the Company

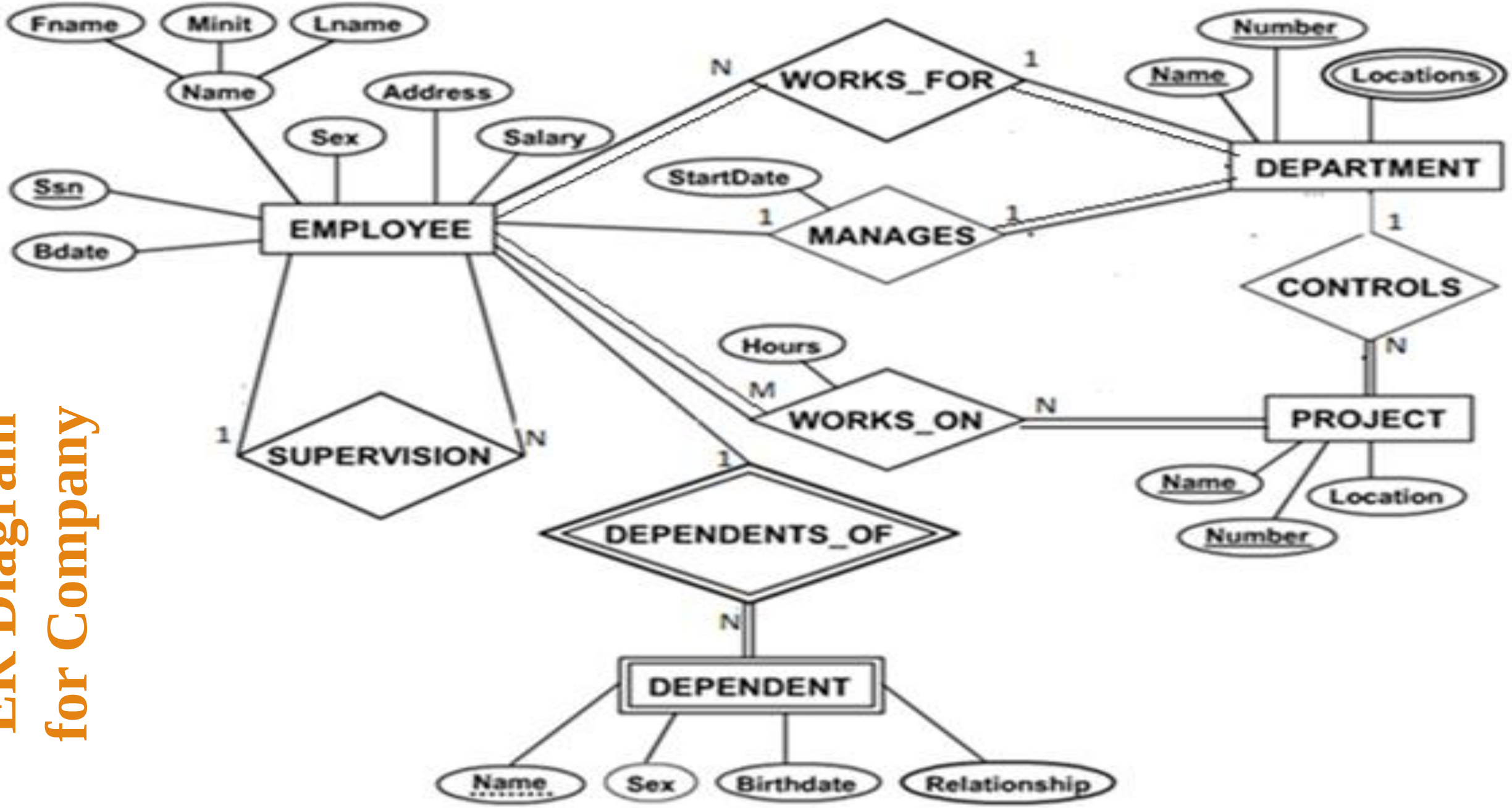
The company is organized into departments. Each **department** has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

A department controls a number of **project**s, each of which has a unique name, a unique number, and a single location.

We store each **employee**'s name, social security number, address, salary, sex, and birth date. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.

We want to keep track of the dependents of each employee for insurance purposes. We keep each **dependent**'s first name, sex, birth date, and relationship to the employee.

ER Diagram for Company



Mapping of Entity Types

- ✓includes all the **simple** attributes
- ✓Choose one of the key attributes as a primary key

Strong Entity

- ✓ includes all the **simple** attributes
- ✓ Include as a foreign key the primary key of owner entity

Weak Entity

Mapping of Relationship Types

- ✓identify the two relations **S** and **T**
- ✓Choose one of the relations- **S** (total participation)
- ✓include as a foreign key in **S** the primary key of **T**
- ✓Include all **simple** attributes as part of relation **S**

1:1 Relationship

- ✓identify the two relations **S** and **T**
- ✓Choose one of the relations- **S** (many side)
- ✓include as a foreign key in **S** the primary key of **T**
- ✓Include all **simple** attributes as part of relation **S**

1:N Relationship

Mapping of Relationship Types

- ✓Create a ***new relation*** as ***S***
- ✓include as a foreign key in **S** the primary key of participating entity types
- ✓Their combination will form the primary key of **S**
- ✓Include all **simple** attributes as part of relation **S**

M:N Relationship

- ✓Create a ***new relation***
- ✓Include multi valued attribute as part of ***Relation***
- ✓ include as a foreign key in **Relation** the primary key of the parent relation
- ✓Their combination will form the primary key of **Relation**

Multi valued Attribute

CHAPTER FOUR

“

Functional Dependencies and Normalization

Informal Design Guidelines for Relation Schemas

❖ four informal measures of quality for relation schema design

Semantics of the attributes

❖ certain meaning has to be associated with the attributes

Reducing the redundant values in tuple

❖ to minimize the storage space

Reducing the null values in tuples

❖ waste space at the storage level

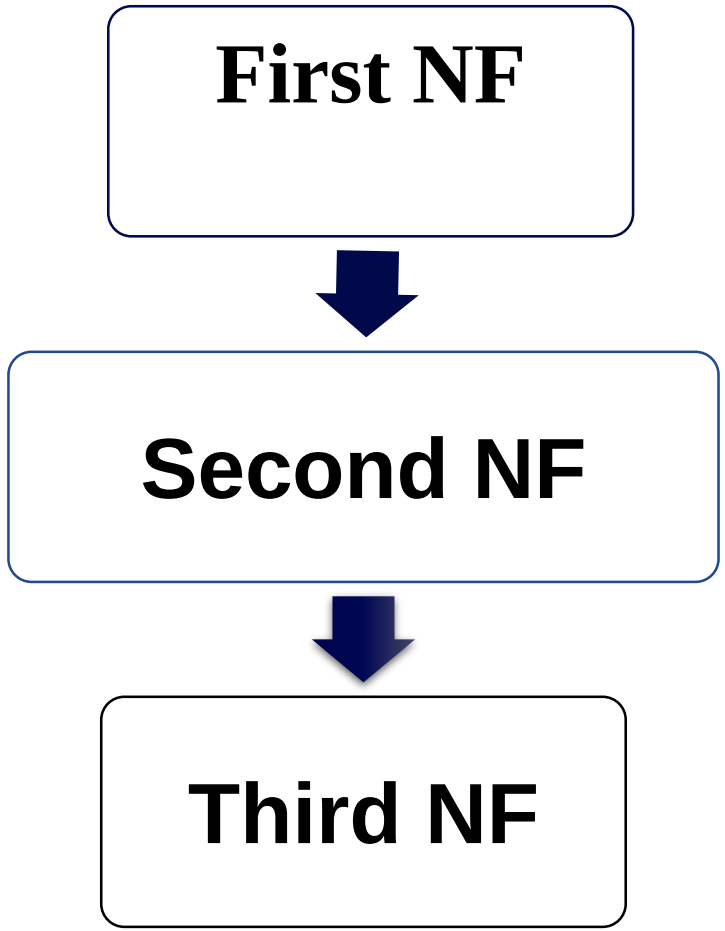
Disallowing generating spurious tuples

❖ avoid matching attributes that are not (foreign key, primary key) combination

Type of Dependency

Partial Dependency	Full Dependency	Transitive Dependency
❖ Let {A, B} is the Primary Key and C is no key attribute. Then if {A, B}➔C and B➔C Then C is partially functionally dependent on {A, B}	❖ Let {A, B} is the Primary Key and C is no key attribute Then if {A, B}➔C and B➔C and A➔C does not hold Then C Fully functionally dependent on {A, B}	❖ If A functionally governs B, AND If B functionally governs C THEN A functionally governs C

Normalization: process of identifying the logical associations between data items and designing a database that will represent such associations



Normalization is formal technique for analyzing a relation based on its:

- Primary key and
- The functional dependencies between the attributes of that relation

Cont…

First Normal Form

- Disallow multivalued attributes, composite attributes, and their combinations.

Three main techniques to achieve first normal form for such a relation:

- Remove the attribute that violates 1NF and place it in a separate relation
- Expand the key
- If a maximum number of values is known for the attribute replace by separate atomic attributes

Cont…

Second Normal Form

- Is based on the concept of full functional dependency.

A relation R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key

NB: Finally decompose and set up a new relation for full functionally dependences.

Cont…

Third Normal Form

- Is based on the concept of transitive dependency.

Relation ***should not*** have a non-key attribute functionally determined by another non-key attribute

NB: Decompose and set up a relation that includes the non key attribute(s) that functionally determine(s) other non key attribute(s)

Exercise: Normalize the given table

EMPLOYEE_ID	NAME	JOB_CODE	JOB	STATE_CODE	HOME_STATE
E001	Alice	J01	Chef	26	Michigan
E001	Alice	J02	Waiter	26	Michigan
E002	Bob	J02	Waiter	56	Wyoming
E002	Bob	J03	Bartender	56	Wyoming
E003	Alice	J01	Chef	56	Wyoming