MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

CET2002B-Database Management Systems

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

CET2002B--Database Management Systems

Course Objectives:

1. Knowledge:

i. Understand the fundamental concepts of database management System
ii. To provide a strong formal foundation in database concepts, DBMS architectures recent technologies and best industry practices.

2. Skills:

i. To learn the SQL database system.
ii. To program PL/SQL including stored procedures, stored functions, cursors and packages.

3. Attitude:

i. To design the database system for real world applications.
ii. To access,modify, program and authenticate the database system.

Module 1- Introduction to Database Management Systems and Data Modeling

Syllabus :

Introduction to DBMS and Data Modeling

DBMS Vs File Systems, Database System Architecture, Database Architectures: Centralized, Client-Server, Parallel, Distributed, Data Abstraction, Data Independence, Data Definition and Data Manipulation Languages, Data Models, E-R diagram: Components of E-R Model, Conventions, Keys, EER diagram Components, Reduce E-R diagram into tables

Assessment Scheme

Assessment Scheme:

Class Continuous Assessment (CCA): 30 Marks

Mid Term	Component 1 (Active Learning)	Component 2
15 Marks	10 Marks	5 Marks

Laboratory Continuous Assessment (LCA): 30 Marks

Practical Performance	Active lerning / Mini Project/Additional implementation/ On paper design	End term practical /oral examination
10 Marks	10 Marks	10 Marks

Learning Resourses

Learning Resources:

Reference Books:

1. Ivan Bayross, "SQL, PL/SQL: The Programming Language of Oracle", BPB Publication
2. Reese G., Yarger R., King T., Williums H, "Managing and Using MySQL", Shroff Publishers and Distributors Pvt. Ltd., ISBN: 81 - 7366 - 465 – X, 2nd Edition

Text Books:

1. Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts 6th Ed, McGraw Hill, 2010.
2. Elmasi, R. and Navathe, S.B., "Fundamentals of Database Systems", 4th Ed., Pearson Education.

Reference Books:

1. Ramakrishnan, R. and Gherke, J., "Database Management Systems", 3rd Ed., McGraw-Hill.
2. Connally T, Begg C.,"Database Systems",Pearson Education

Database Management System Basics

- It contains information about a particular enterprise.
- It provides :
  - Collection of interrelated data.
  - Set of programs to access the collected data.
  - An environment that is both convenient and efficient to use.
  - Databases touch all aspects of our lives.

Database Applications

- Banking: Transactions
- Airlines: Reservations, Schedules
- Universities : Student Registration.
- Sales: Customers, Orders, Products
- Online retailers : Order tracking, recommendations
- Manufacturing :Production, Inventory, Supply Chain
- Human Resources :Employee payroll.

Motivation for Database Management Systems

Traditional file Systems have following drawbacks 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.

Question : What can be other drawbacks related to usage of file systems ?

Motivation for Database Management Systems

Answer : Traditional file Systems have following drawbacks to store data:

- **Integrity Problems**
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Difficult to add new constraints or change existing ones
- **Atomicity of Updates**
  - Failures may leave database in an inconsistent state with partial updates carried out.

Motivation for Database Management Systems

Traditional file Systems have following drawbacks to store data :

- **Concurrent Access by Multiple users**
  - Concurrent access needed for performance.
  - Uncontrolled concurrent accesses can lead to inconsistencies.
- **Security Problems**
  - Hard to provide user access to some, but not all, data

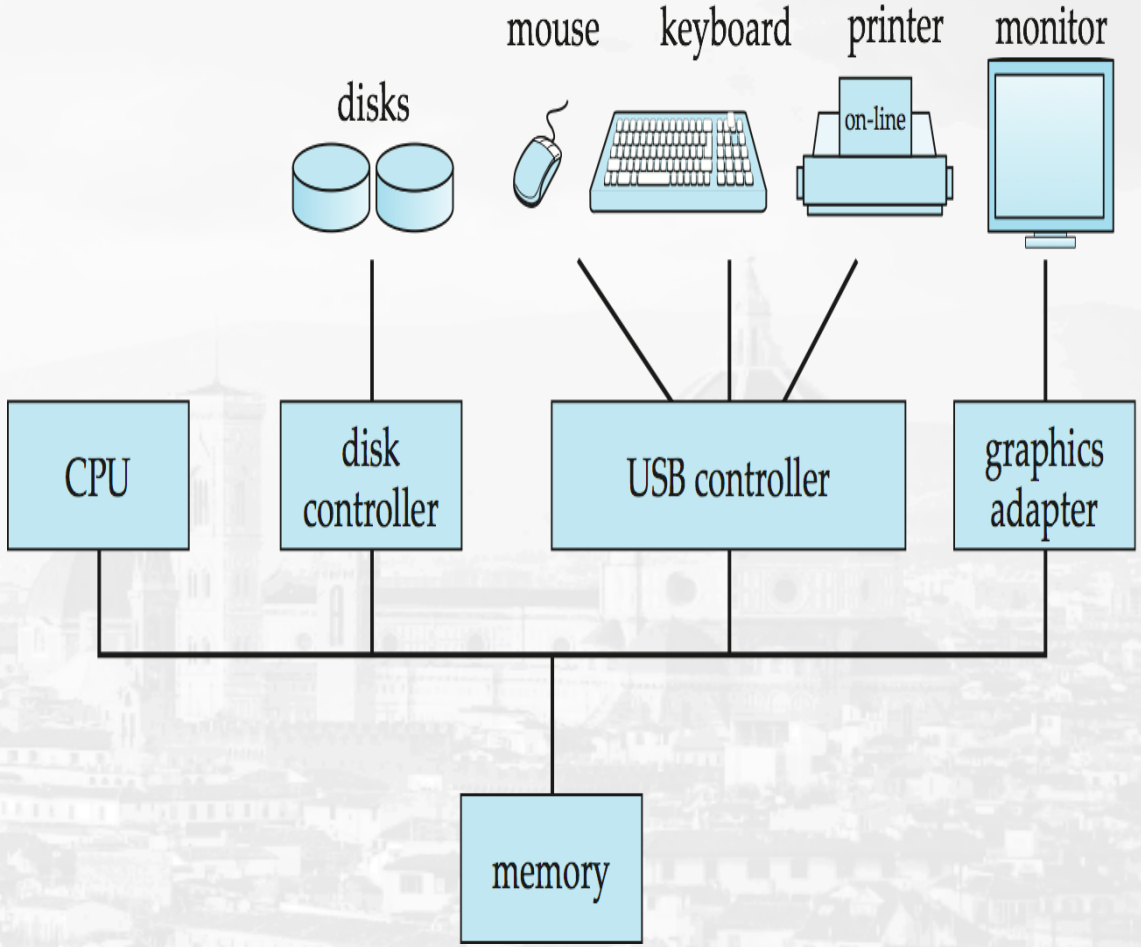
Database Management Systems offers solutions to all the above problems/limitations of traditional file systems.

Database System Architectures

- Centralized Systems
- Client-Server Systems
- Parallel Systems
- Distributed Systems

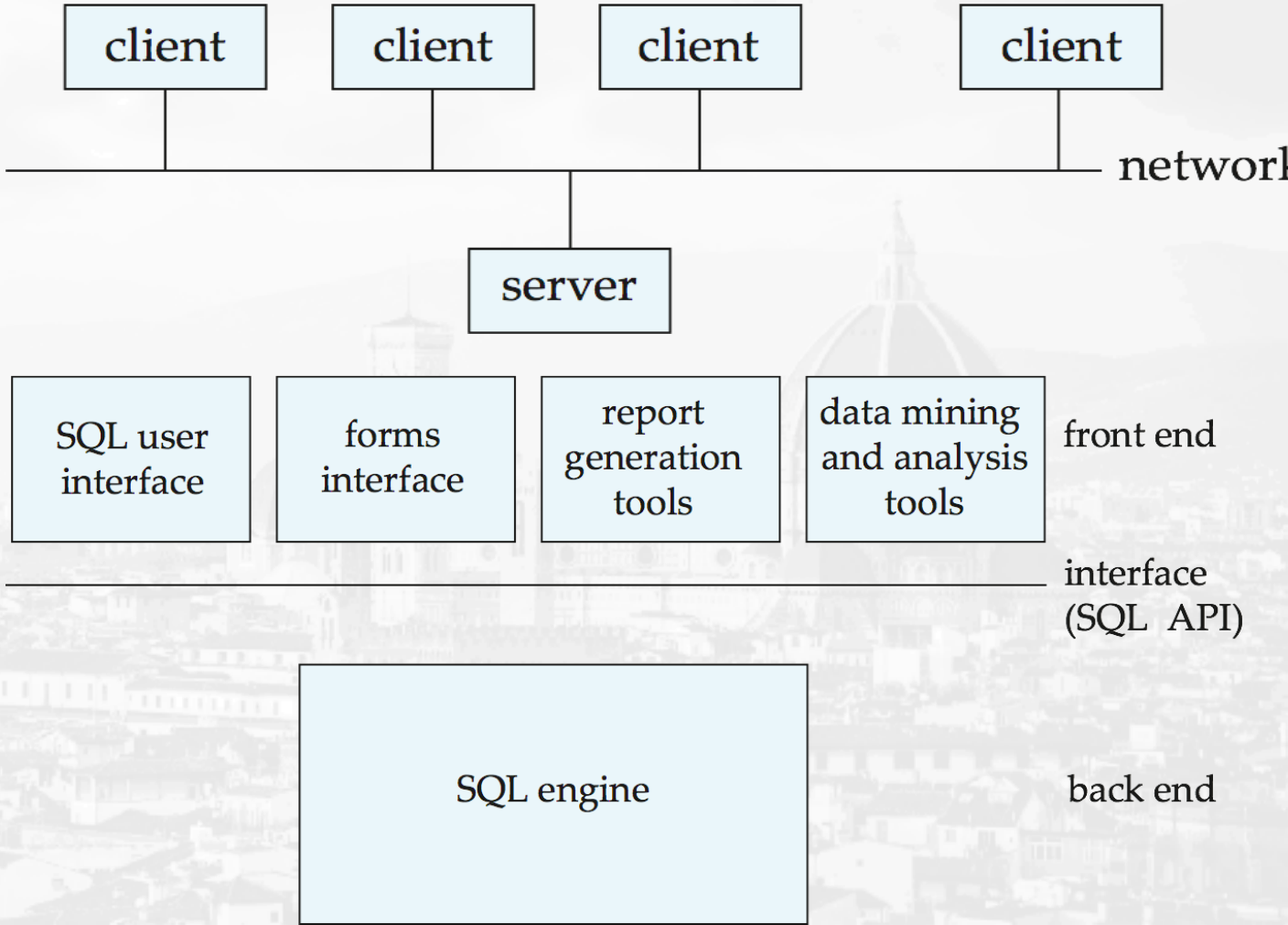
Centralized Systems

- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system vie terminals. Often called *server* systems.



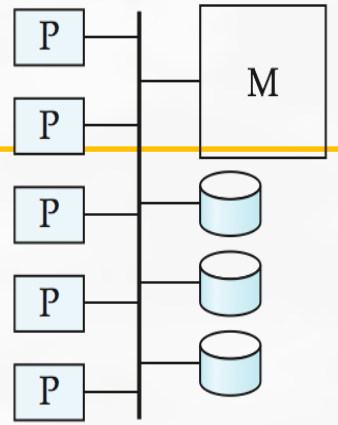
Client-Server Systems

- **Server systems satisfy requests generated at *m* client systems, whose general structure is shown in the dig.**
- **Database functionality can be divided into:**
  - **Back-end:** manages access structures, query evaluation and optimization, concurrency control and recovery.
  - **Front-end:** consists of tools such as forms, report-writers, and graphical user interface facilities.
- **The interface between the front-end and the back-end is through SQL or through an application program interface.**

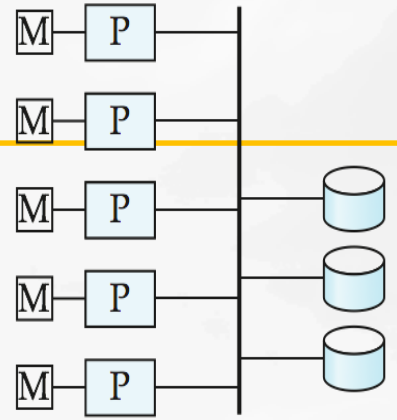


Parallel Systems

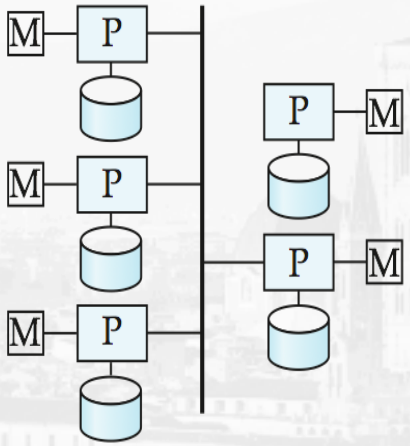
- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
- A coarse-grain parallel machine consists of a small number of powerful processors
- A massively parallel or fine grain parallel machine utilizes thousands of smaller processors.
- Two main performance measures:
  - throughput --- the number of tasks that can be completed in a given time interval
  - response time --- the amount of time it takes to complete a single task from the time it is submitted



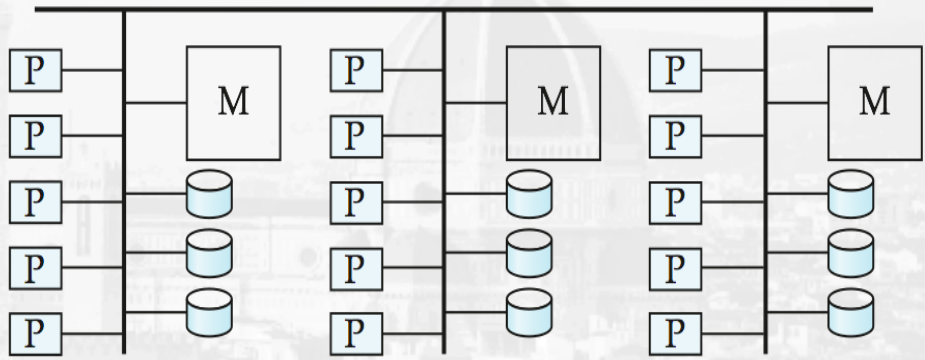
(a) shared memory



(b) shared disk



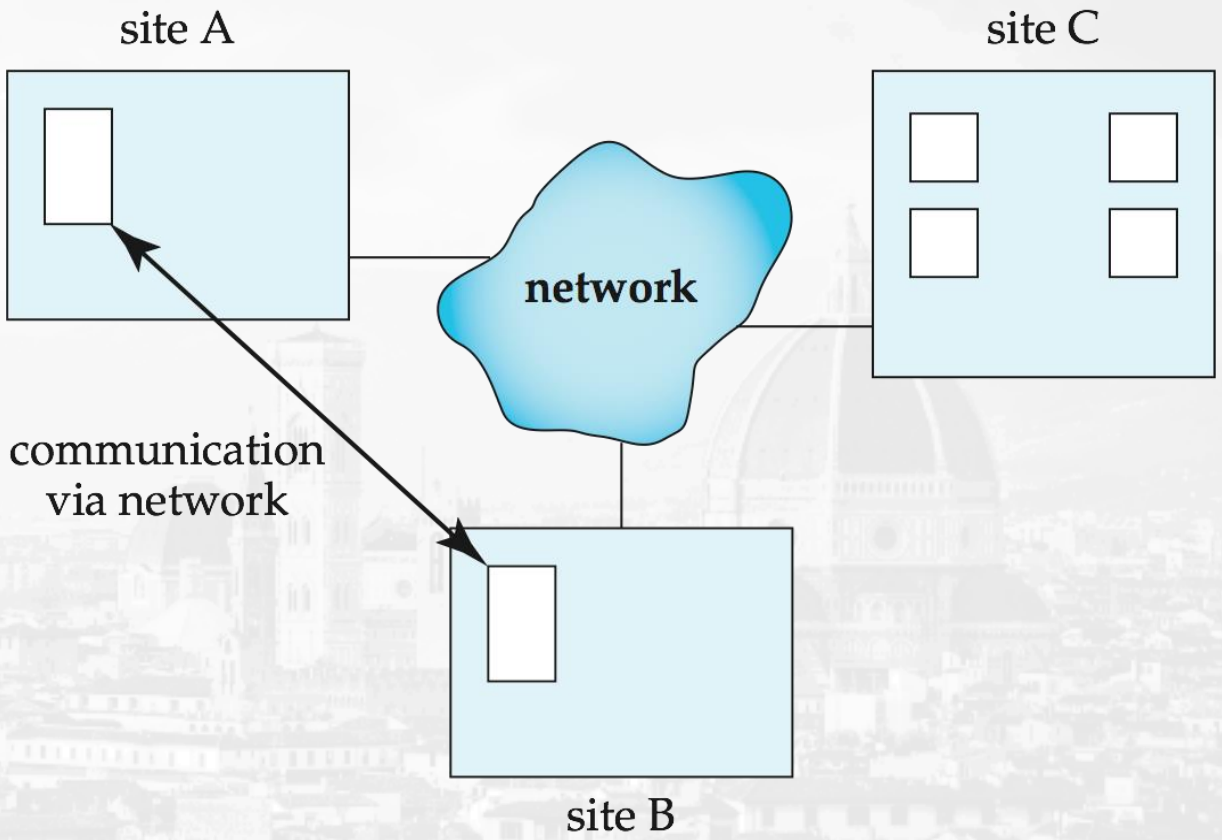
(c) shared nothing



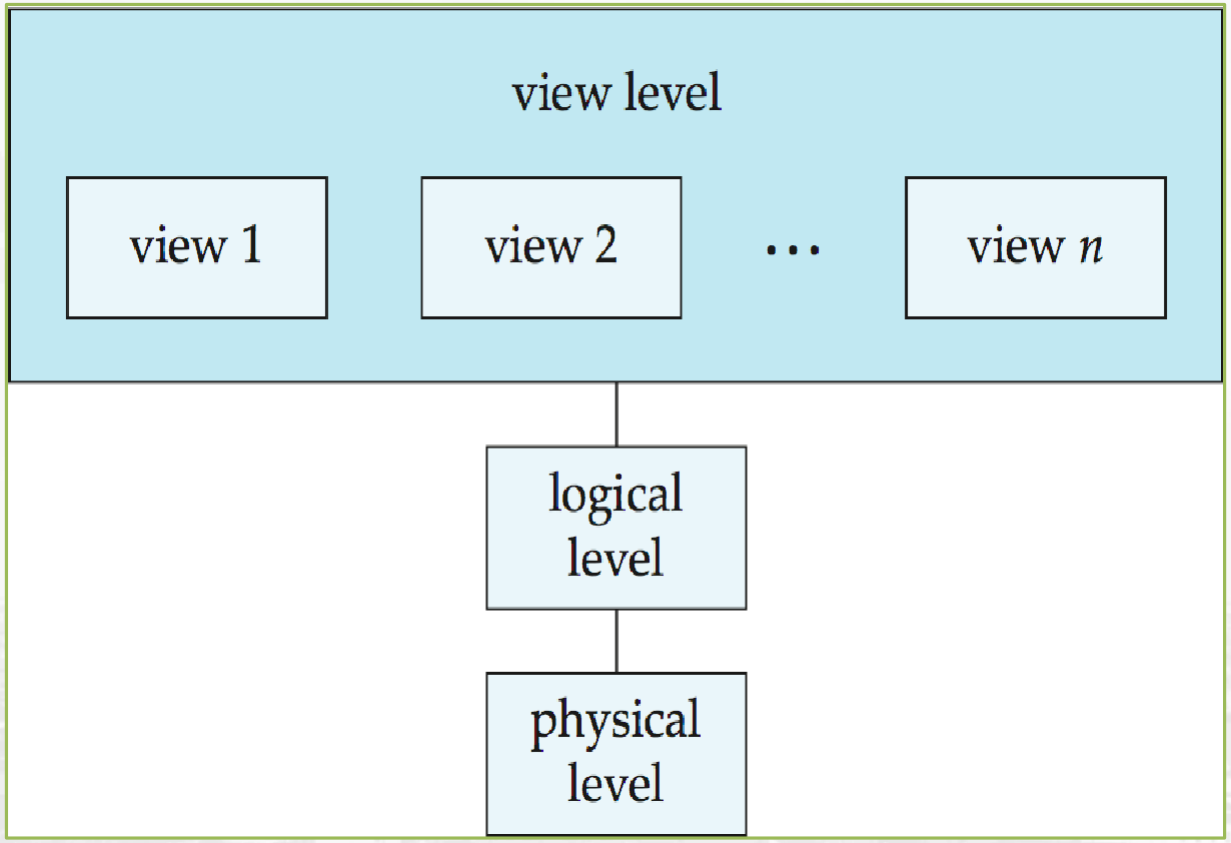
(d) hierarchical

Distributed Systems

- **Data spread over multiple machines (also referred to as sites or nodes).**
- **Network interconnects the machines**
- **Data shared by users on multiple machines**
- **Homogeneous distributed databases**
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- **Heterogeneous distributed databases**
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality
- **Differentiate between *local* and *global* transactions**
  - A **local transaction** accesses data in the *single* site at which the transaction was initiated.
  - A **global transaction** either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.



Levels of Data Abstraction in Database System



Levels of Abstraction in a Database System

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

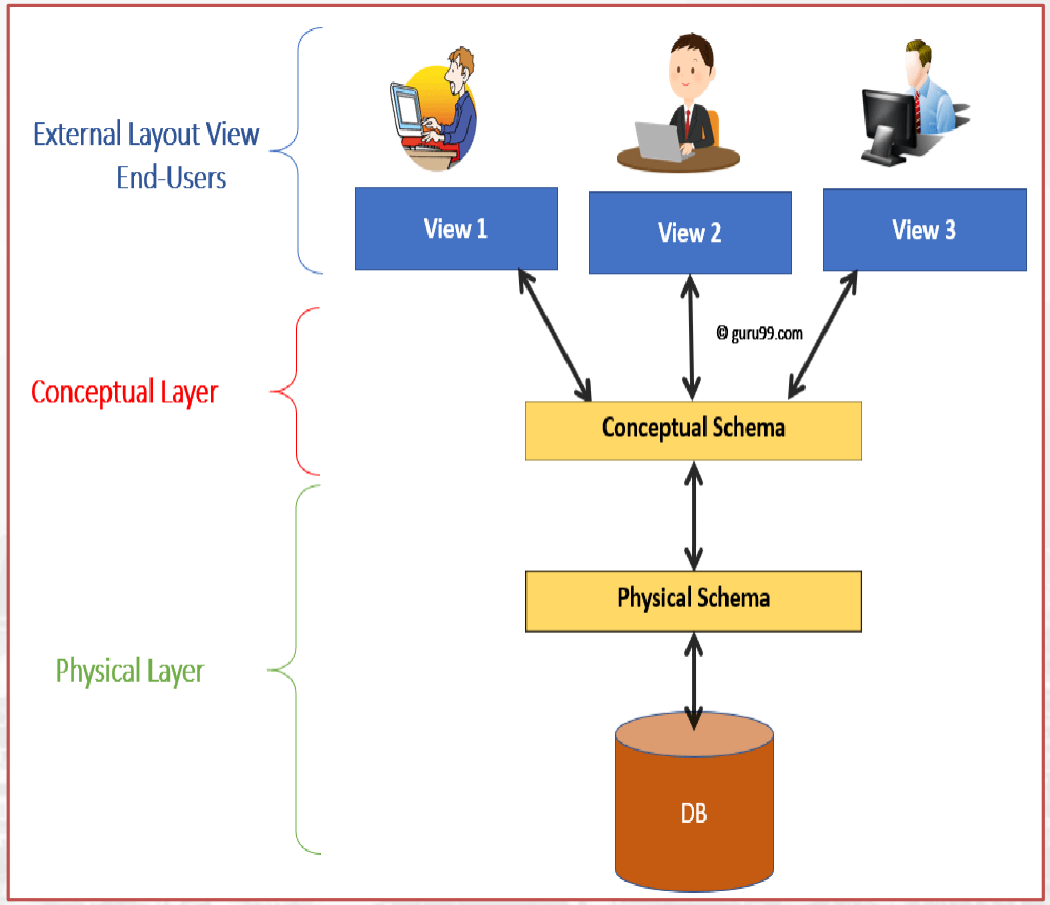
```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

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

Instances and Schema

- **Schema – the logical structure of the database**
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level

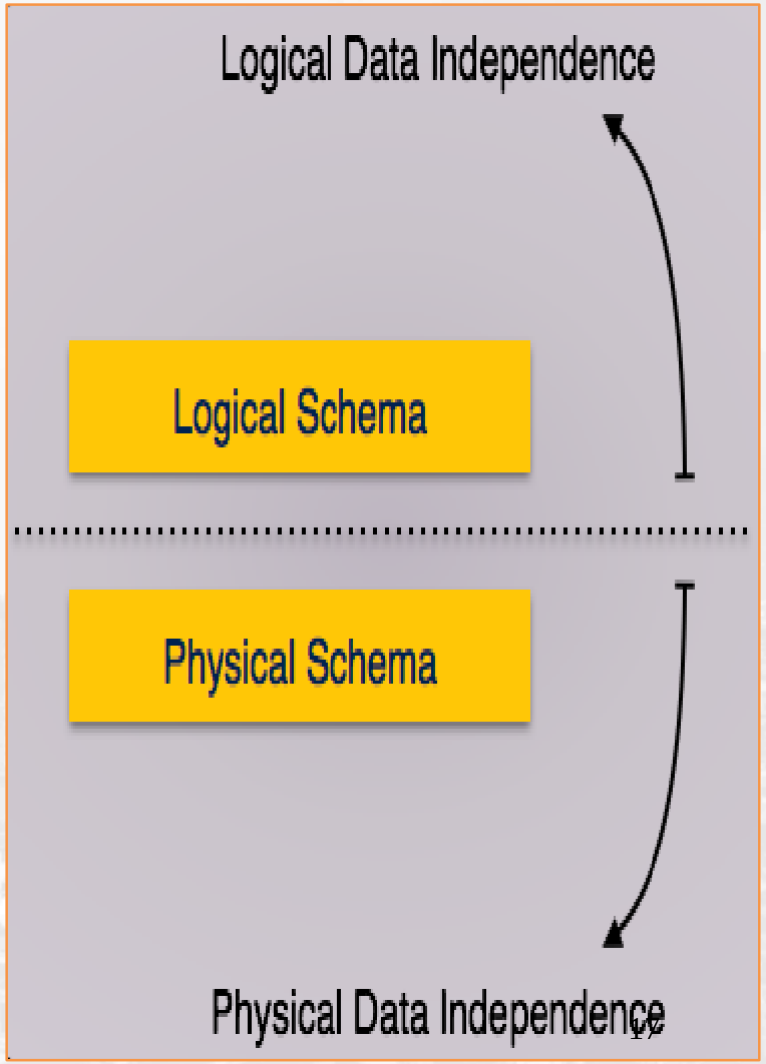
- **Instance – the actual content of the database at a particular point in time**
  - Analogous to the **value of a variable.**



Data Independence

Types of Data Independence :

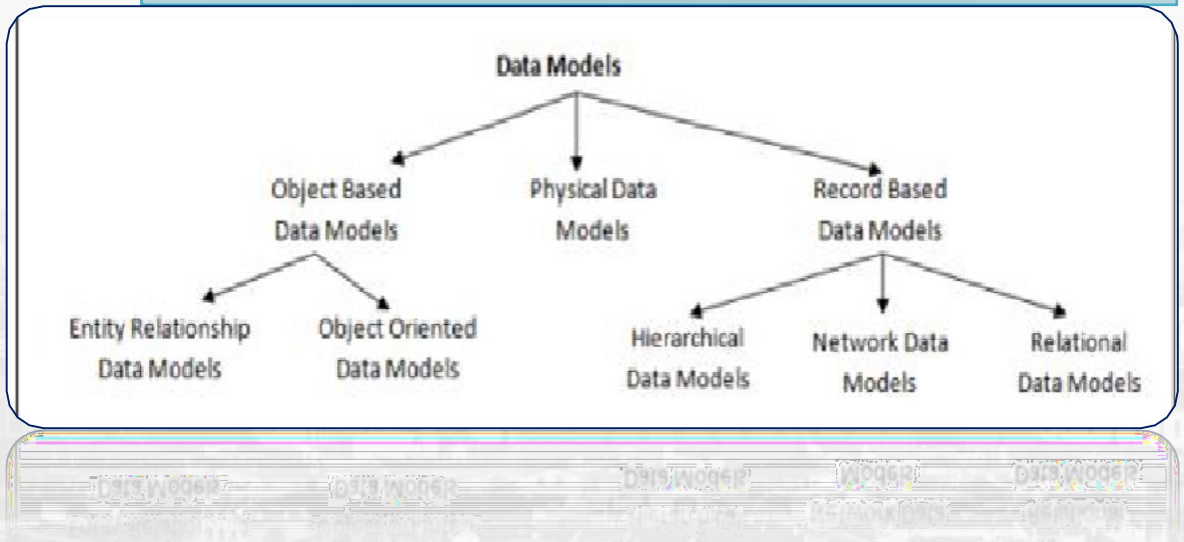
- **Physical Data Independence** : the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- **Logical Data Independence :** the ability to change the conceptual schema without changing
  - External views
  - External API or programs
  - Any change made will be absorbed by the mapping between external and conceptual levels.
  - When compared to Physical Data independence, it is challenging to achieve logical data independence.

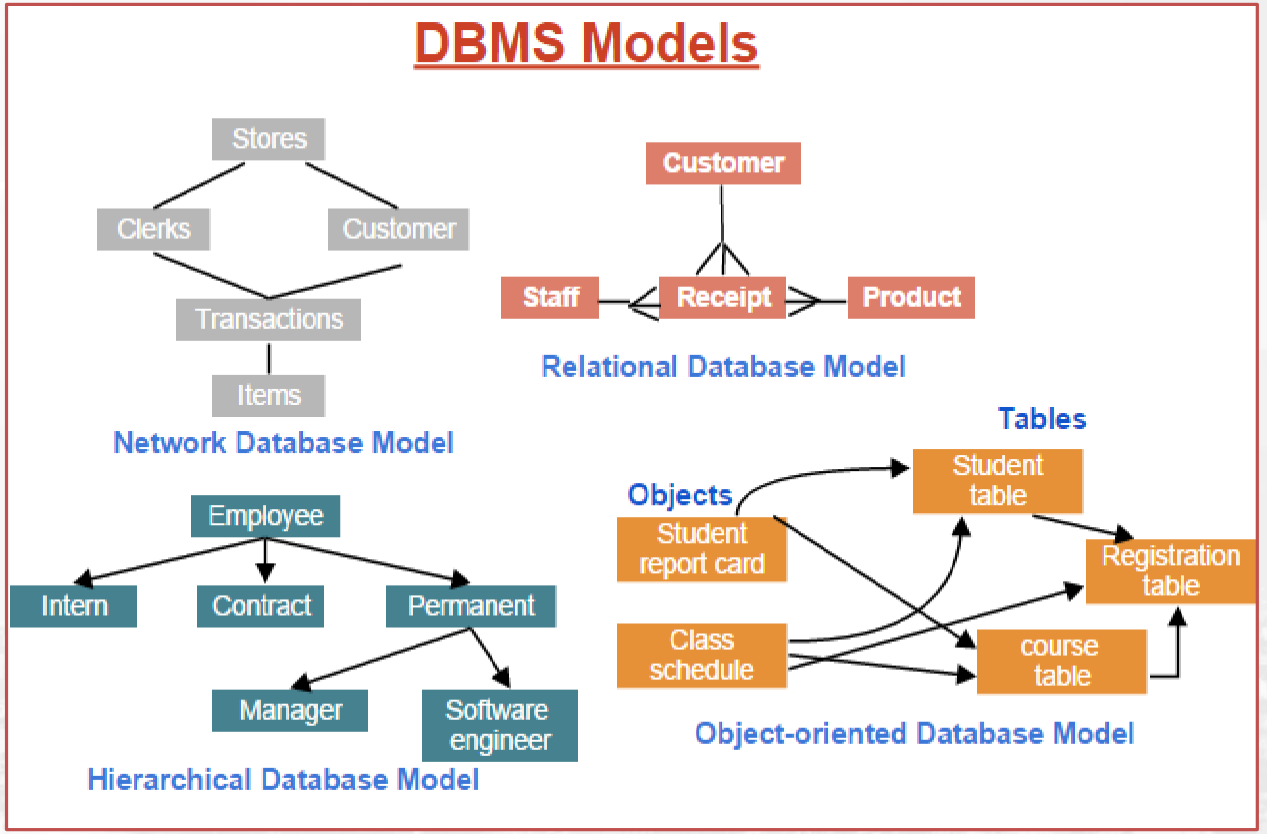


Data Models

A collection of tools for describing :

- Data
- Data relationships
- Data semantics
- Data constraints





Database System Languages

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)
  - Referential integrity (**references** constraint in SQL)
    - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation

Data Manipulation Language(DML)

Language for accessing and manipulating the data organized by the appropriate data model

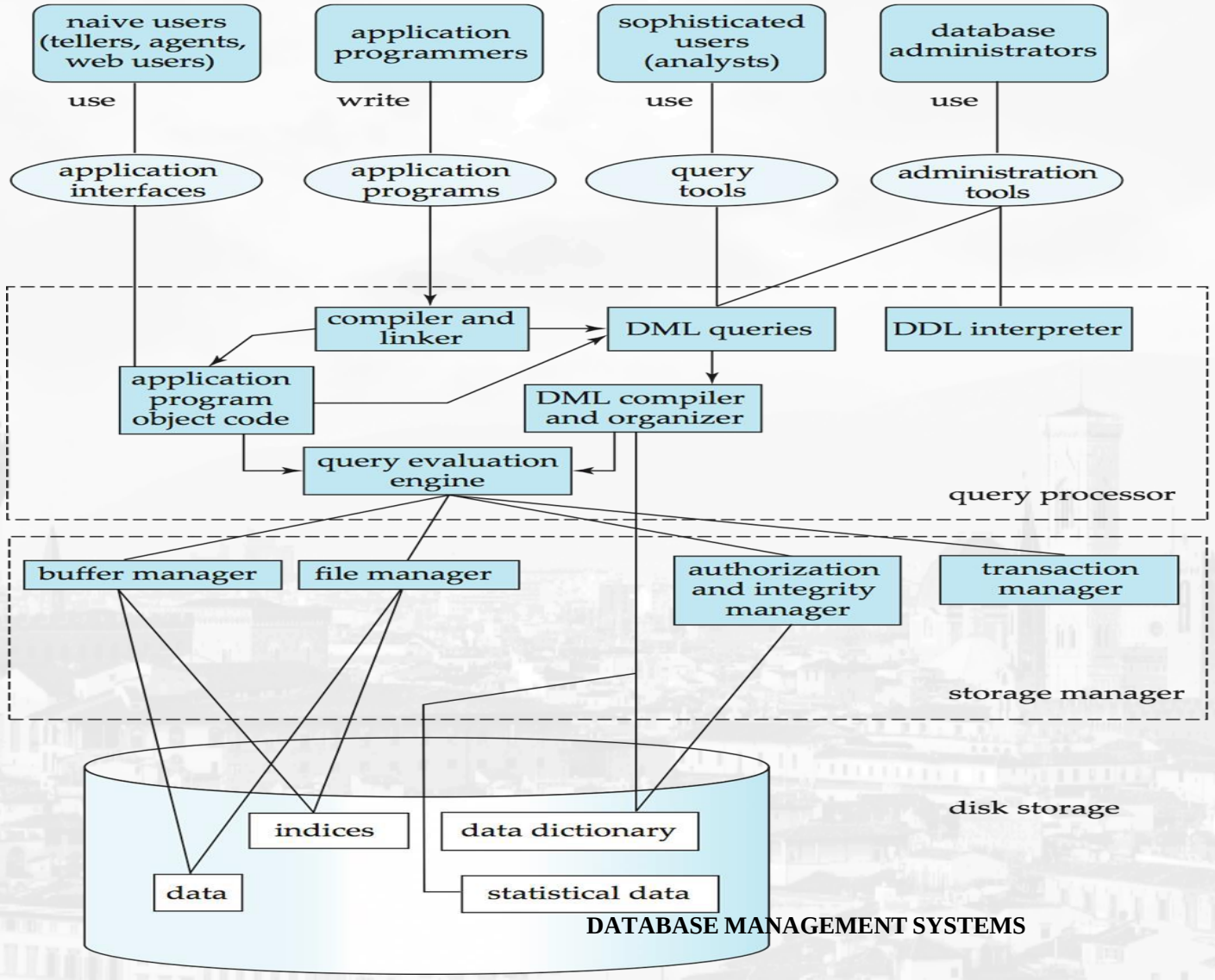
- DML also known as query language

Two classes of languages

- **Procedural** – user specifies what data is required and how to get those data
- **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data

SQL is the most widely used query language

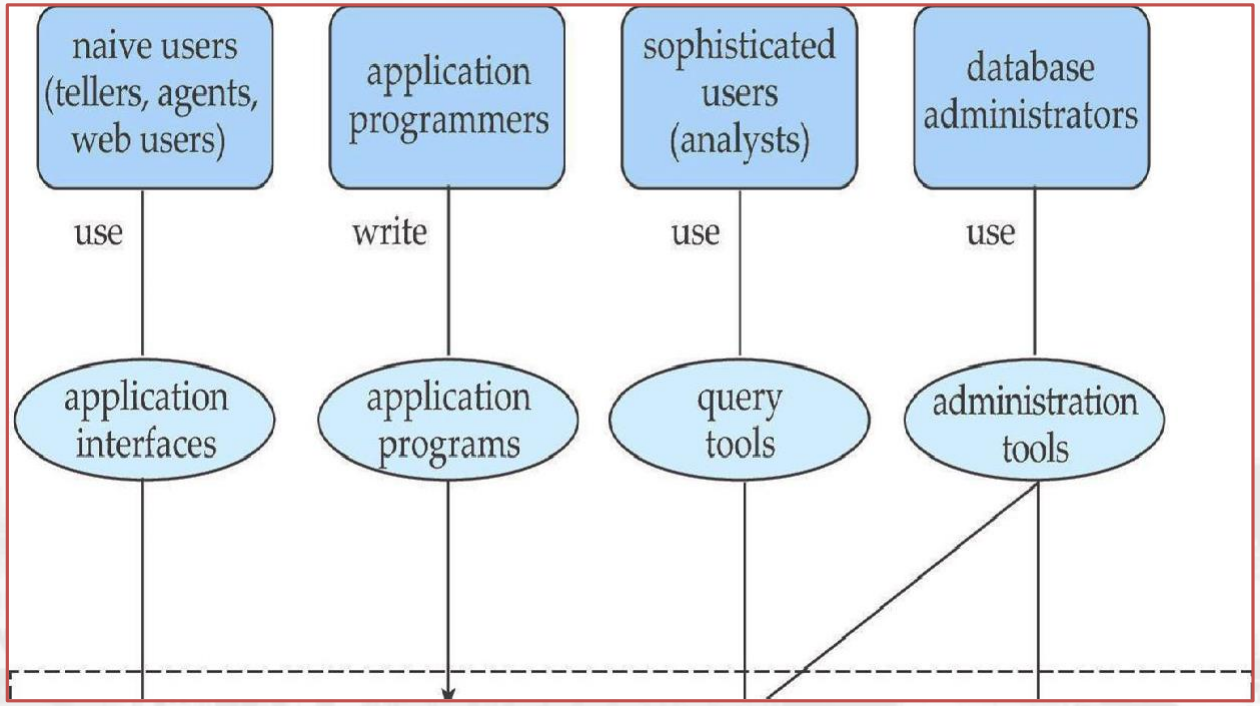
Database System Structure/Architecture



Important Components of Database System :

- Database Users
- Query Processing
- Storage Management
- Transaction Management

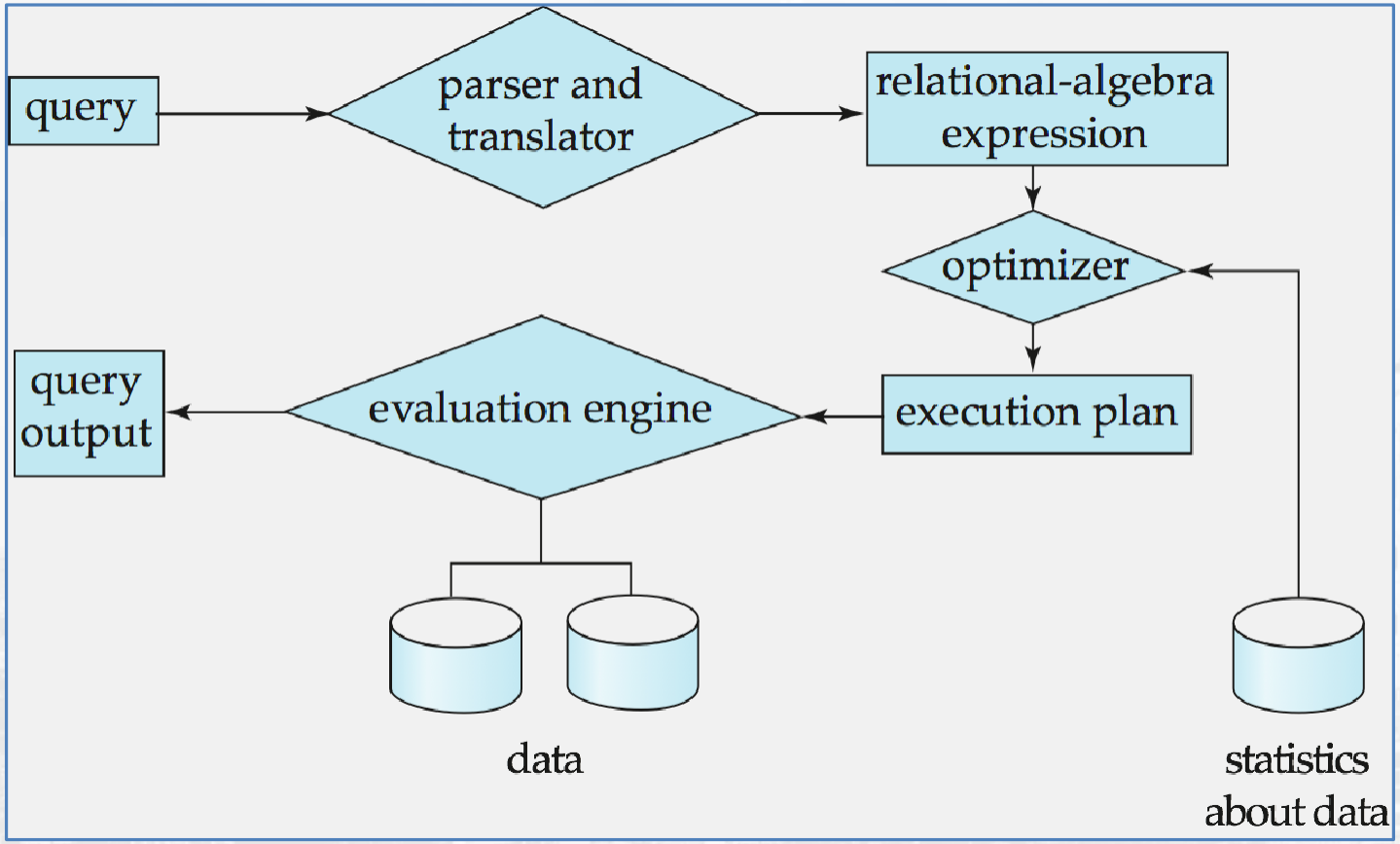
Database System Components : Database Users



Types of Database Users :

- Naive Users
- Application Programmers
- Sophisticated Users
- Database Administrators

Database System Components : Query Processing



Query Processing Steps :

1.Parsing and Translation

2.Optimization

3.Evaluation

Database System Components :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.

It is responsible for the following tasks:

- Interaction with the file manager
- Efficient storing, retrieving and updating of data

Issues:

- Storage access
- File organization
- Indexing and hashing

Database System Components :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.

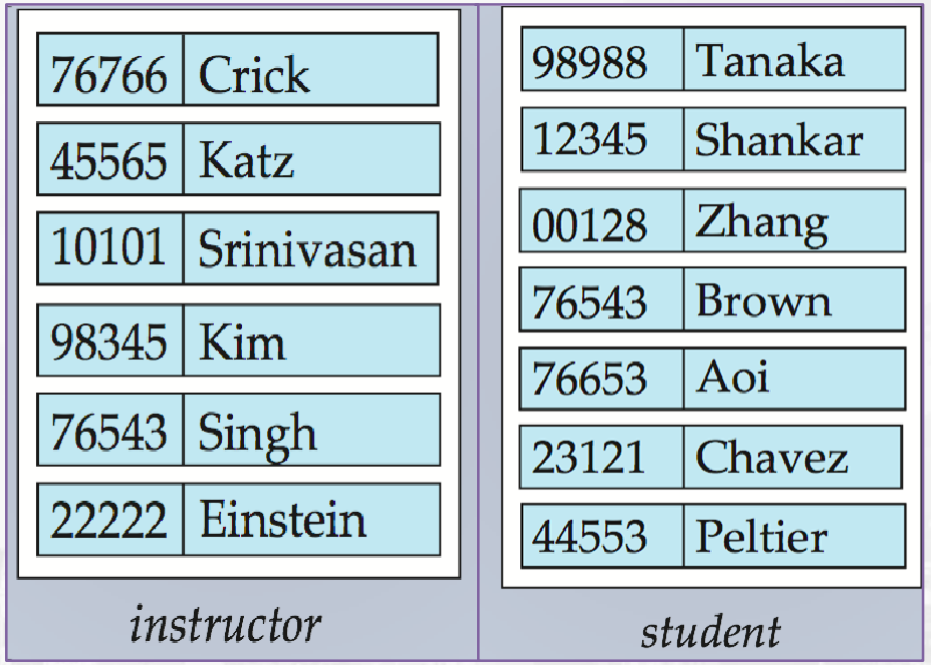
Two Important Components related to Transactions:

- **Transaction Manager**
- **Concurrency Control Manager**

- **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.

Entity Relationship Model

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have **attributes**
  - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays



Relationship Sets

A **relationship** is an association among several entities

Example:

44553 (Peltier) *advisor* 22222 (Einstein)

student* entity** **relationship set** ***instructor **entity**

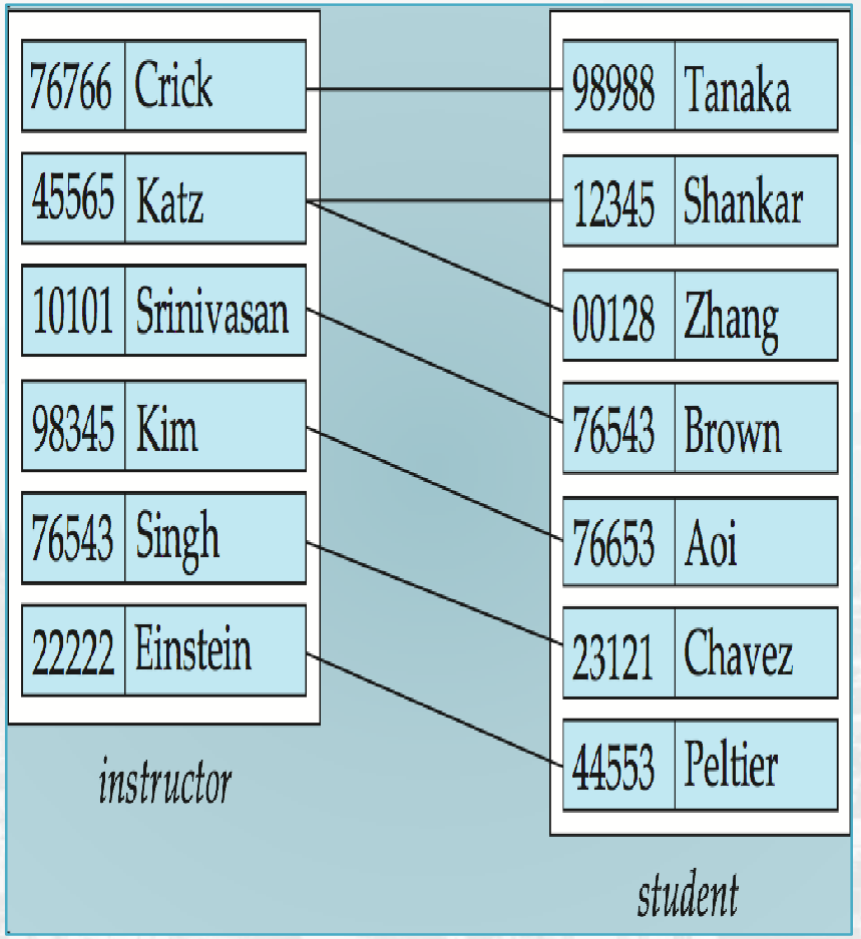
A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

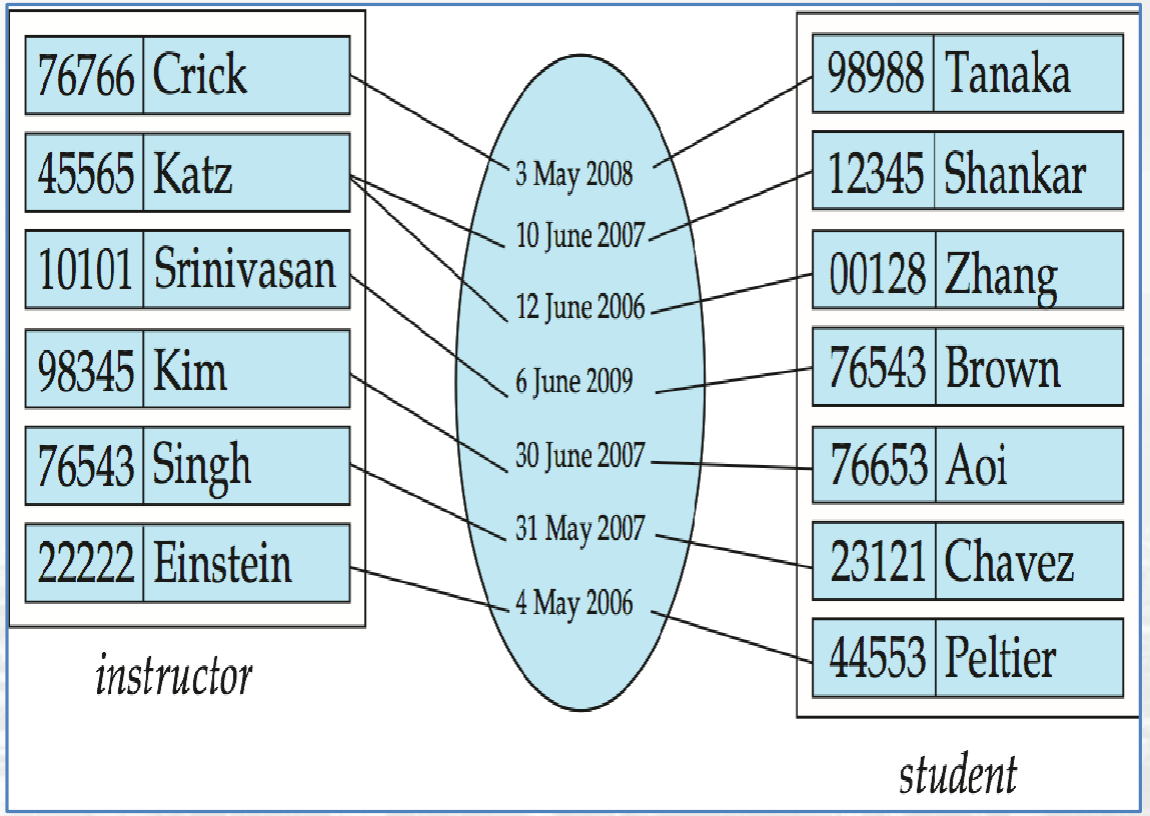
- Example:

$(44553,22222) \in advisor$



Relationship Sets

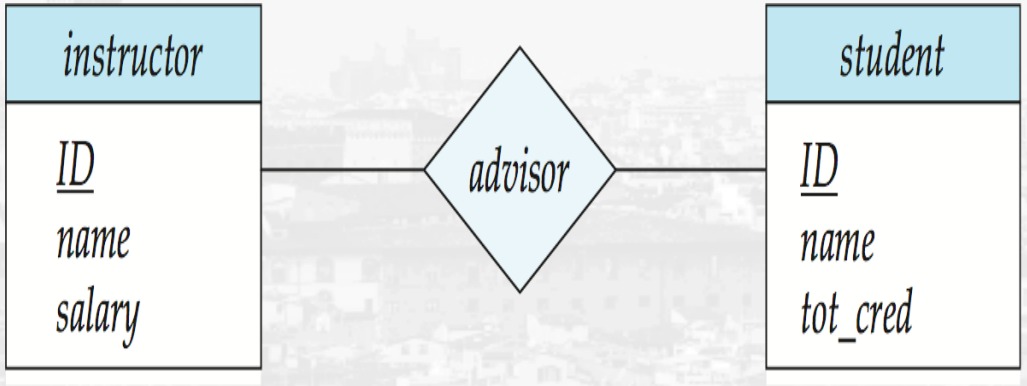
- An **attribute** can also be property of a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



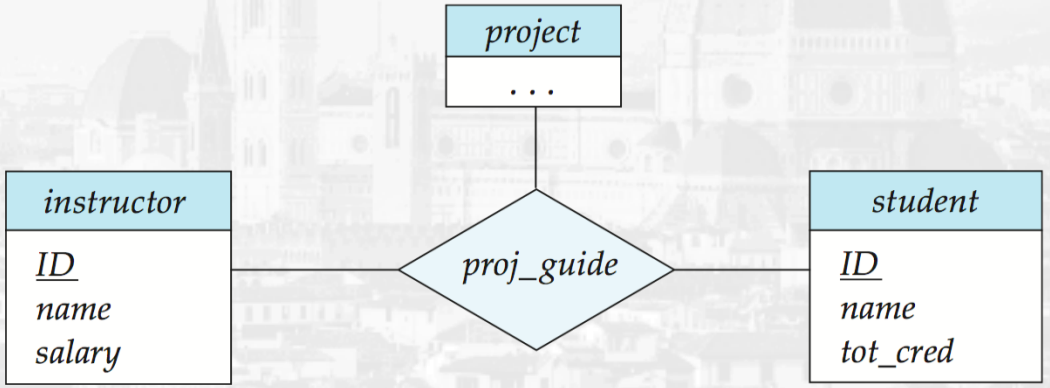
Degree of a Relationship Set

- **Binary relationship**
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.

Relationships between more than two entity sets are rare. Most relationships are binary.

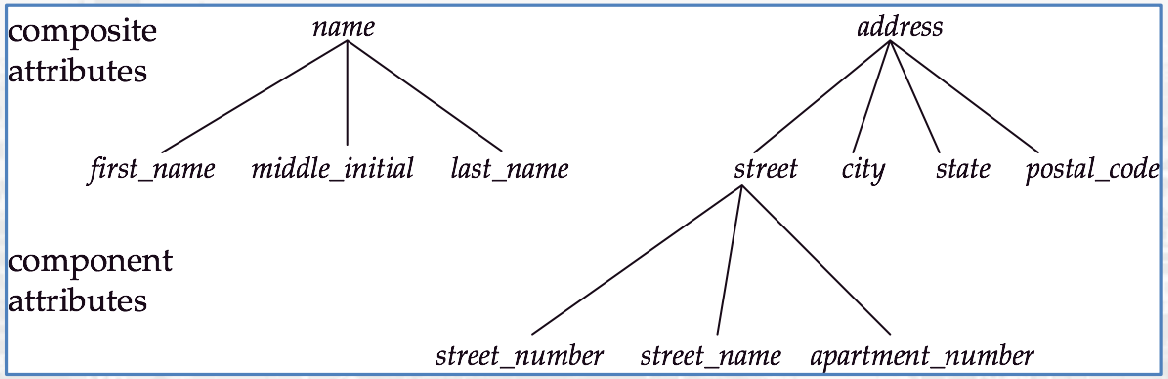


- **Ternary relationship**
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*



Attributes

- **Attribute types**
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes.
    Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    Can be computed from other attributes
    Example: age, given date_of_birth



- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (*ID*, *name*, *street*, *city*, *salary*)

    course= (*course_id*, *title*, *credits*)

- **Domain** – the set of permitted values for each attribute

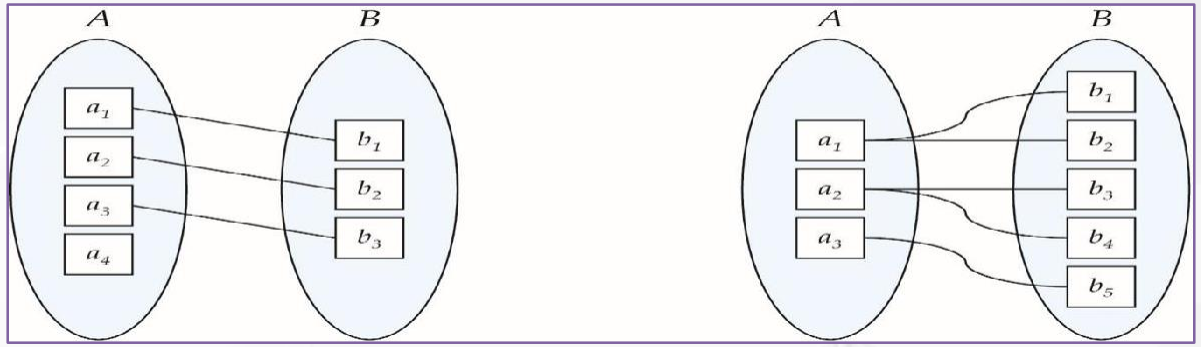
Mapping Cardinality Constraints

Express the number of entities to which another entity can be associated via a relationship set.

Most useful in describing binary relationship sets.

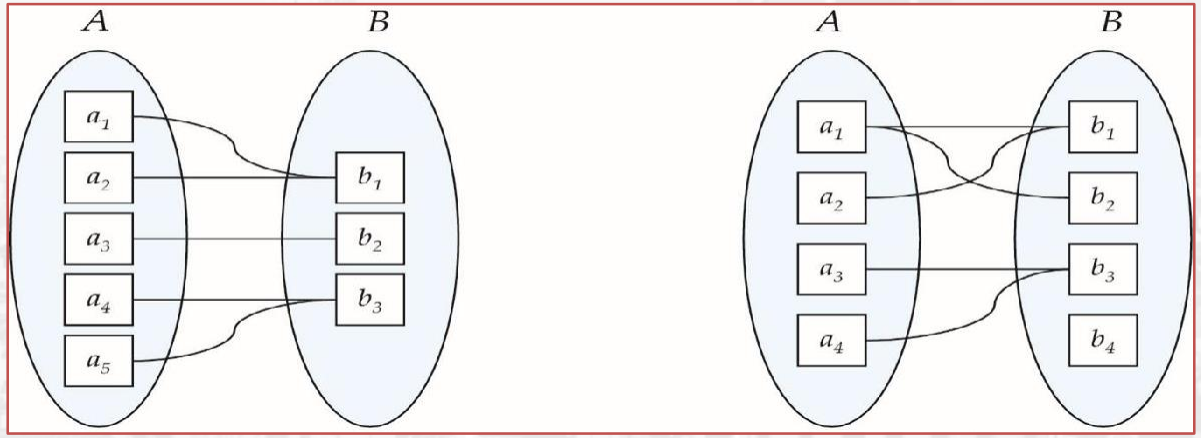
For a binary relationship set the mapping cardinality must be one of the following types:

- ***One to one***
- ***One to many***
- ***Many to one***
- ***Many to many***



One to one

One to many



Many to one

Many to many

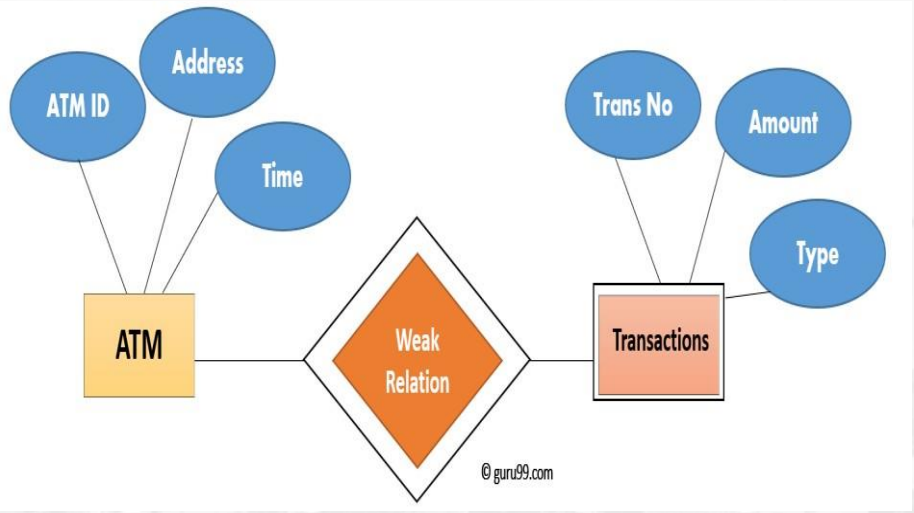
Keys for Relationship Sets

The combination of primary keys of the participating entity sets forms a super key of a relationship set.

- **(*s_id, i_id*) is the super key of *advisor***
- *NOTE:* *this means* ***a pair of entity sets can have at most one relationship in a particular relationship set.***
  - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. We can use a multivalued attribute though

Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys

Entity Relationship Dia

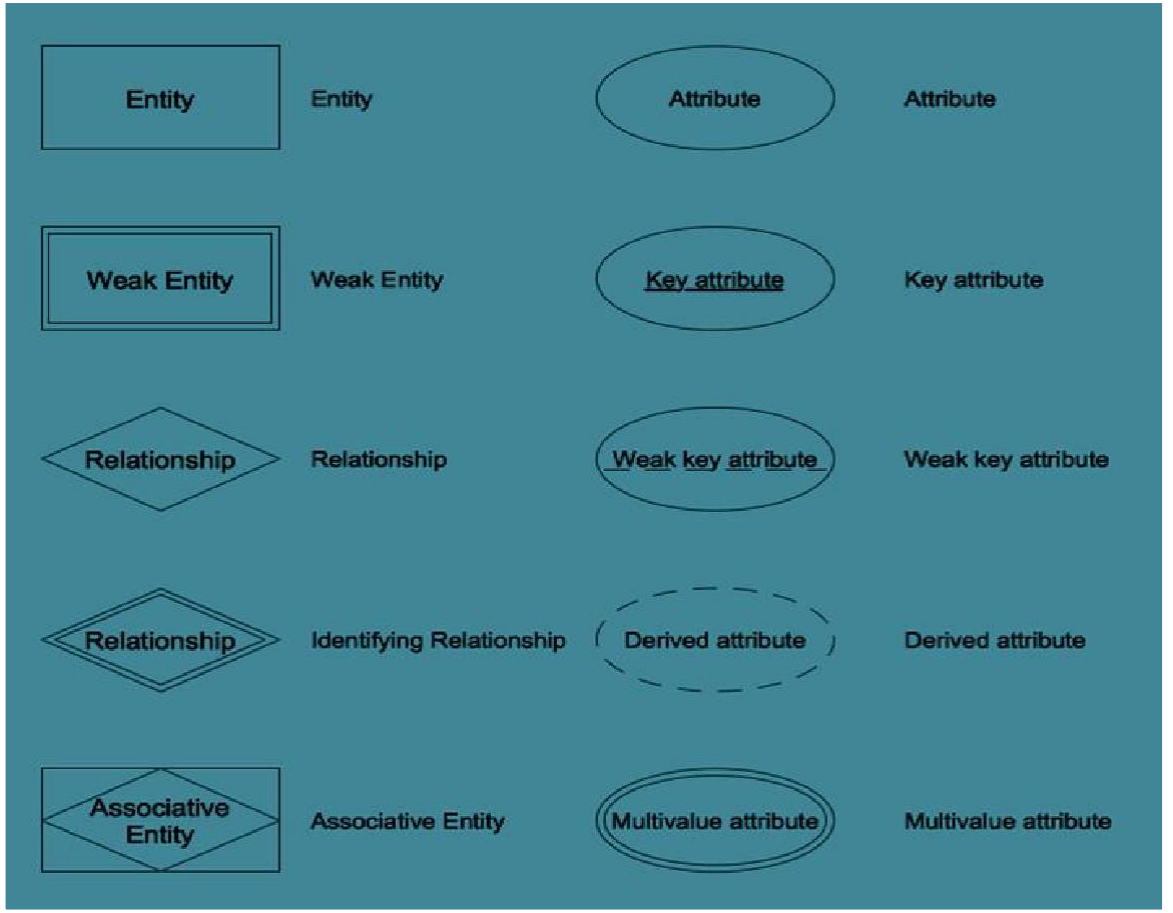


Entity-Relationship Diagram

It is a graphical Representation of ER Model

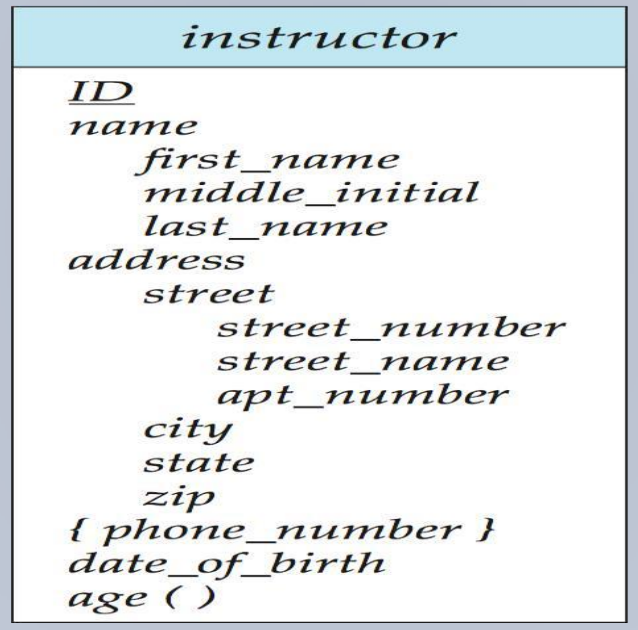
Basic Notations :

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

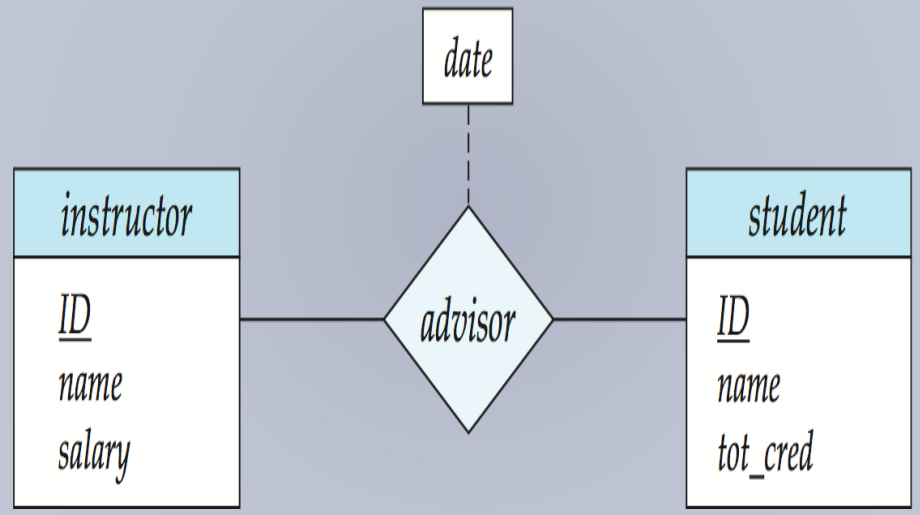


Entity Relationship Diagram Continued

- **Entity With Composite, Multivalued, and Derived Attributes**



- **Relationship Sets with Attributes**

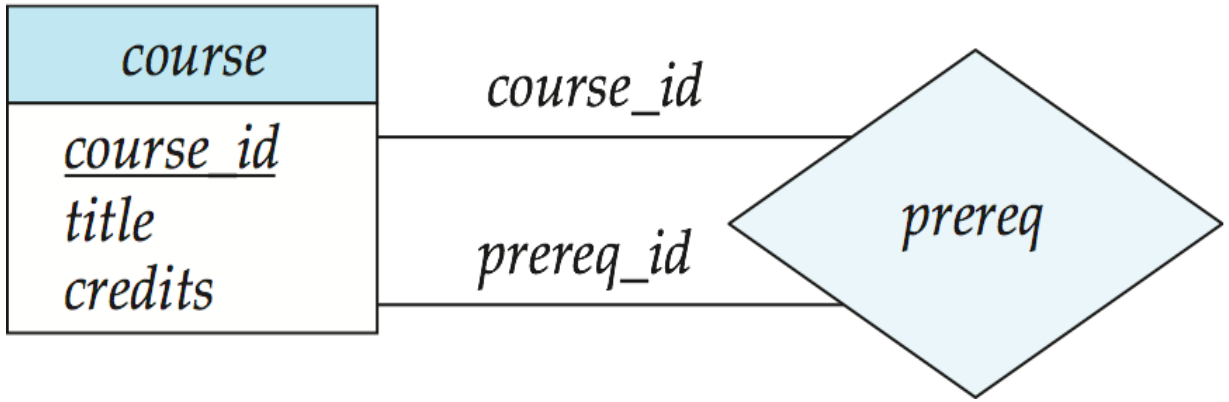


Entity Relationship Diagram Continued

Roles

- Entity sets of a relationship need not be distinct.
- Each occurrence of an entity set plays a "role" in the relationship.

The labels "*course_id*" and "*prereq_id*" are called **roles**.



Entity Relationship Diagram Continued

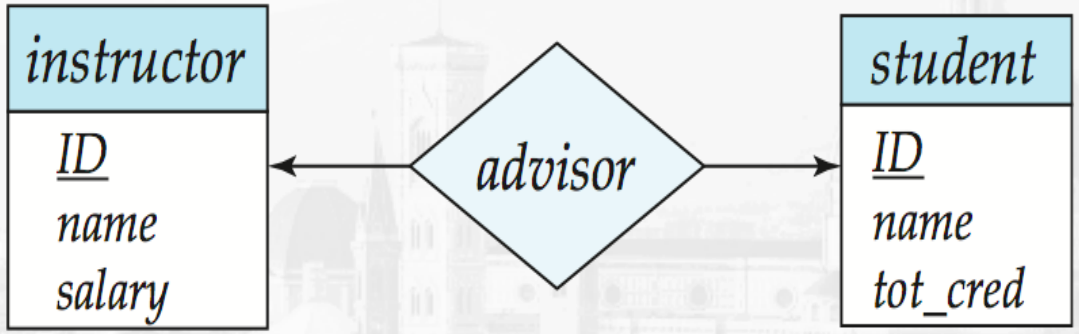
Cardinality Constraints

We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

1. One-to-One Relationship

one-to-one relationship between an *instructor* and a *student*

- an instructor is associated with at most one student via *advisor*
- and a student is associated with at most one instructor via *advisor*

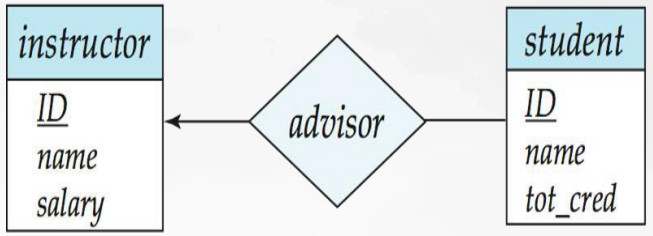


Entity Relationship Diagram Continued

b. One-to-Many Relationship

A one-to-many relationship between an *instructor* and a *student*

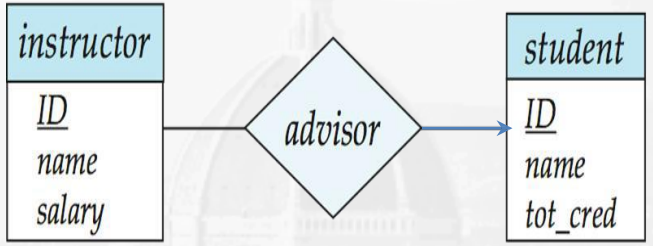
- a an instructor is associated with several (including 0) students via *advisor*
- student is associated with at most one instructor via advisor



c. Many-to-One Relationship

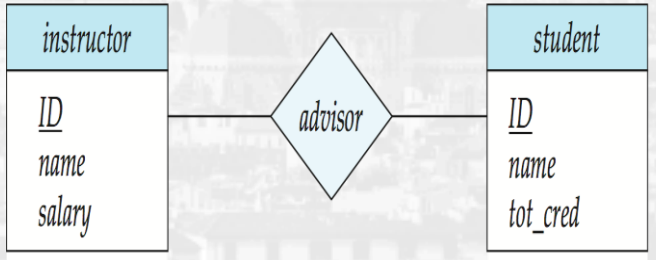
In a many-to-one relationship between an *instructor* and a *student*,

- an instructor is associated with at most one student via *advisor*,
- and a student is associated with several (including 0) instructors via *advisor*



d. Many-to Many Relationship

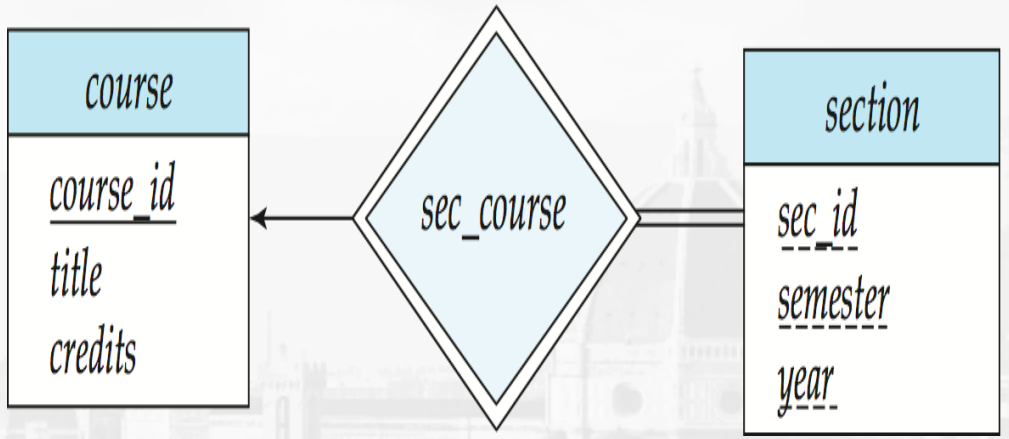
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



Entity Relationship Diagram Continued

Participation of an Entity Set in a Relationship Set

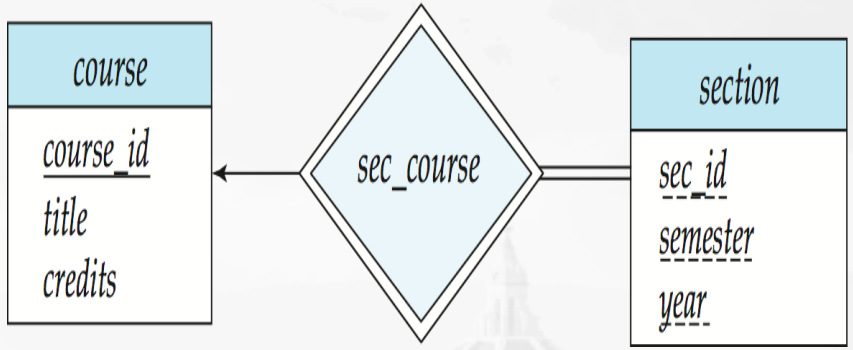
- **Total participation** (**indicated by double line**): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *section* in *sec_course* is total
    - every *section* must have an associated course
- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial



Entity Relationship Diagram : Weak Entity Sets

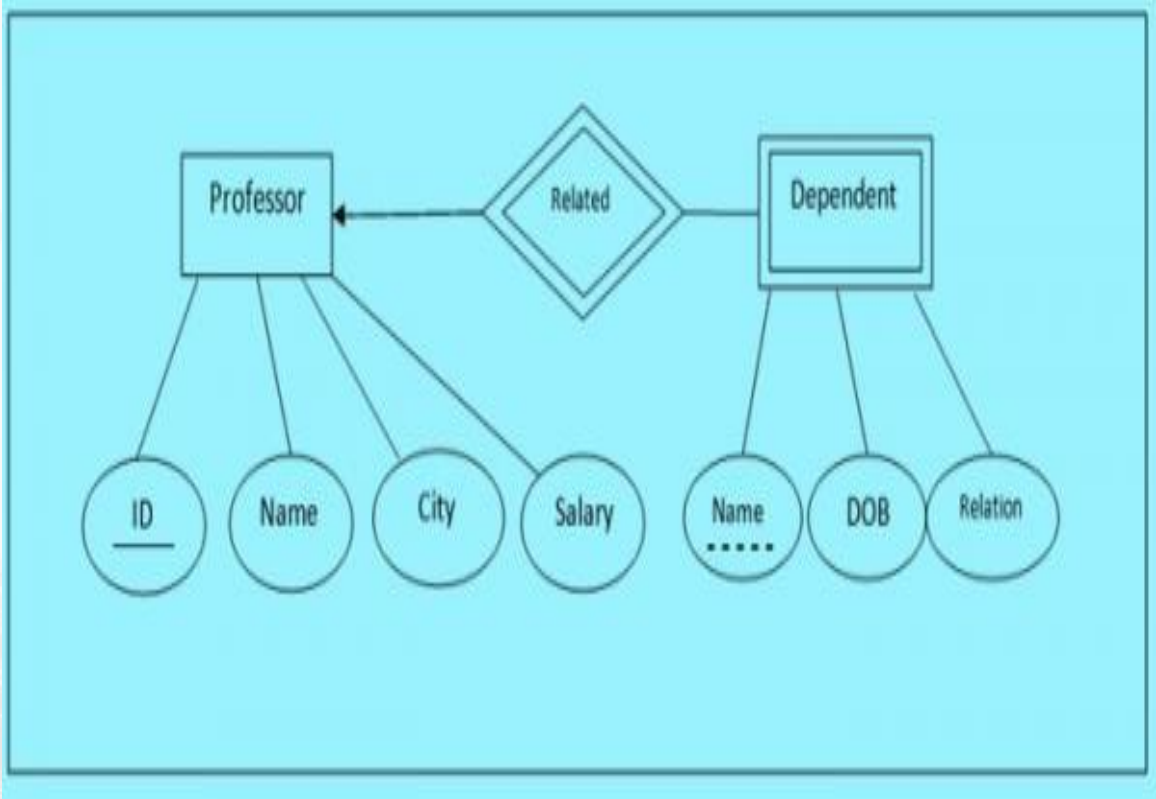
- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - **Identifying relationship depicted using a double diamond**
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.

The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.



- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – (*course_id*, *sec_id*, *semester*, *year*)

Example of Strong and Weak Entity



Strong Entity :
Professor(ID,Name,City,Salary)

Weak Entity :
Dependent(Name,DOB,Relation)

The Dependent Entity will share the ID attribute of Professor.

Resultant Schema :

Dependent(ID,Name,DOB,Relation)

The primary key for Weak Entity Dependent will be ID + Name as Name is the discriminator attribute.

E-R Diagram Example

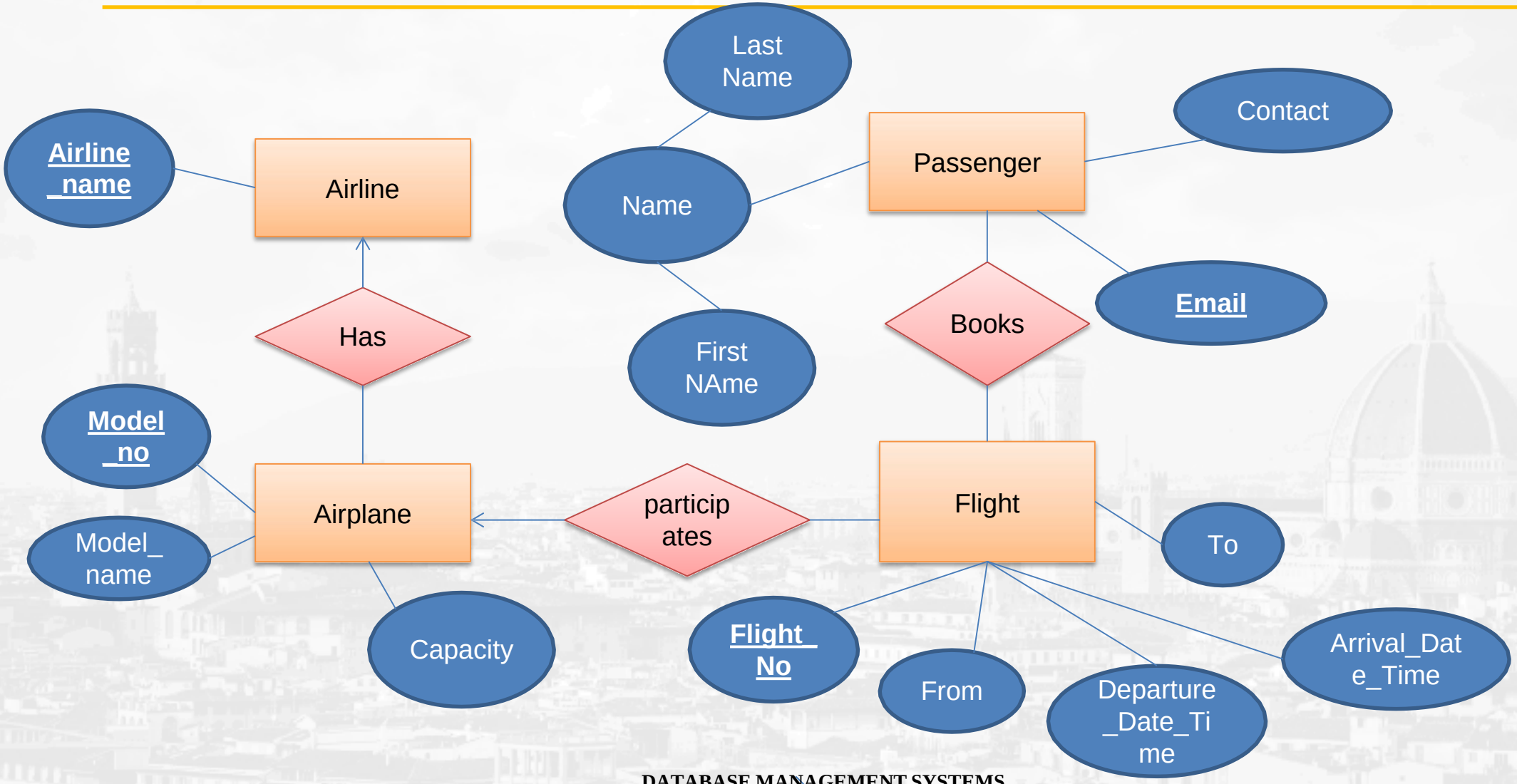
Question : **Design an ER Diagram for Airline Reservation scenario given below :**

The flight database stores details about an airline's fleet, flights, and seat bookings.

Consider the following scenario:

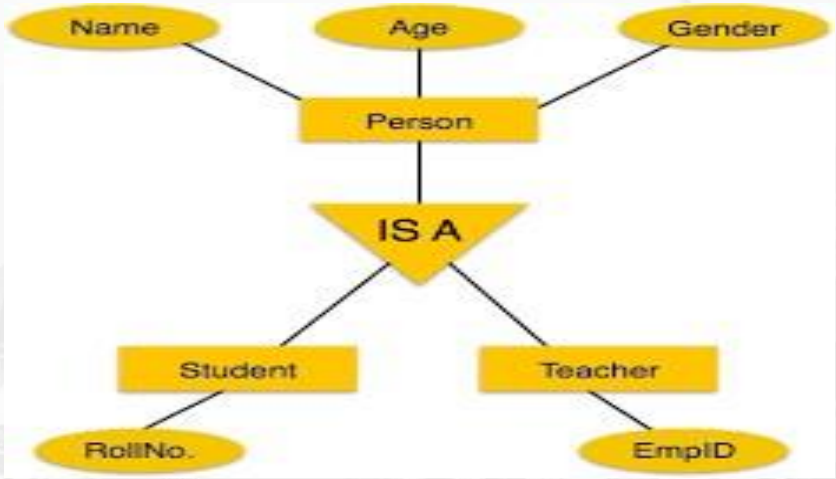
- The airline has one or more airplanes.
- An airplane has a model number, a registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- Each flight is carried out by a single airplane.
- A passenger has given names(first name, last name),contact and a unique email address.
- Passengers can book seats on flights.

Flight Reservation ERD



Extended ER Features :

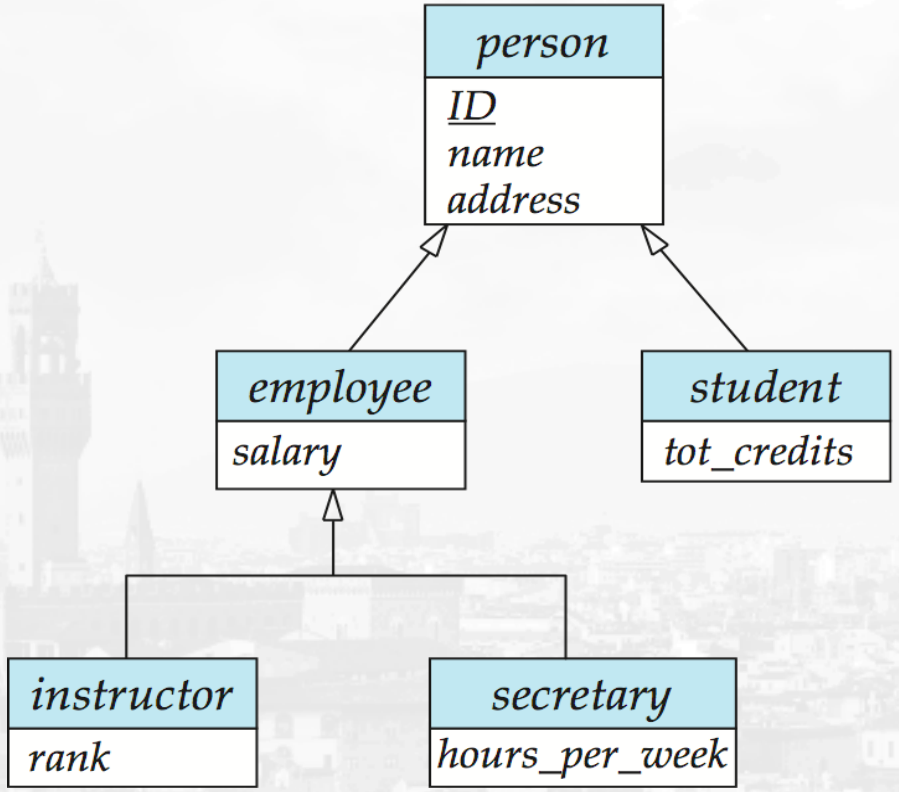
- Specialization
- Generalization
- Aggregation

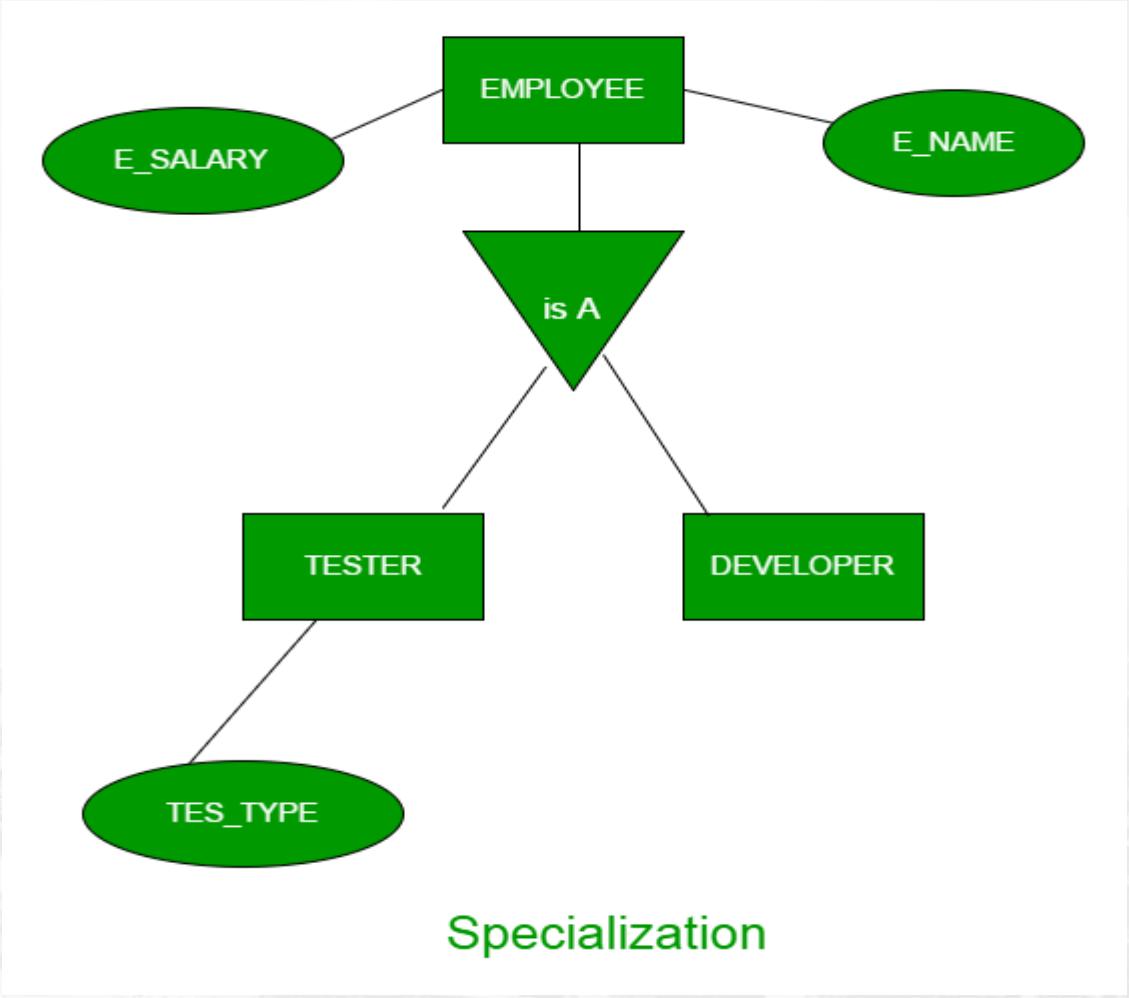


Extended ER Features : Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled IS A (E.g., *instructor* “is a” *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization : Example

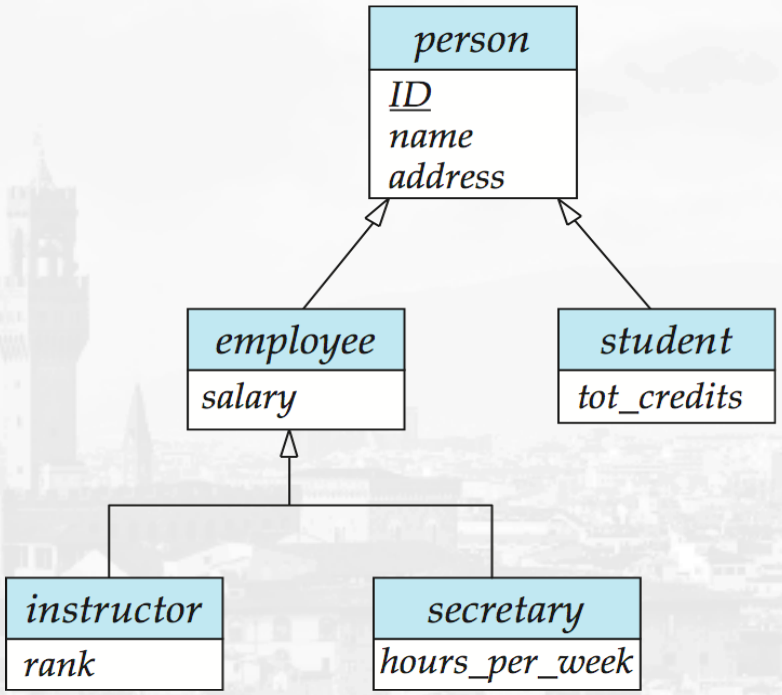


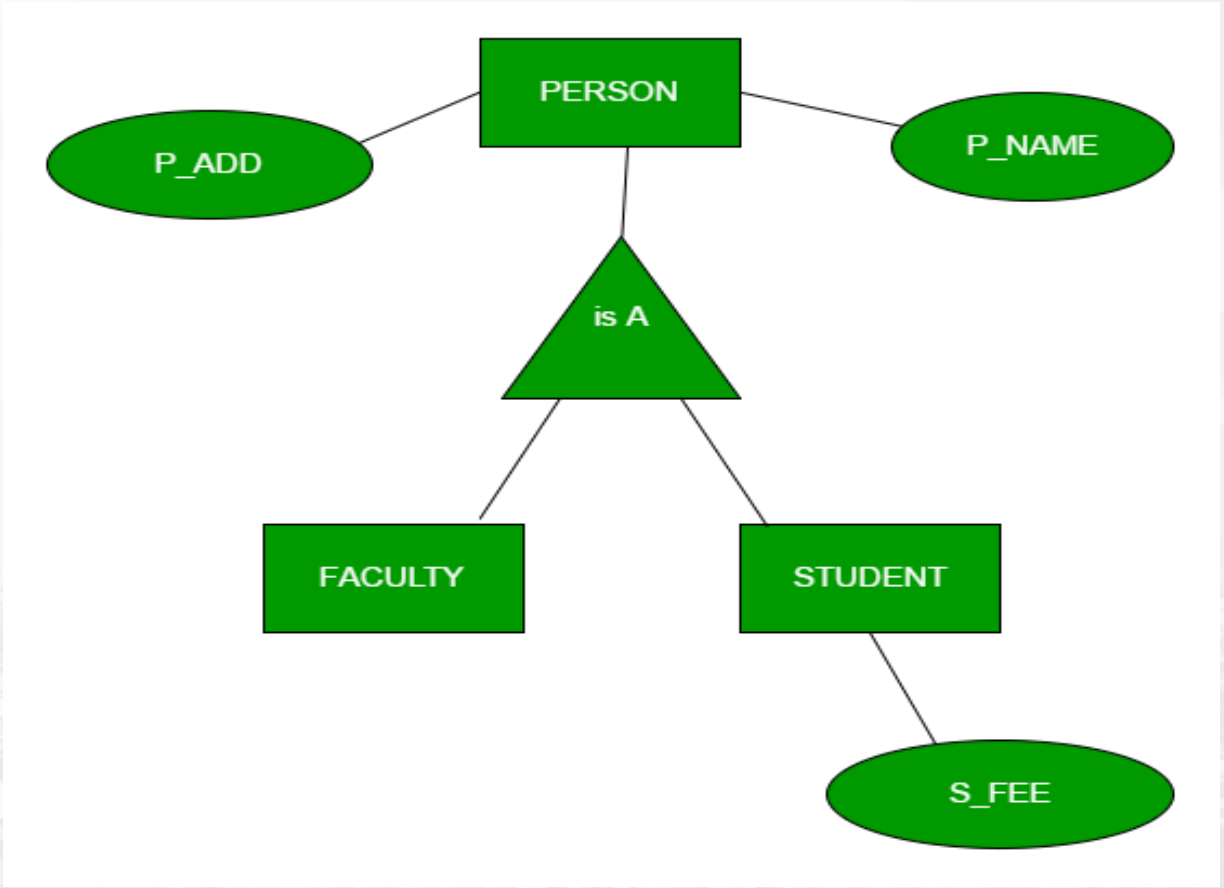


Extended ER Features : Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

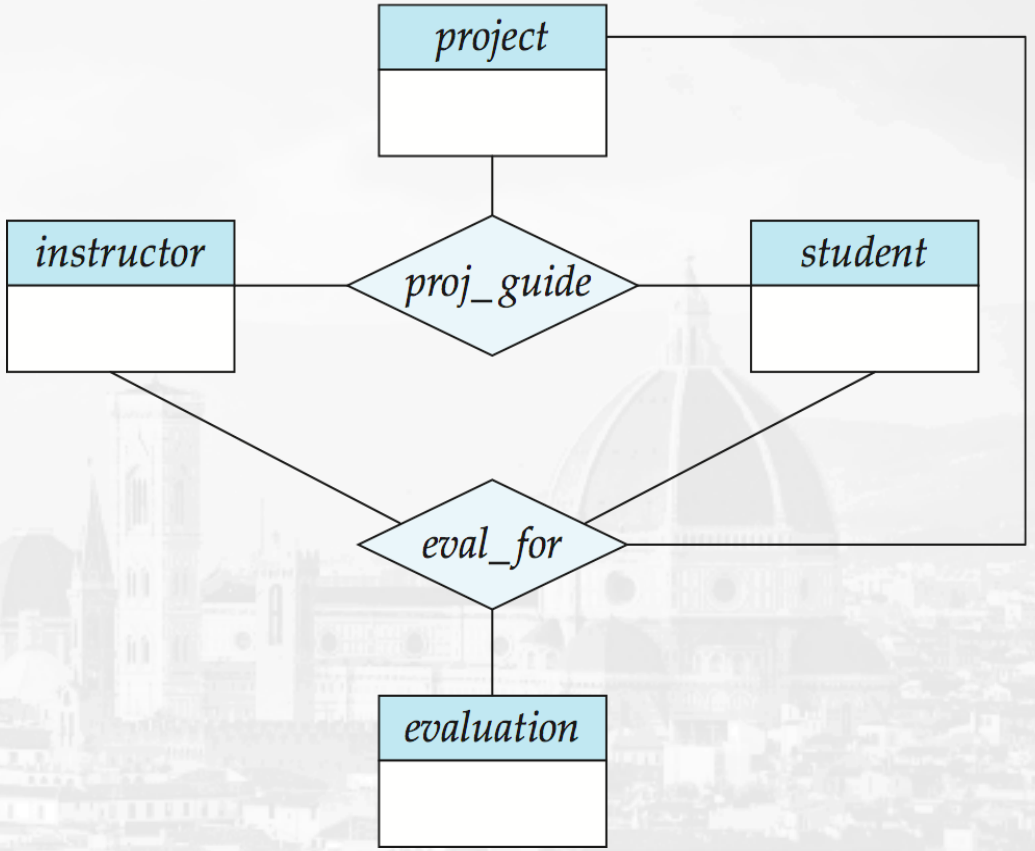
Generalization : Example





Extended ER Features : Aggregation

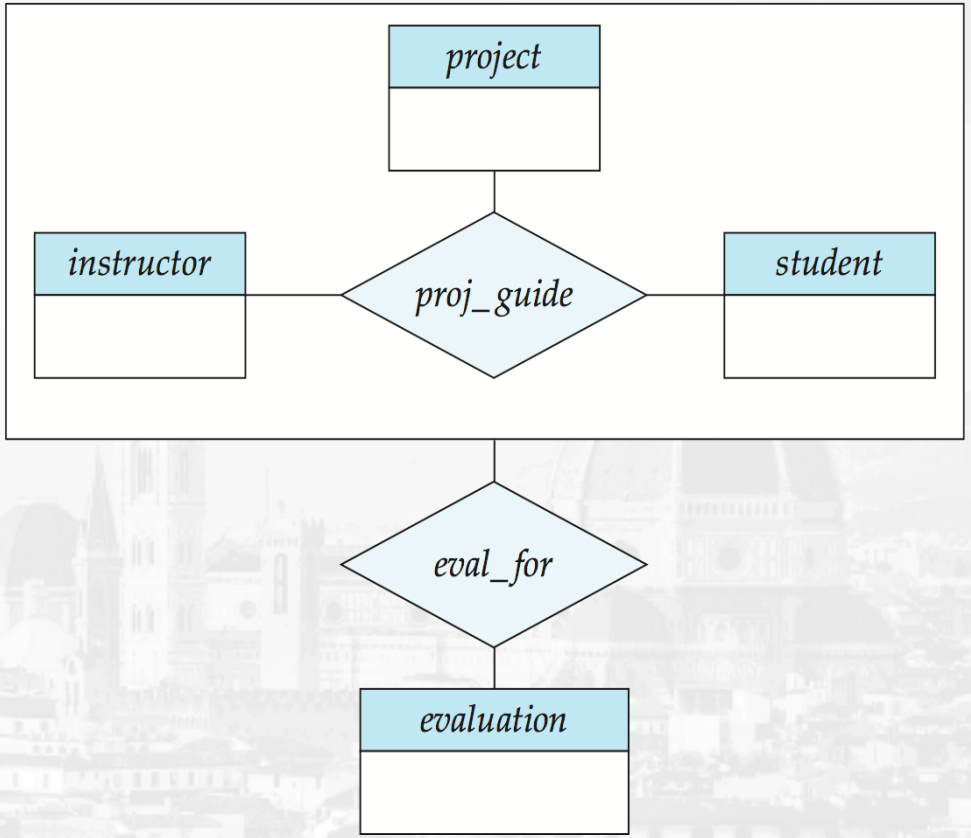
- Consider the ternary relationship *proj_guide*, which we saw earlier
- Suppose we want to record evaluations of a student by a guide on a project.
- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So we can't discard the *proj_guide* relationship



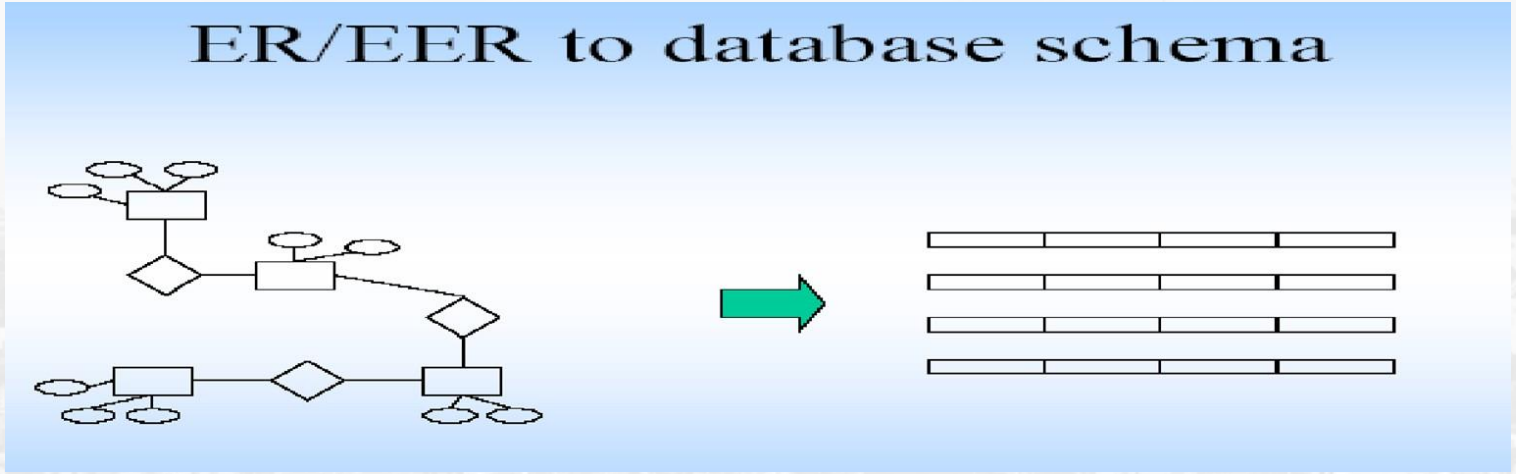
Extended ER Features : Aggregation

- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

- Without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



Reduction of ER Diagram to Relation Schemas

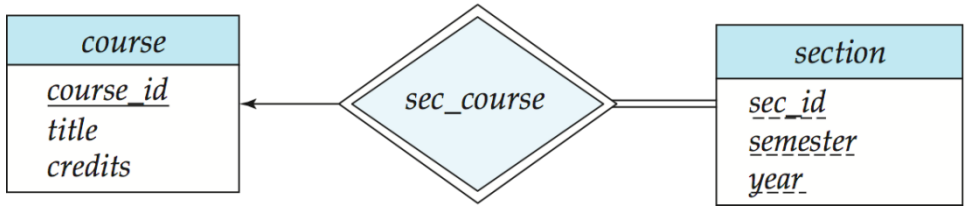


Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names

Reduction to Relation Schemas

Representing Entity Sets with Simple Attributes



A strong entity set reduces to a schema with the same attributes

course(course_id,title,credits)

course_id	title	credits

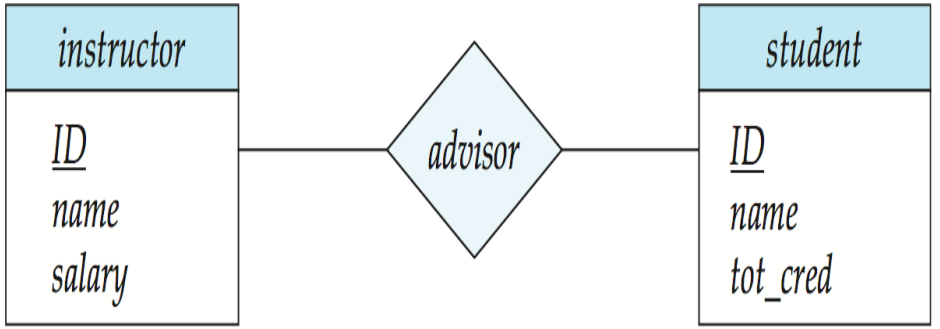
A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

section (cours _id, id, sem,)

Course_id	Sec_id	sem	year

e

Representing Relationship Sets :



A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

Example: schema for relationship set *advisor*

advisor = (*s_id, i_id*)

S_id	i_id

Reduction to Relation Schema

Redundancy of schemas:

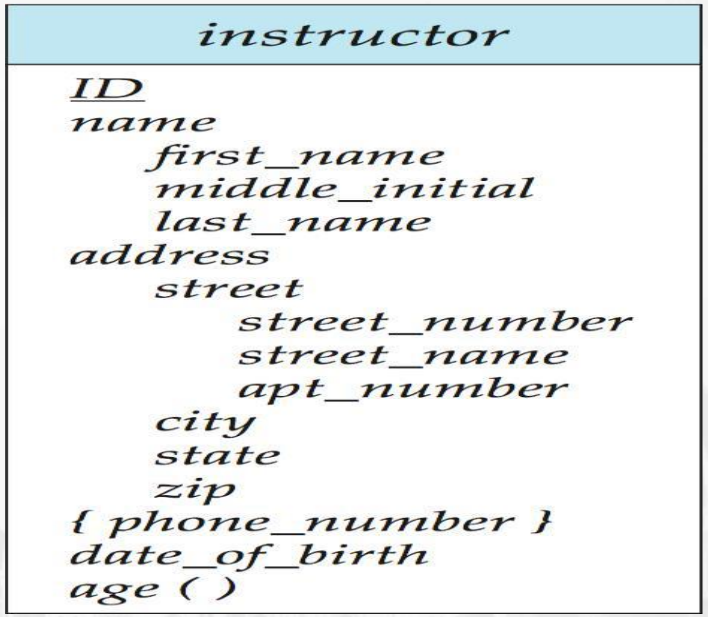
- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side.

Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*

Reduction to Relation Schema

Representing Composite and Multi-valued Attributes

- **Composite attributes** are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
    - *Prefix omitted if there is no ambiguity*



- Ignoring multivalued attributes, extended instructor schema is :

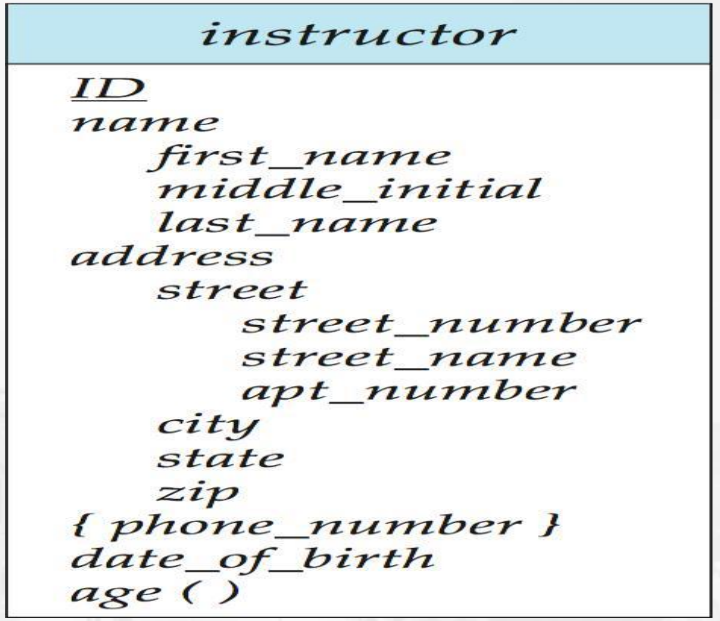
ID	First_name	Middle_initial	Last_name	Street_number	Street_name	Apt_number	city	state	zip	Date_of_birth

Reduction to Relation Schema

Representing Composite and Multi-valued Attributes

A **multivalued attribute** *M* of an entity *E* is represented by a separate schema *EM*

- Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)

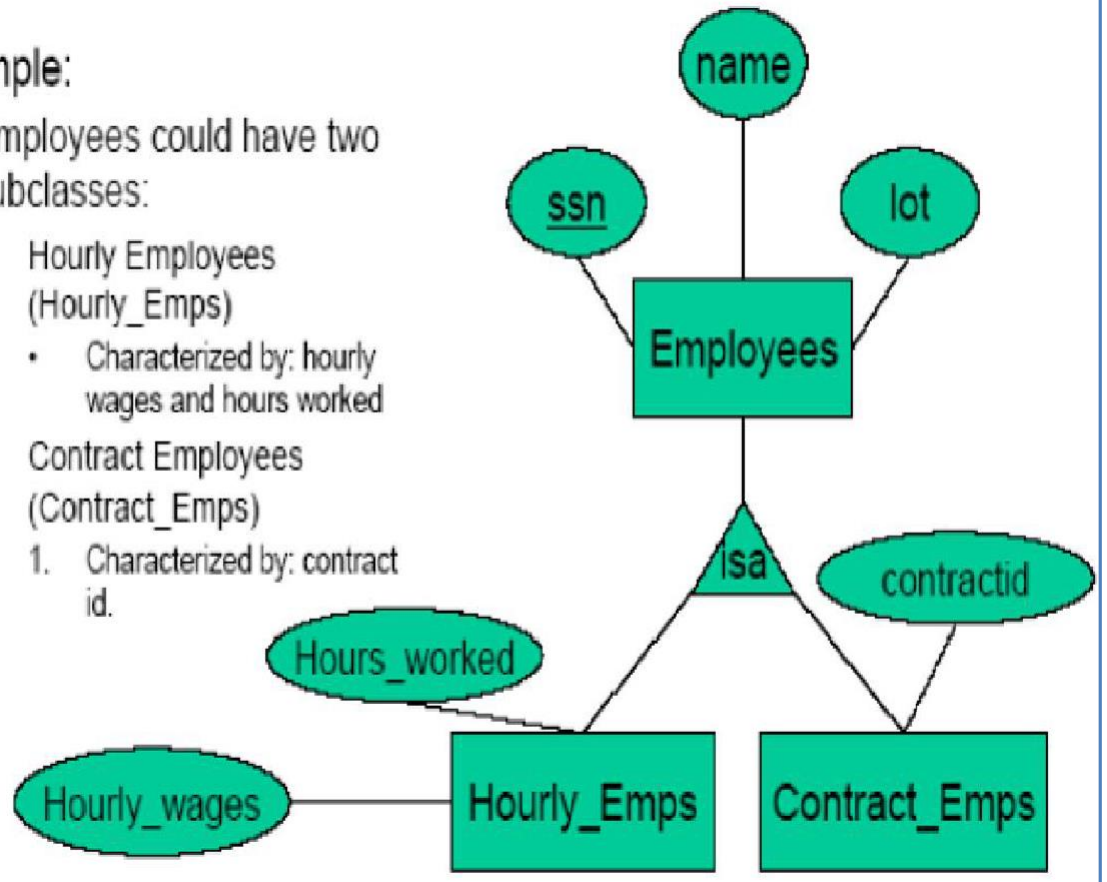


ID	Phone_number
22222	456-7890
22222	123-4567

Extended ER Features Reduction to Relation Schemas

Example:

- Employees could have two subclasses:
  1. Hourly Employees (Hourly_Emps)
     - Characterized by: hourly wages and hours worked
  2. Contract Employees (Contract_Emps)
     1. Characterized by: contract id.



- Two approaches
  - Three tables: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*);
    - We must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
    - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.
- Alternative: Just Hourly_Emps and Contract_Emps.
  - *Hourly_Emps*: *ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*.
  - Each employee must be in one of these two subclasses.

The ‘Staff’ & ‘Course’ Relations

Staff

Staff-ID	Name	Address	ScalePoint	RateOfPay	DOB

Course(CourseCode, Name, Duration, ...)

becomes:

Course

CourseCode	Name	Duration

Reflection Spot 2

Q. Based on the relationship between Staff and Course can we add foreign key in any of above relations in the translation process?

‘Staff’, ‘Course’ & ‘Team’ Relations

Staff

Staff-ID	Name	Address	ScalePoint	RateOfPay	DOB

Team

CourseCode	*Staff-ID*

Course

CourseCode	Name	Duration

Answer : We can’t add a Foreign Key; as Both Relations have a ‘M’ end.

Rule : We must create an ‘artificial’ linking Relation.

The ‘artificial’ Relation (i.e. Team):

- The **Primary Key** is a **composite** of CourseCode & Staff-ID
- **Foreign Key** CourseCode **references** Course.CourseCode
- **Foreign Key** Staff-ID **references** Staff.Staff-ID

4 Relations from 3 Entities?

Student

Enrol-No	Name	Address	OLevelPoints	*Tutor*

Staff

Staff-ID	Name	Address	ScalePoint	RateOfPay	DOB

Team

CourseCode	*Staff-ID*

Course

CourseCode	Name	Duration