CST 204 Database Management Systems

MODULE 1

PREPARED BY SHARIKA T R

Syllabus

- **Introduction & Entity Relationship (ER) Model**
  - Concept & Overview of Database Management Systems (DBMS). Characteristics of Database system,
  - Database Users, structured, semi-structured and unstructured data. Data Models and Schema - Three Schema architecture. Database Languages, Database architectures and classification.
  - ER model - Basic concepts, entity set & attributes, notations, Relationships and constraints, cardinality, participation, notations, weak entities, relationships of degree 3.

Data, Database & DBMS

- Data
  - Known facts that can be recorded and have implicit meaning
- Database
  - The collection of data
- Database-management system (DBMS)
  - is a collection of interrelated data and a set of programs to access those data.
  - General purpose software system that facilitates process of defining, constructing, manipulating, and sharing database

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

- Database systems are designed to manage large bodies of information.
- Management of data involves both storage of information and mechanisms for manipulation of information.
- The database system must ensure the safety of the information stored
- If data are to be shared among several users, the system must avoid possible anomalous results.

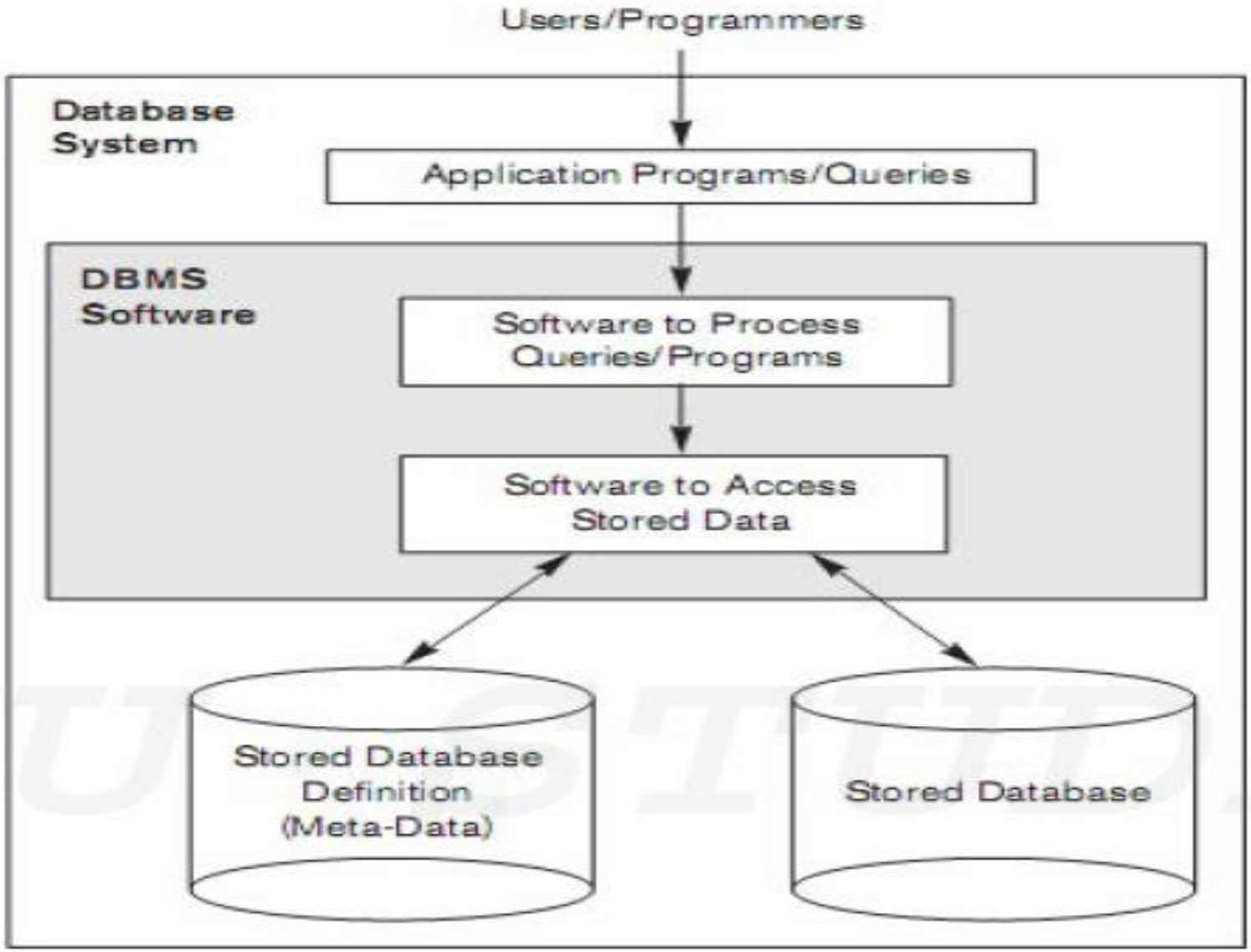
Database implicit properties

- Universe of discourse(UoD) or Miniworld
  - Database represent some aspects of real world
  - Changes to miniworld affects database
- A database is a logically coherent collection of data with some inherent meaning
- A database is designed, built and populated with data for specific purpose

DBMS is a general purpose software system that facilitates process of defining, constructing, manipulating, and sharing database

- An application program access the database by sending queries or request for data to the DBMS.
  - A query typically causes some data to be retrieved
  - A transaction cause some data to be read and some data to be written into database
- We call the database and DBMS software together a database system.

Database System Environment



EXAMLE STUDENT DB

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

Characteristics of the Database Approach

1. Self describing nature of a database system
2. Insulation between programs and data, and data abstraction
3. Support of multiple views of the data

Self-Describing Nature of a Database System

- database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
- This definition is stored in the DBMS catalog
- information stored in the catalog is called meta-data and it describes the structure of the primary database.

Insulation between Programs and Data, and Data Abstraction

- The structure of data files is stored in the DBMS catalog separately from the access programs.
- This property is called program-data independence
- An operation (also called a function or method) is specified in two parts.
- Interface
  - The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters).
- Implementation
  - The implementation (or method) of the operation is specified separately and can be changed without affecting the interface.

- The characteristic that allows program-data independence and program operation independence is called data abstraction.
- A data model is a type of data abstraction that is used to provide conceptual representation
- In database approach detailed structure and organization of each file are stored in a catalog

Support of Multiple Views of the Data

- A database has many users, each user may require a different perspective or view of the database.
- A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

Sharing of Data and Multiuser Transaction Processing

- DBMS must include *concurrency control* software
  - to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct
- DBMS must enforce several transaction properties
  - Isolation property
  - Atomicity property

Isolation Property

- ensures that each transaction appears to execute in isolation from other transactions
- even though hundreds of transactions may be executing concurrently.

Atomicity Property

- ensures that either all the database operations in a transaction are executed or none are.
- Any mechanical or electrical device is subject to failure, and so is the computer system.
- In this case we have to ensure that data should be restored to a consistent state.
- For example an amount of Rs 50 has to be transferred from Account A to Account B.
- Let the amount has been debited from account A but have not been credited to Account B and in the meantime, some failure occurred.
  - So, it will lead to an inconsistent state.
  - So, we have to adopt a mechanism which ensures that either full transaction should be executed or no transaction should be executed i.e. the fund transfer should be atomic.

Concurrent access Problems

- Many systems allows multiple users to update the data simultaneously.
- It can also lead the data in an inconsistent state.
- Suppose a bank account contains a balance of Rs 500 & two customers want to withdraw Rs100 & Rs 50 simultaneously.
- Both the transaction reads the old balance & withdraw from that old balance which will result in Rs 450 , Rs 400 which is incorrect.

Security Problems

- All the user of database should not be able to access all the data.
- For example a payroll Personnel needs to access only that part of data which has information about various employees & are not needed to access information about customer accounts.

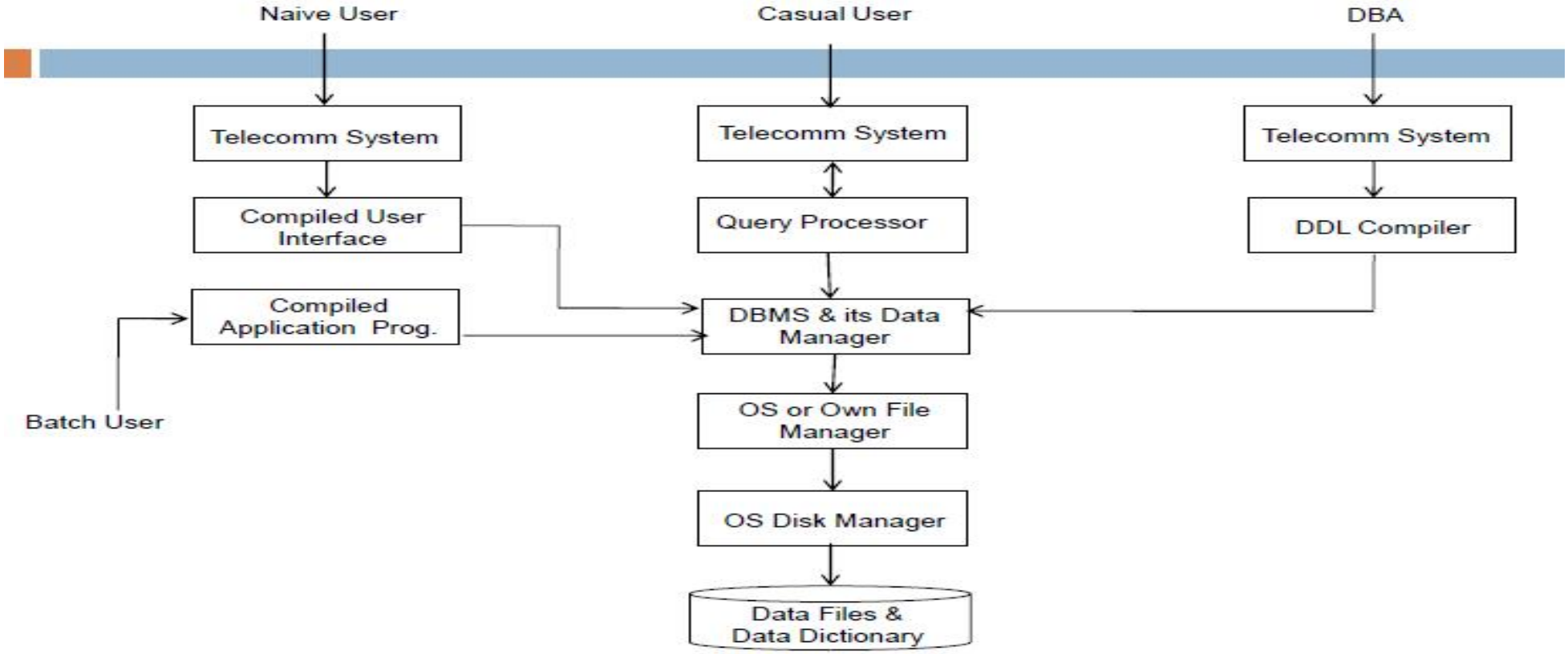
Advantages of DBMS

- Controlling Redundancy
- Restricting Unauthorized Access
- Providing Storage Structures for Efficient
- Query Processing
- Providing Backup and Recovery
- Providing Multiple User Interfaces
- Representing Complex Relationship among Data
- Enforcing Integrity Constraints
- Permitting Inferencing and Actions using Rules

Disadvantages of DBMS

- Cost of Hardware & Software
- Cost of Data Conversion
- Cost of Staff Training
- Appointing Technical Staff
- Database Damage

DBMS Structure



Database Users and Administrators

- A primary goal of a database system is to retrieve information from and store new information in the database.
- People who work with a database can be categorized as database users or database administrators.

Different types of users

- Na¨ıve users
- Application programmers
- Sophisticated users

1. Naïve Users

- unsophisticated users who interact with the system by using predefined user interfaces, such as web or mobile applications
- typical user interface is a forms interface, where the user can fill in appropriate fields of the form
- may also view read *reports* generated from the database.

2. Application programmers

- Are computer professionals who write application programs
- they can choose from many tools to develop user interfaces

3. Sophisticated users

- interact with the system without writing programs.
- Instead they form their requests either using a database query language or by using tools such as data analysis software.
- Analysts who submit queries to explore data in the database fall in this category

Database Administrator

- A person who has such central control over the system is called a database administrator (DBA).
- The functions of a DBA include:
  - Schema definition.
    - The DBA creates the original database schema by executing a set of data definition statements in the DDL.
  - Storage structure and access-method definition.
    - The DBA may specify some parameters pertaining to the physical organization of the data and the indices to be created.

- Schema and physical-organization modification
  - The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization
  - or to alter the physical organization to improve performance
- Granting of authorization for data access.
  - The authorization information is kept in a special system structure that the database system consults whenever a user tries to access the data in the system.

ACTORS ON THE SCENE

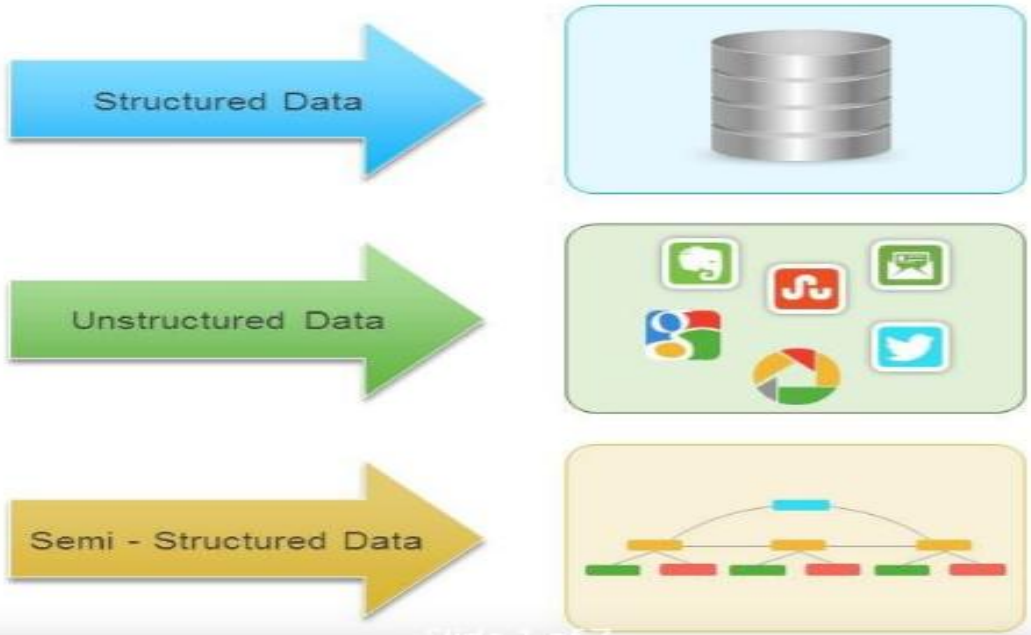
- The people whose jobs involve the day-to-day use of a large database are called as the actors on the scene.
  1. Database Administrators
  2. Database Designers
  3. End Users
  4. System Analyst and Application Programmers(Software engineers)

WORKERS BEHIND THE SCENE

- The people who work to maintain the database system environment but who are not actively interested in the database contents as part of their daily job are called as the workers behind the scene

1. DBMS system designers and implementers
2. Tool developers
3. Operators and maintenance personnel (system administration personnel)

Structured, Semi-structured and Unstructured data



Structured data

- Represented in a strict format
- It has been organized into a formatted repository that is typically a database.
- It concerns all data which can be stored in database SQL in a table with rows and columns
- . *Example:* Relational data

Semi-Structured data

- information that does not reside in a relational database but that have some organizational properties that make it easier to analyze
- With some process, you can store them in the relation database
- but Semi-structured exist to ease space.
- *Example*: XML data
  - digital photographs, the image does not have a pre-defined structure itself but has certain structural attributes making them semi-structured
  - For instance, it is taken from a smartphone; it would have some structured attributes like geotag, device ID, and DateTime stamp. After being stored, images can also be assigned tags such as ‘pet’ or ‘dog’ to provide a structure.

Unstructured data

- data which is not organized in a predefined manner or does not have a predefined data model,
- thus it is not a good fit for a mainstream relational database
- there are alternative platforms for storing and managing, it is increasingly prevalent in IT systems and is used by organizations in a variety of business intelligence and analytics applications.
- *Example*: Word, PDF, Text, Media logs.

FEATURES	STRUCTURED	SEMI STRUCTURED	UNSTRUCTURED
Format Type	Relational Database	HTML, XML, JSON	Binary, Character
Version Management	Rows, columns, tuples	Not as common – graph is possible	Whole data
Implementation	SQL	Anonymous nodes	-
Robustness	Robust	Limited robustness	-
Storage Requirement	Less	Significant	Large
Applications	DBMS, RDF, ERP system, Data Warehouse, Apache Parquet, Financial Data, Relational Table	Server Logs, Sensor Output	No SQL, Video, Audio, Social Media, Online Forums, MRI, Ultrasound

Data Models

- a collection of concepts that can be used to describe the structure of a database
- *structure of a database* we mean the data types, relationships, and constraints that should hold on the data.
- Most data models also include a set of **basic operations** for specifying retrievals and updates on the database.

Categories of Data Models

- High-level or conceptual data models
- Low-level or physical data models
- Representational (or implementation) data models

High-level or conceptual data models

- provide concepts that are close to the way many users perceive data
- use concepts such as entities, attributes, and relationships
- An **entity** represents a real-world object or concept
- An **attribute** represents some property of interest that further describes an entity,
- A **relationship** among two or more entities represents an interaction among the entities

Low-level or physical data models

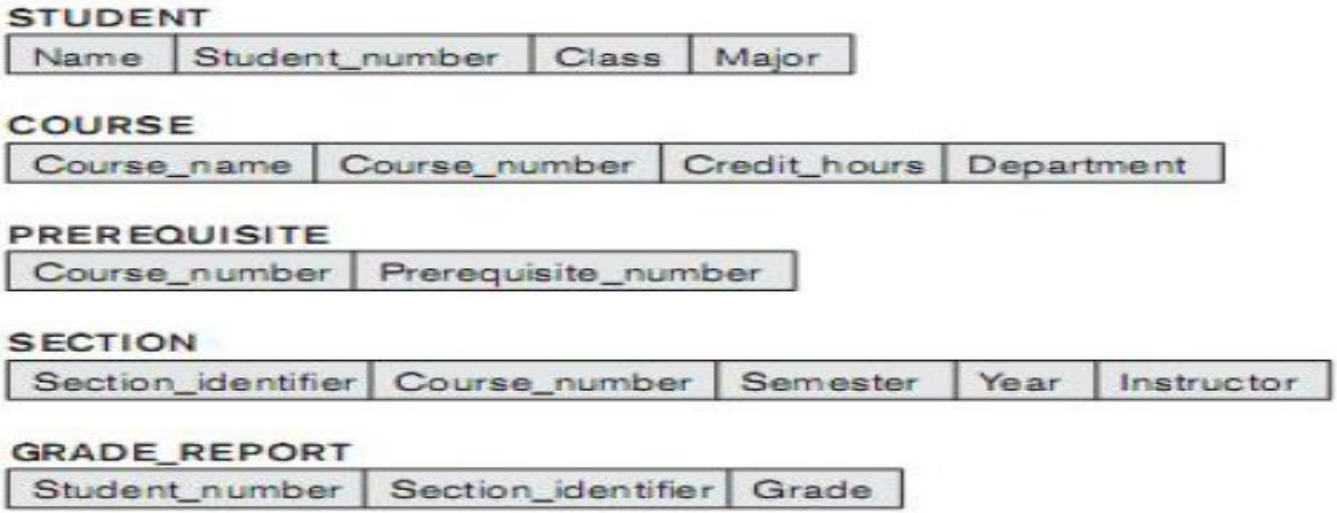
- Provide concepts that describe the details of how data is stored in the computer.
- Concepts provided by low-level data models are generally meant for computer specialists, not for typical end users.
- Describe how data is stored in the computer by representing information such as record formats, record orderings, and access paths.
- An **access path** is a structure that makes the search for particular database records efficient.

Representational (or implementation) data models

- Which provide concepts that may be understood by end users but that are not too far removed from the way data is organized within the computer.
- It hides some details of data storage but can be implemented on a computer system in a direct way.
- used most frequently in traditional commercial DBMSs, and they include the widely-used **relational data model**, the **network** and **hierarchical models**
- sometimes called **record-based data models**

Schemas, Instances, and Database State

- The description of a database is called the **database schema,** which is specified during database design and is not expected to change frequently
- A displayed schema is called a **schema diagram.** We call each object in the schema a **schema construct.**
- The data in database at particular instant or moment of time is called **database state** or **snapshot**



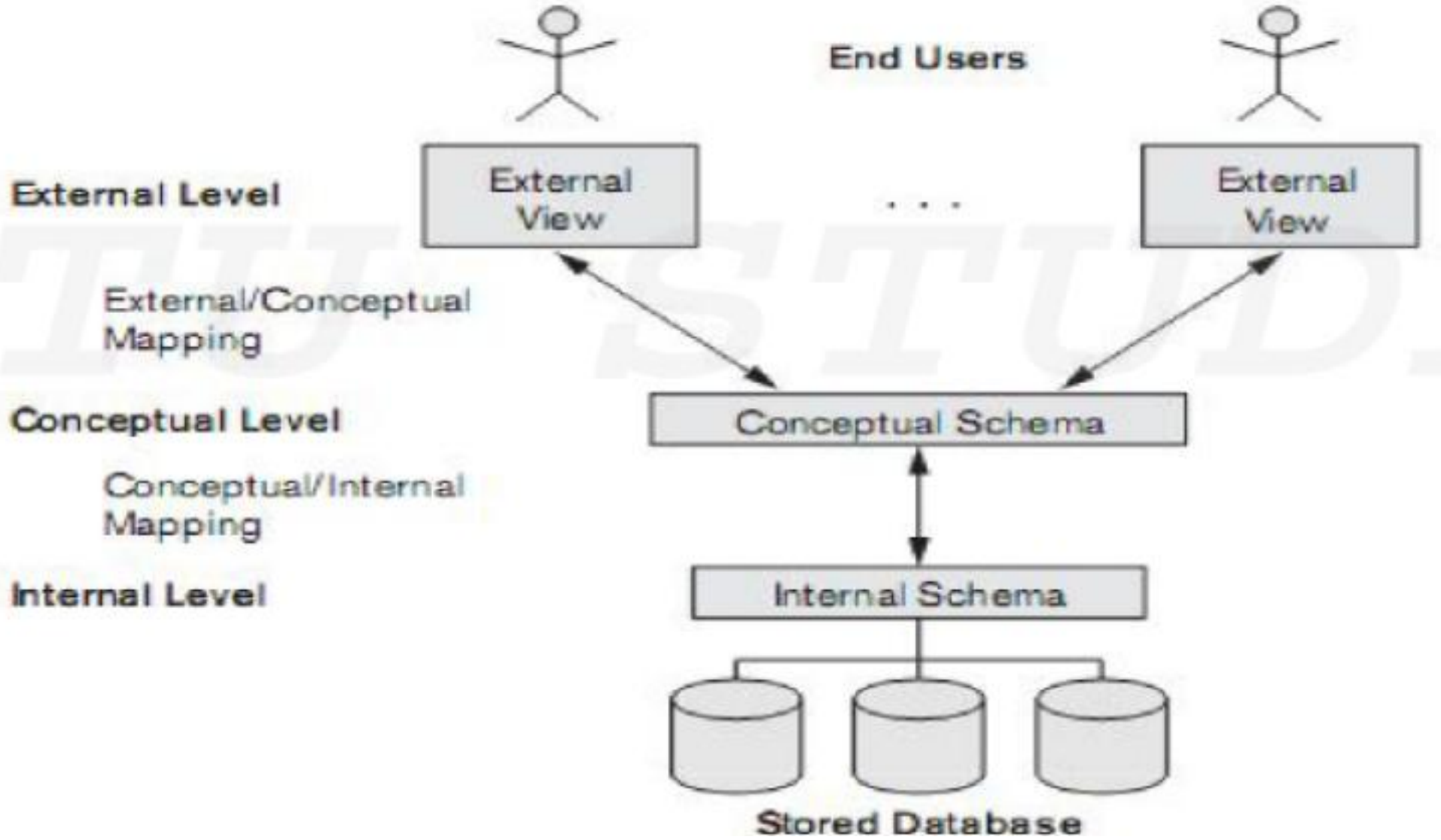
Database schema

Name	Student_number	Class	Major
Ram	CS001	R4	CSE
Shyam	CS002	R4	CSE

Database state

- The schema is not supposed to change frequently, but it is not uncommon that changes occasionally need to be applied to the schema as the application requirements change. It is called **schema evolution**.

The Three-Schema Architecture



Internal level

- The **internal level** has an **internal schema,** which describes the **physical storage** structure of the database.
- The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Conceptual level

- Describes the structure of the whole database for a community of users.
- The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- A high-level data model or an implementation data model can be used at this level.

External or view level

- The **external** or **view level** includes a number of **external schemas** or **user views.**
- Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
- A high-level data model or an implementation data model can be used at this level.

- In a DBMS based on the three-schema architecture, each user group refers only to its own external schema.
- Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.
- If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.

Mappings

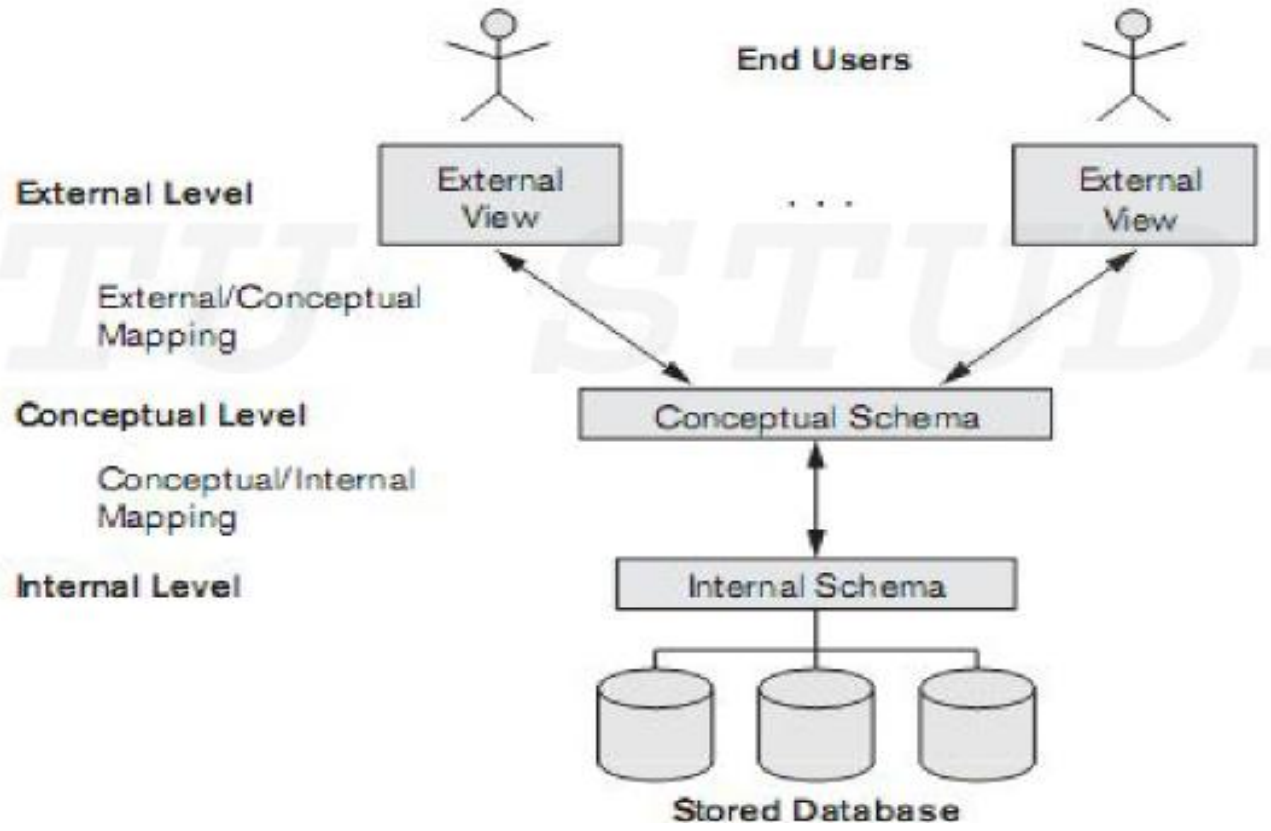
- The processes of transforming requests and results between levels are called **mappings.**
- These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views.
- a certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

Data Independence

- The capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
- Two types of data independence:
  1. Logical data independence
  2. Physical data independence

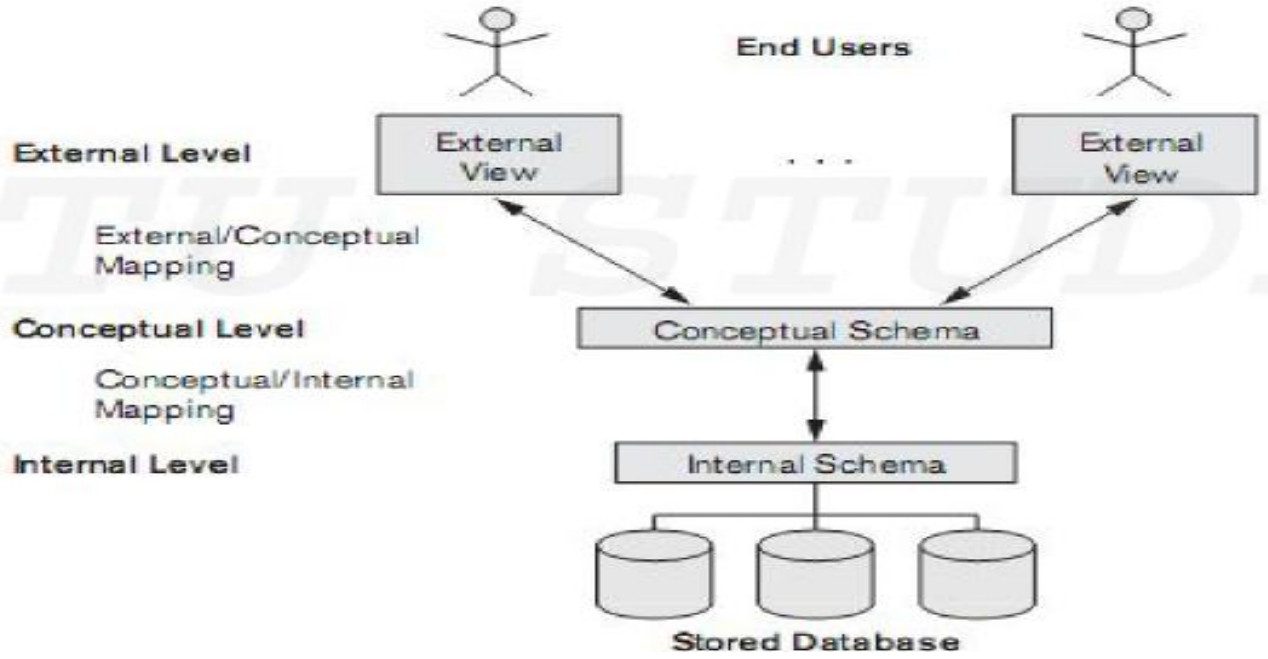
Logical data independence

- Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.



Physical data independence

- Physical data independence is the capacity to change the internal schema without having to change the conceptual schema.
- Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.



Database Languages and Interfaces
DBMS Languages

- **Data definition language (DDL),** is used by the DBA and by database designers to define conceptual and internal schemas schemas.
- **Storage definition language (SDL),** is used to specify the internal schema.
- **View definition language (VDL),** to specify user views and their mappings to conceptual schema
- **Data Manipulation Language(DML)** is used for retrieval, insertion, deletion, and modification of the data

- There are two main types of DMLs.
- **High-level or nonprocedural DML** can be used on its own to specify complex database operations concisely.
- **Low-level or procedural DML** must be embedded in a general-purpose programming language. This type of DML typically retrieves individual records or objects from the database and processes each separately. Therefore, it needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records. Low-level DMLs are also called record-at-a-time DMLs

DBMS Interfaces

- User-friendly interfaces provided by a DBMS may include the following:

1. Menu-Based Interfaces for Web Clients or Browsing.
2. Forms-Based Interfaces.
3. Graphical User Interfaces (GUI)
4. Natural Language Interfaces
5. Speech Input and Output
6. Interfaces for Parametric Users.

Database System Environment

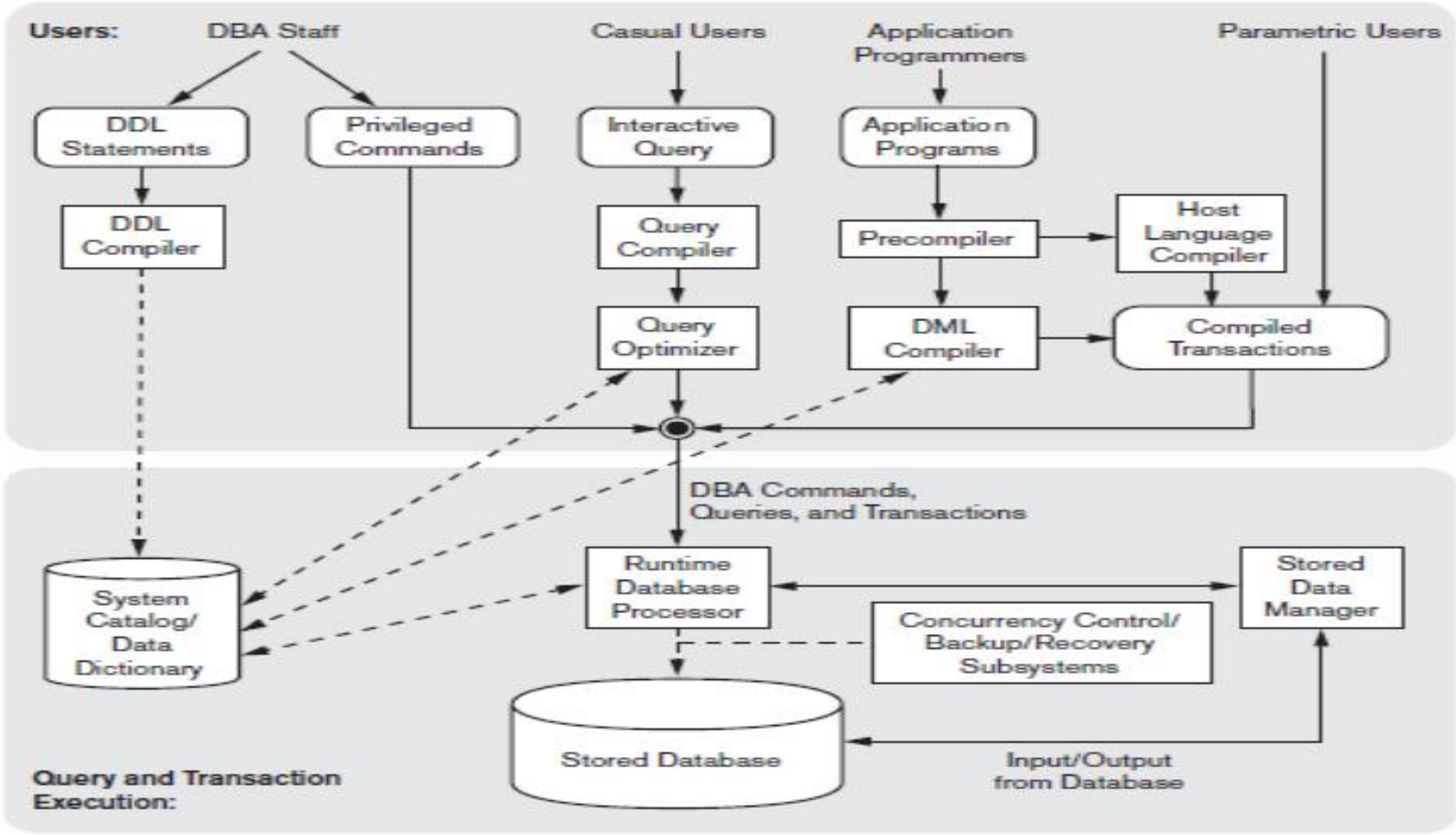


Figure 2.3
Component modules of a DBMS and their interactions.

DBMS component modules

- The entire figure is divided into two parts
  - Top part of the figure refers to various users of the database enviornment and their interfaces
  - Lower part shows the internals of the DBMS responsible for storage of data and processing of transaction

1. Users

- DBA,
- Casual users
- Application programmers
- Parametric users
  - Do data entry work by supplying parameters to predefined transactions
- DBA staff
  - Works on defining the database and tuning it by making changes to its definition using DDL and other privileged commands

2. DDL Compiler

- It processes schema definition specified in DDL
- Stores description of schema(meta-data) in DBMS Catalog

3. Query Compiler

- Interactive queries are passed and validated for correctness of query syntax
- Compiles into an internal form

4. Query Optimizer

- Rearrangement and possible reordering of operations,
- Elimination of redundancies and use of correct algorithm and indices during execution
- It consults the system catalog for statistical and other physical information about stored data and generate executable code that performs the necessary operation for query and make call on run time processor

5. Precompiler

- Precompiler extract DML commands from application programs written in host programming language(java, c, python..)

6. DML Compiler

- Collect DML commands from precompiler for compilation into object code for database access

7. Host Language compiler

- It collect other programs excluding DML Commands from precompiler and compiler them to produce object code

8.Compiled transaction/canned transaction

- Object codes commands and rest of the program are linked forming a canned transaction whose executable code include call to runtime database processor
- Canned transaction are executed repeatedly by parametric users who simply supply parameter to transaction
- Each execution is a separate transaction

- Lower part of architecture includes
  1. Runtime database processor
  2. Stored data manager
  3. Concurrency control/ Data dictionary
  4. Stored database

1. Runtime database processor

- It execute privileged commands, the executable query plans and canned transactions with runtime parameter
- It handle database access at runtime, it receives retrieval or update operations and causes them out on the database
- It works with the system catalog and may update it with statistics
- It also works with stored data manager

2. Stored data manager

- It controls the access to DBMS information that is stored in disk, whether it is part of database or catalog
- It uses basic operating system services for carrying out low level input/output operations between disk and main memory

3.Concurency control/ Backup /recovery subsystem

- It act as a separate module
- They are integrated into working of runtime database processor for the purpose of transaction management
- Backup system creates a backup copy of the database
- Backup copy can be used to restore the database in case of system failure

4.System Catalog/ Data Dictionary

- It store meta-data about structure of database in particular schema of database

Database Architecture Classification

- DBMS architecture are of different types
  1. Centralized database
  2. Basic client/server architecture
  3. Two tire client server architecture
  4. Three and n-tire architecture for web applications

1.Centralized DB Architecture

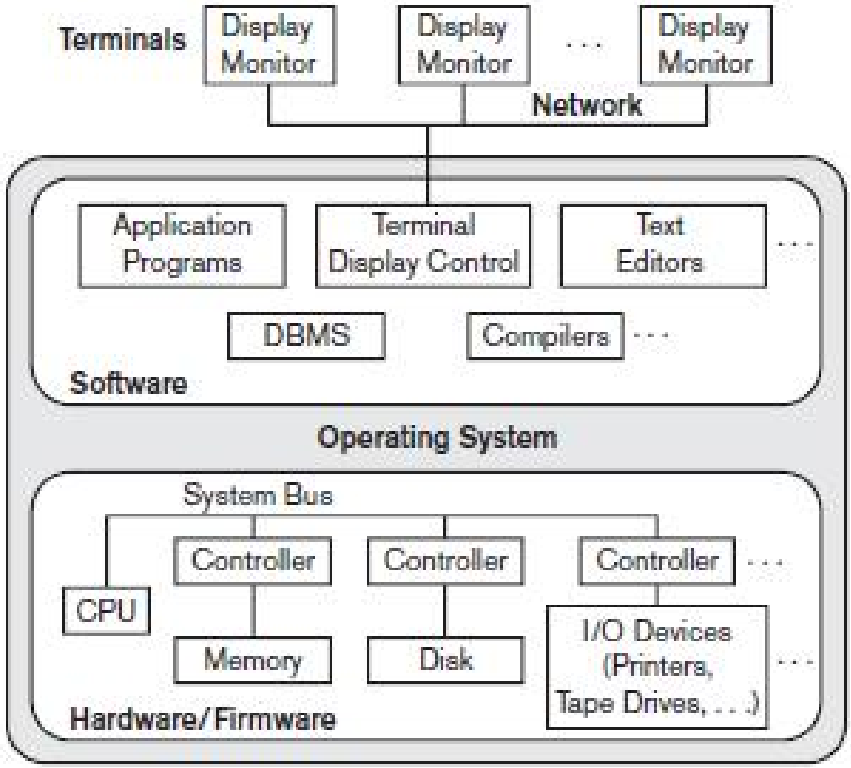


Figure 2.4
A physical centralized architecture.

2. Basic Client Server Architecture

- Client is a user machine that provide user interface capabilities and local processing
- Server system provide service to client such as file

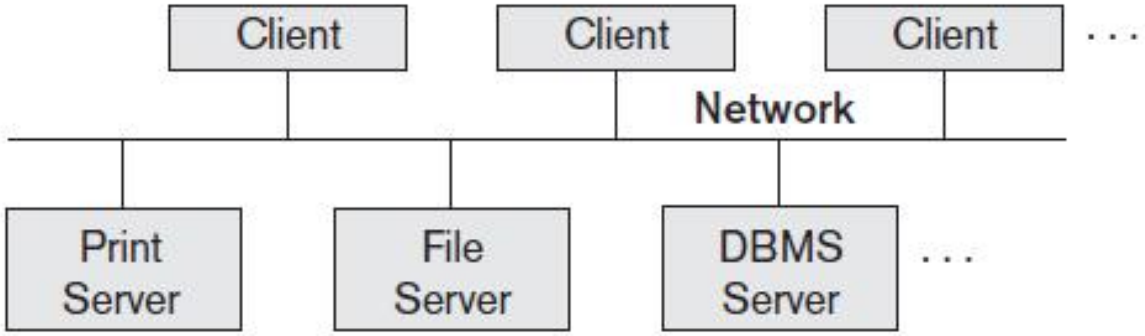


Figure 2.5
Logical two-tier client/server architecture.

3. Two-tier client/server architecture

- Software components are distributed over two systems: client and server

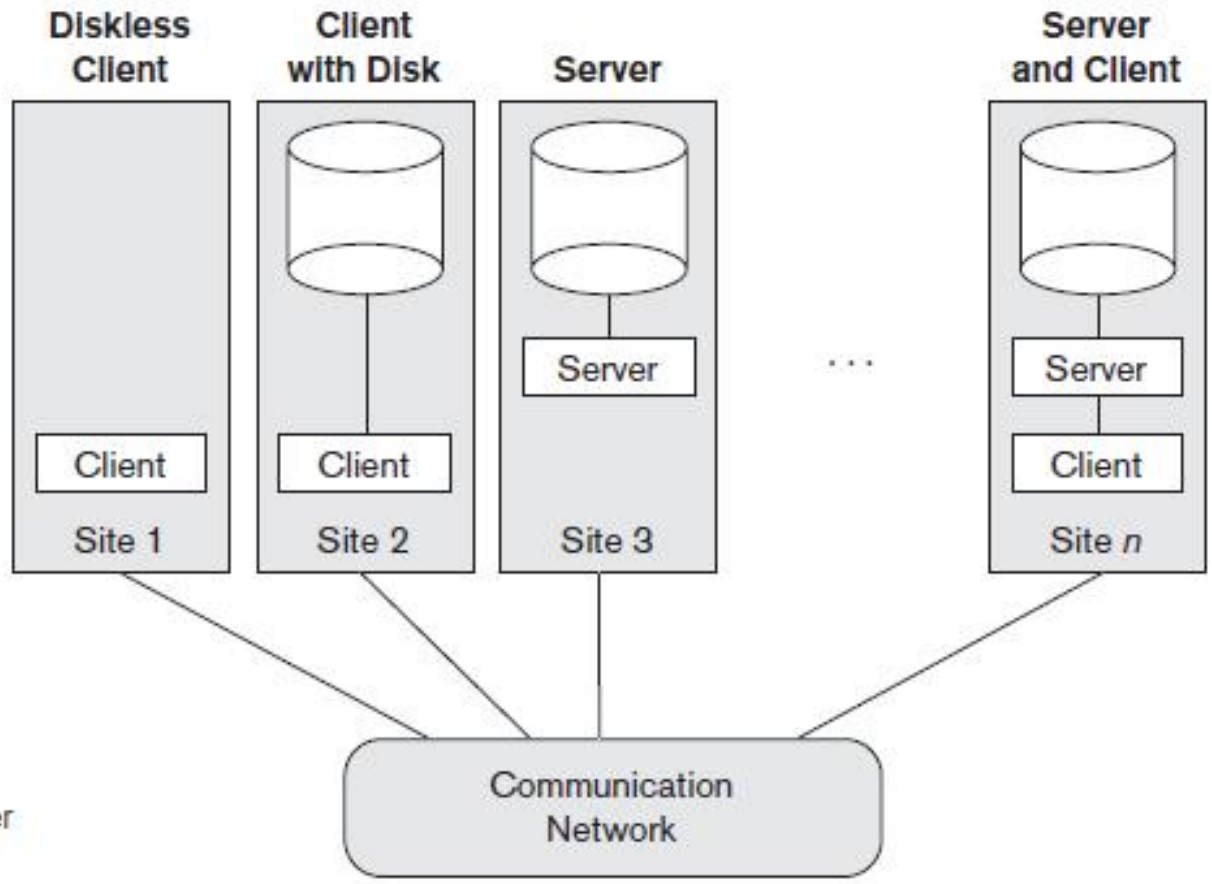


Figure 2.6
Physical two-tier client/server architecture.

4. Three tier and n-tier architecture

- Adds an intermediate layer between client and database server called application server

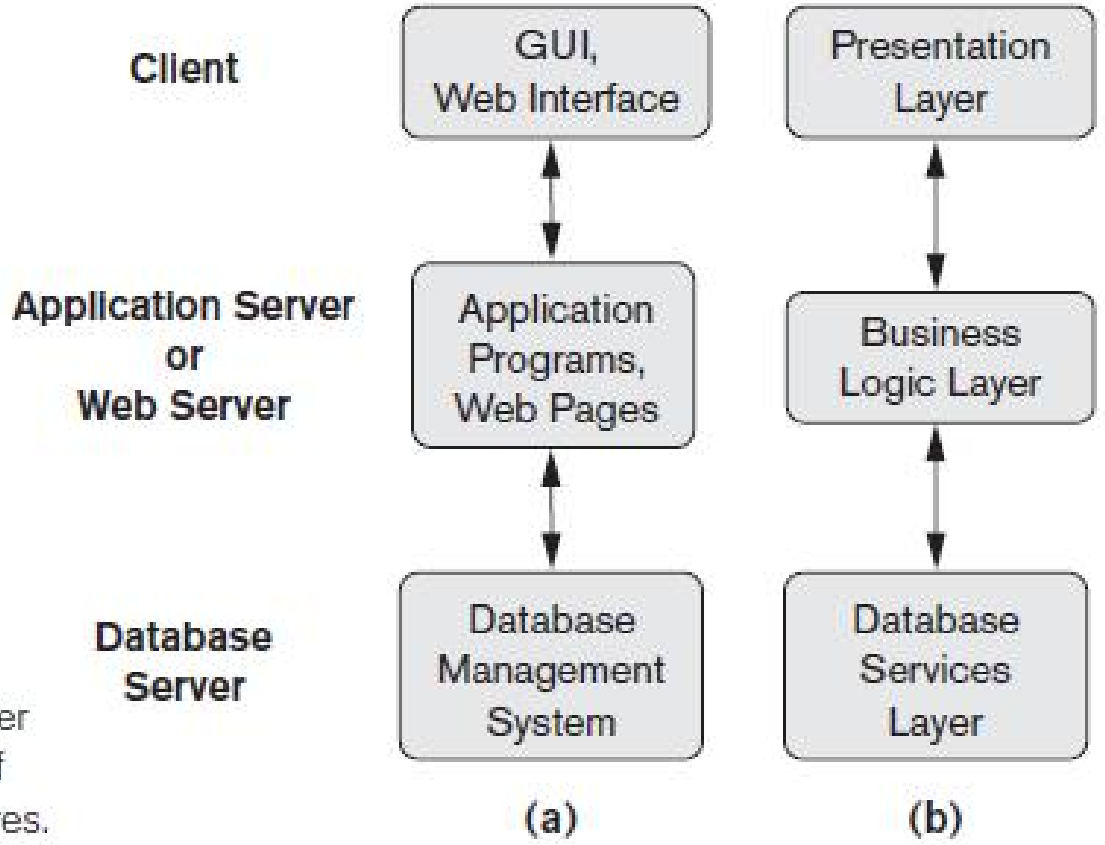


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

DATABASE MODELING USING ENTITY-RELATIONSHIP

ER MODEL

ER Diagram

- It is not a technical method
- High level conceptual data model
- It is used for conceptual data design of database applications
- Collection of **entities** and their properties called **attributes** and relationship between them
- Diagrammatic representation and easy to understand for non technical users

Highlevel conceptual data model for data

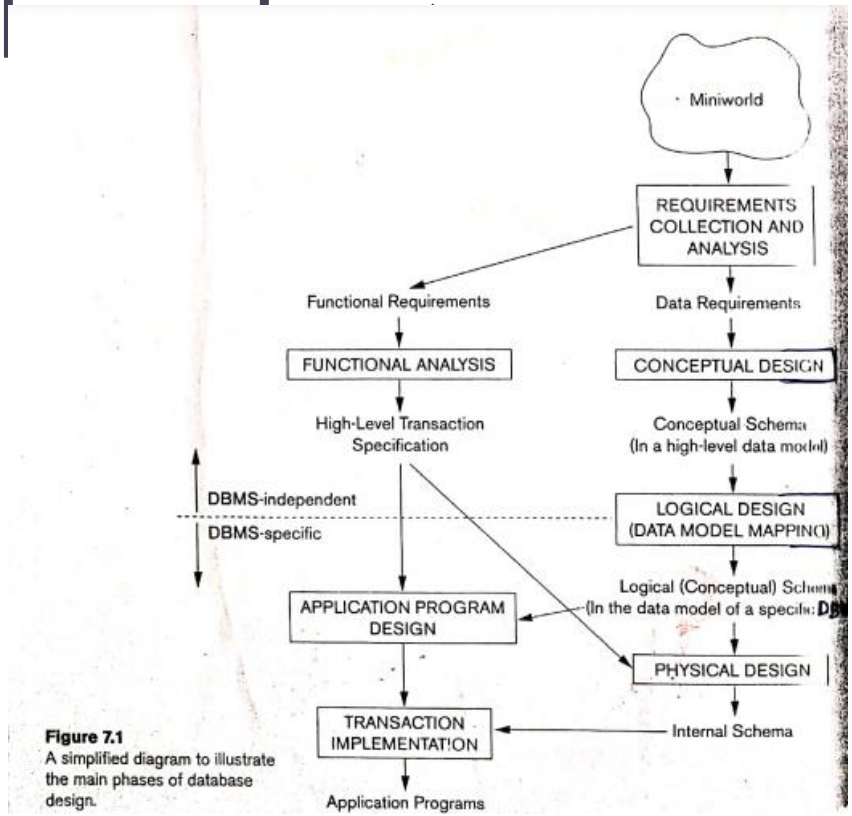


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

1.Requirement collection and analysis

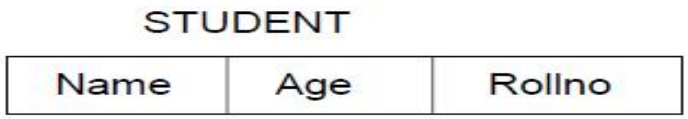
Entity

- The basic object that the ER model represents
- A thing in real world with existence
- Entity is distinguished from other objects on basis of attributes
- Entities can be tangible and intangible

Entity Type

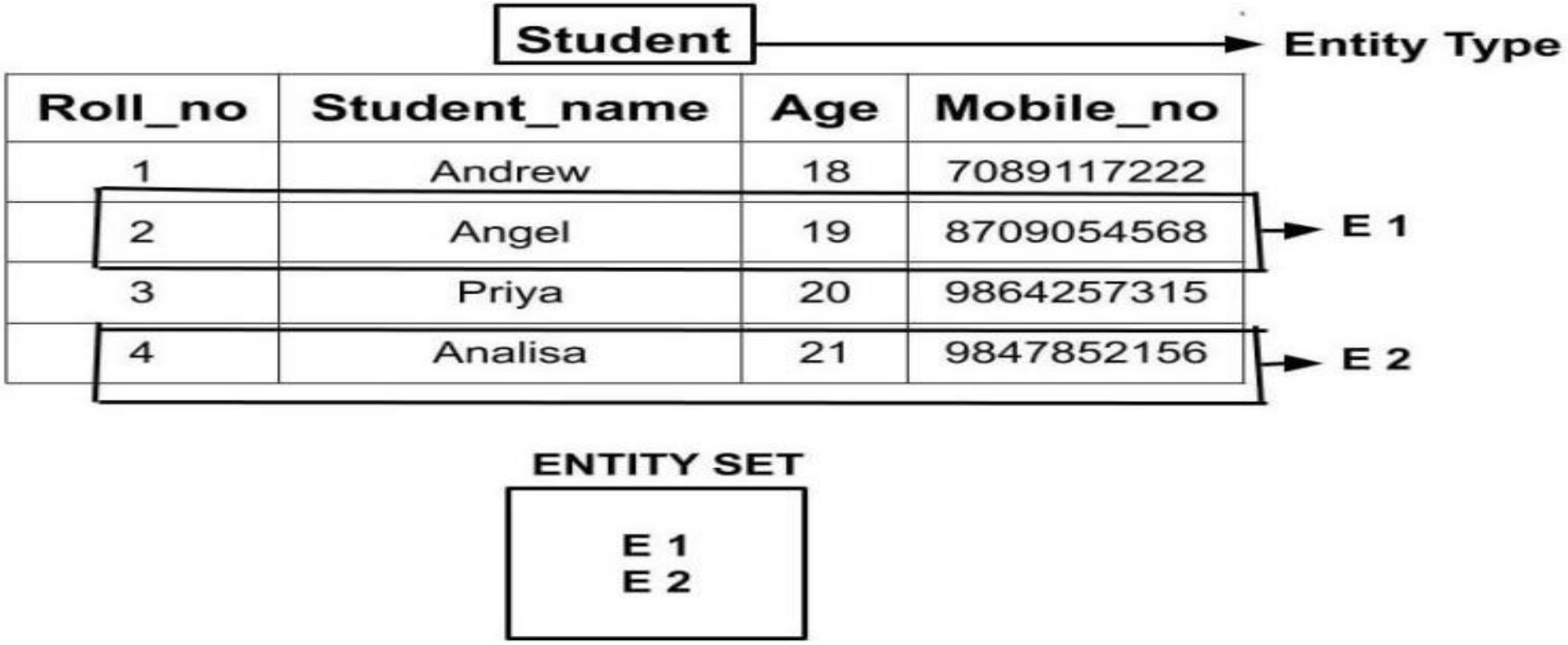
- The entity type is a collection of the entity having similar attributes.
- an entity type in an ER diagram is defined by a name and a set of attributes
- *We use a rectangle to represent an entity type in the*

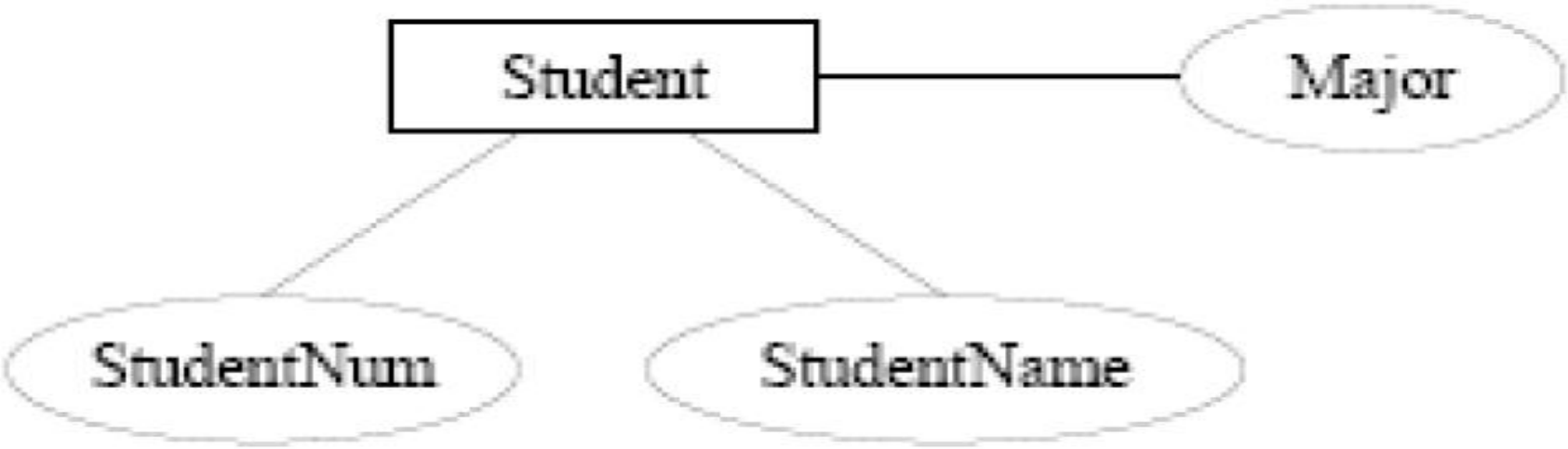
Ex: STUDENT, UNIVERSITY



Entity set

- The collection of same type of entities that is their attributes are same is called entity set
- We can say that entity type is a superset of the entity set as all the entities are included in the entity type


Student
Entity Type
Roll_no
Student_name
Age
Mobile_no
1
Andrew
18
7089117222
2
Angel
19
8709054568
E 1
3
Priya
20
9864257315
4
Analisa
21
9847852156
E 2
ENTITY SET
E 1
E 2


Student
Major
StudentNum
StudentName

Attributes

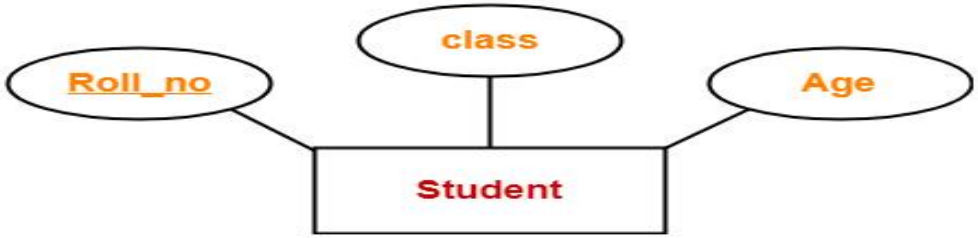
- The properties of entity that basically describes it
- Attributes describes characteristics of entity
- Suppose we have a entity EMPLOYEE and its attributes are ENO, ESAL, ENAME etc..
- Attributes have some set of allowed or permitted values called Domain
- Attributes are represented by OVAL
- Each attribute of an entity set is associated with domain that means the set of values that can be assigned to that attribute for an entity

Types of attribute

- Simple attribute vs Composite attribute
- Single valued vs Multivalued attributes
- Stored vs Derived attributes

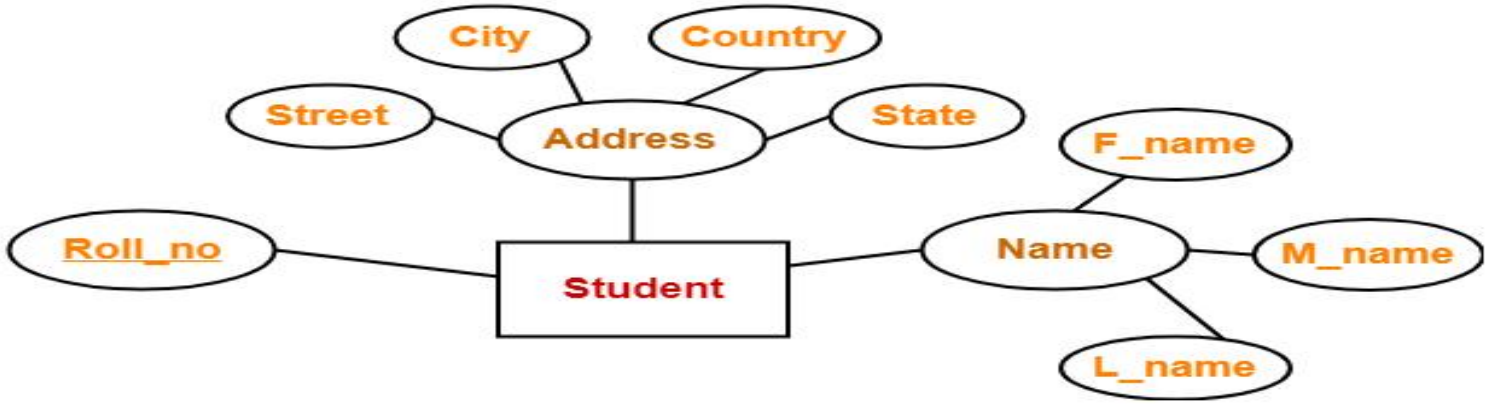
Simple attribute vs Composite attribute

- Simple attributes
  - Attributes which are not divisible that is they cannot be divided
  - Eg: City, State, etc



Here, all the attributes are simple attributes as they can not be divided further.

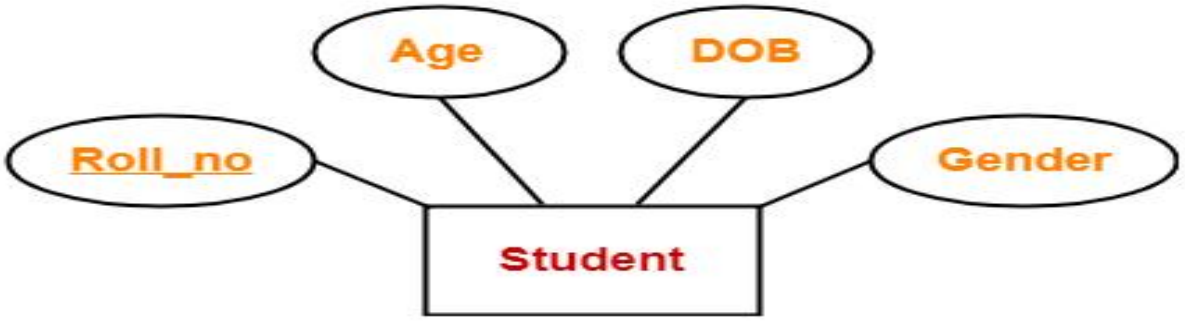
- Composite Attribute
  - Attributes that can be divided into smaller sub parts
  - Example: Name attribute can be divided into FirstName, MiddleName, LastName



Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.

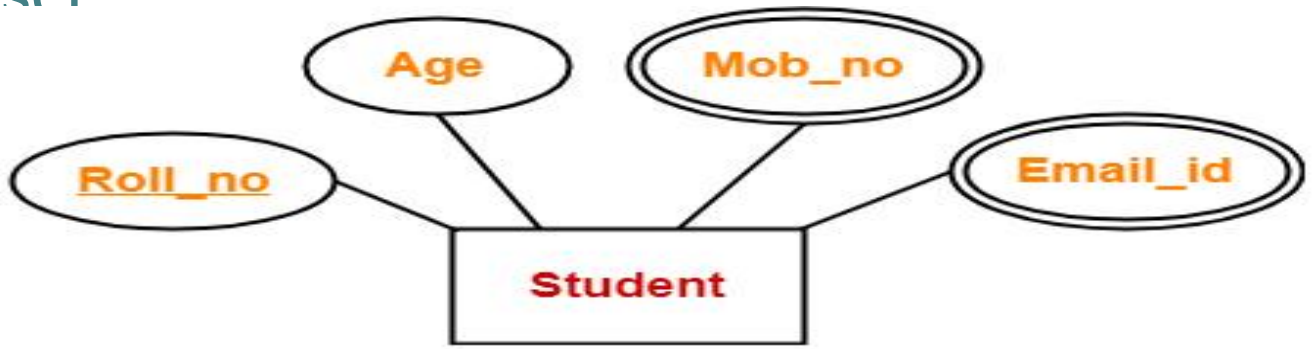
Single valued vs Multivalued Attributea

- Single Valued
  - Attributes which are having single values
  - Example: Age



Here, all the attributes are single valued attributes as they can take only one specific value for each entity.

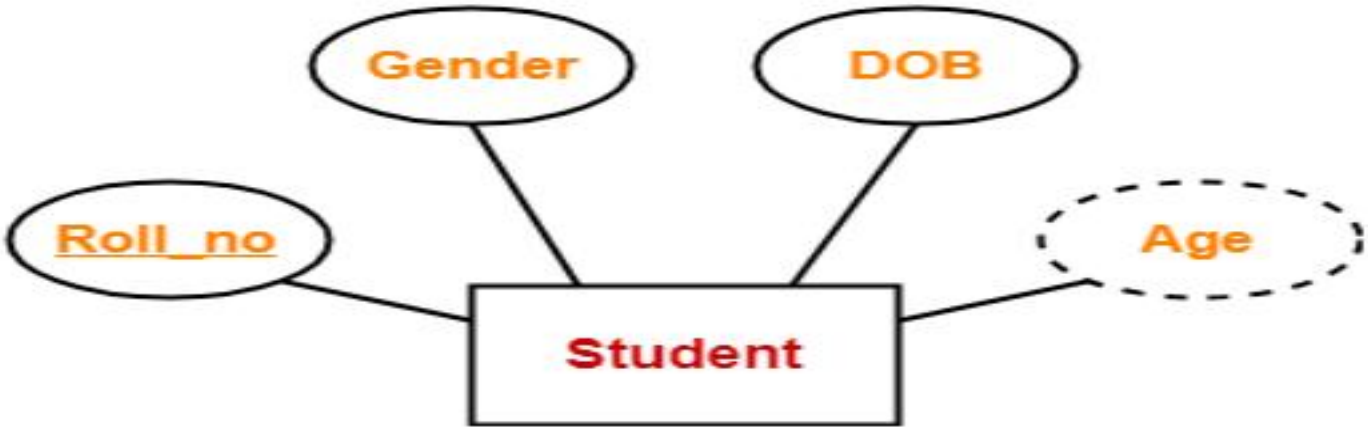
- Multi Valued Attributes
  - Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set



Here, the attributes "Mob_no" and "Email_id" are multi valued attributes as they can take more than one values for a given entity.

Stored Vs Derived Attribute

- In some cases, two (or more) attribute values are related ,for example, the Age and Birthdate attributes of a person.
- For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's birthdate.
- The Age attribute is hence called a derived attribute and is said to be derivable from the birthdate attribute, which is called a stored attribute .



Here, the attribute “Age” is a derived attribute as it can be derived from the attribute “DOB”.

Complex Attribute

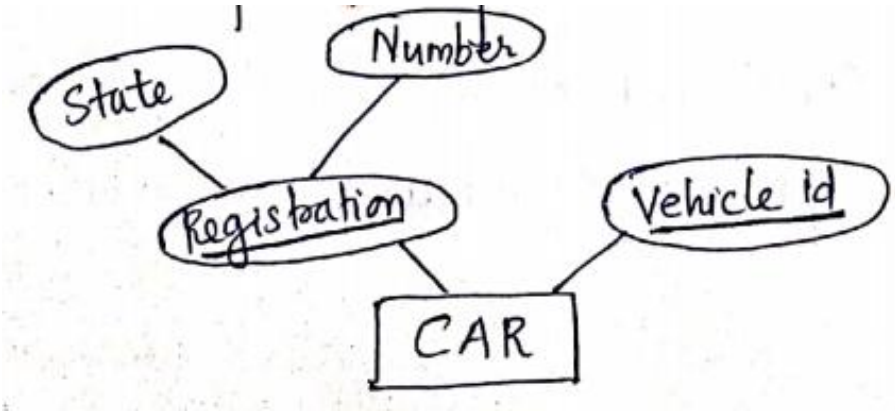
- Complex attribute is a combination of composite and multi-valued attributes.
- Complex attributes are represented by { } and composite attributes are represented by ( ).
- Example: Address_phone attribute will hold both the address and phone_no of any person.
- Example: {(2-A, St-5, Sec-4, Bhilai), 239812 4}

Null Valued Attributes

- Null value is a value which is not inserted but it does not hold zero value.
- The attributes which can have a null value called null valued attributes.
- Example: Mobile_no attributes of a person may not be having mobile phones.

Key attribute in an entity type

- Key attributes will be having a unique value for each entity of that attribute.
- It identifies every entity in the entity set.
- Key attribute will never be a null valued attribute.
- Any composite attribute can also be a key attribute.
- There could be more than one key attributes for an entity type.
- Example: roll_no, enrollment _no

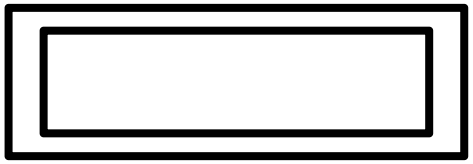


Domain of value set of an attribute

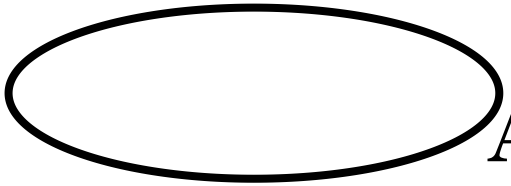
- Domain of an attribute is the allowed set of values of that attribute.
- Example: if attribute is ‘grade’, then its allowed values are A,B,C,F.
- Grade ={A, B,C,F}
- Mathematically, an attribute A of entity set E whose value set is V can be defined as a function from E to power set P(V) of V
- A:E ---> P(V)
- The value of attribute A for entity e is as A(e)

NOTATIONS USED IN E-R DIAGRAM

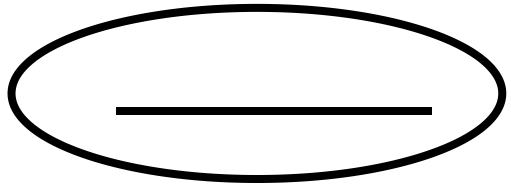
Entity



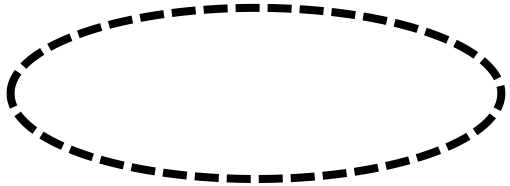
Weak Entity



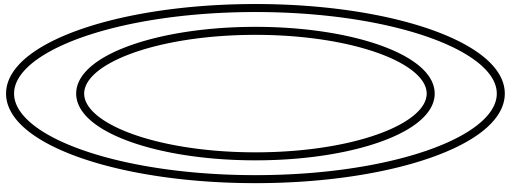
Attribute



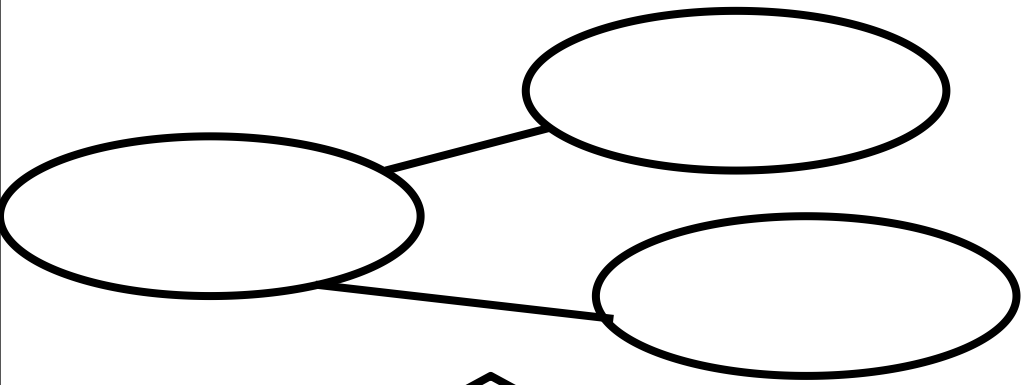
Key attribute



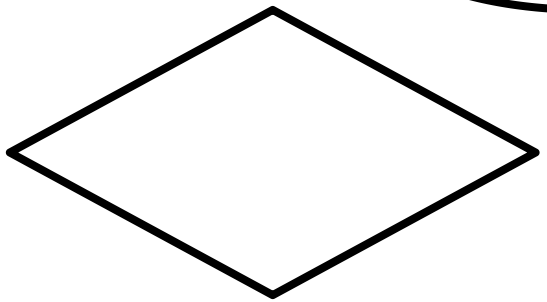
Derived attribute



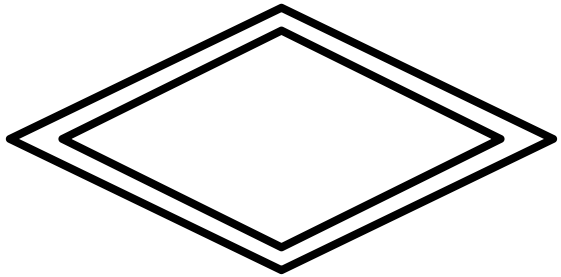
Multivalued Attribute



Composite Attribute



Relationship type



Identifying Relationship

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Relationship

- Relates two or more distinct entities with a specific meaning.
- When attribute of one entity refers to another entity it is a relationship
- It is an association between two or more entities of same or different entity set
  - For example, EMPLOYEE John works on the ProductX PROJECT or
  - EMPLOYEE Franklin manages the Research DEPARTMENT.
- **Terms used:**
  - Relationship type

Relationship type

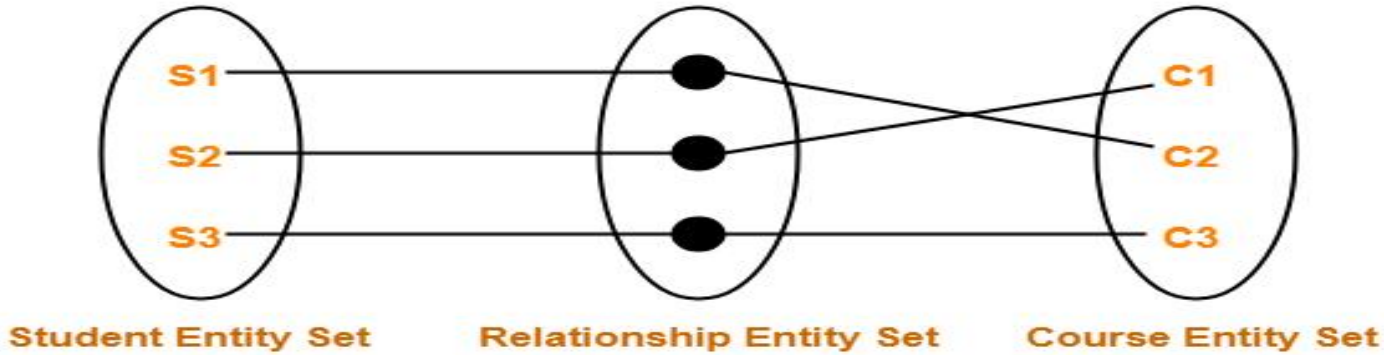
- A set of similar types of relationship
- Relationship type among n entity types E1,E2,..En defines a set of associationas or a relationship set among entities from these entity types

'Enrolled in' is a relationship that exists between entities **Student** and **Course**.

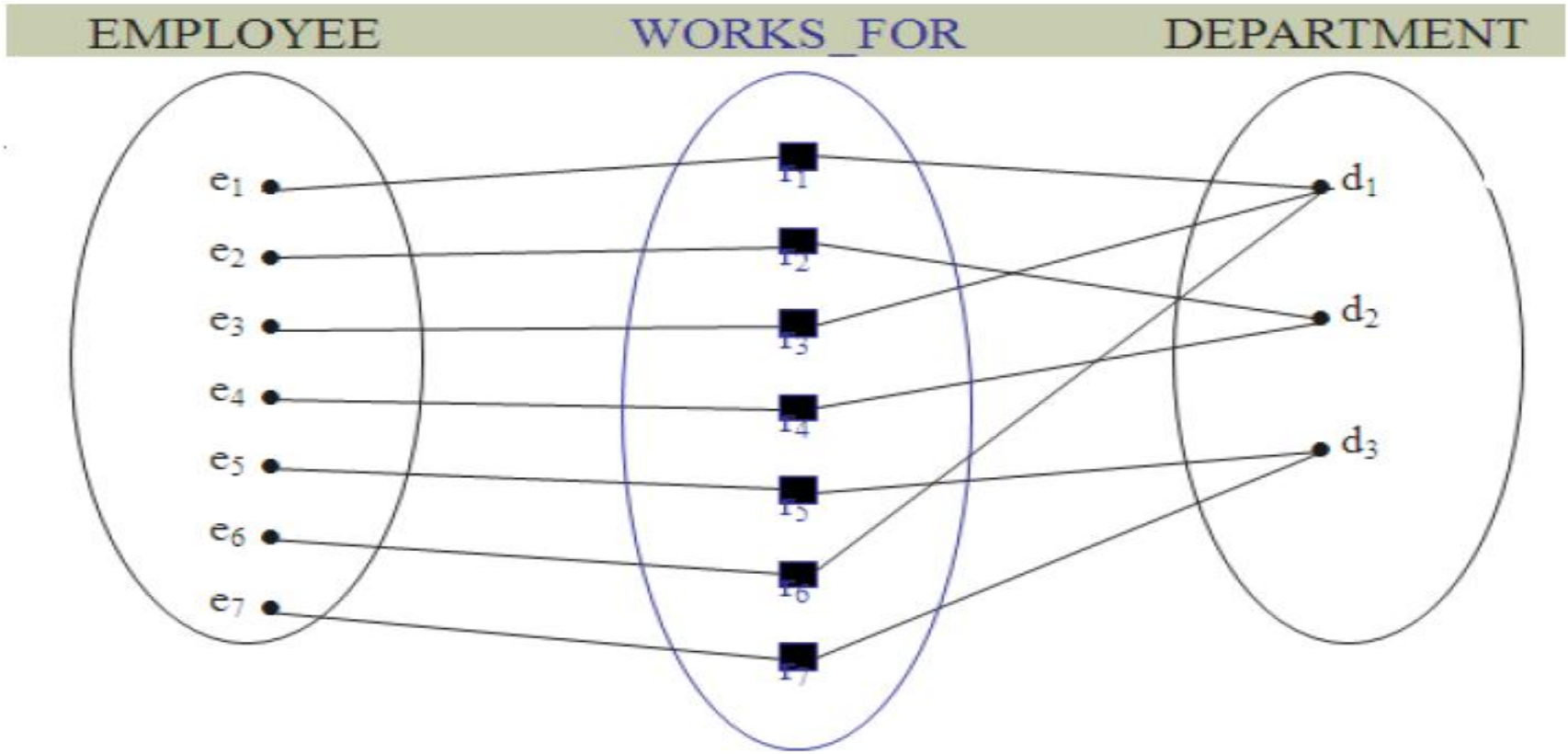
- Relationship type and set are referred as R
- Relationship set is a set of relationship instance ri where each ri asccociate with n individual entities (e1,e2...en)
- each entity ej in ri is member of entity set Ej 1<=j<=n

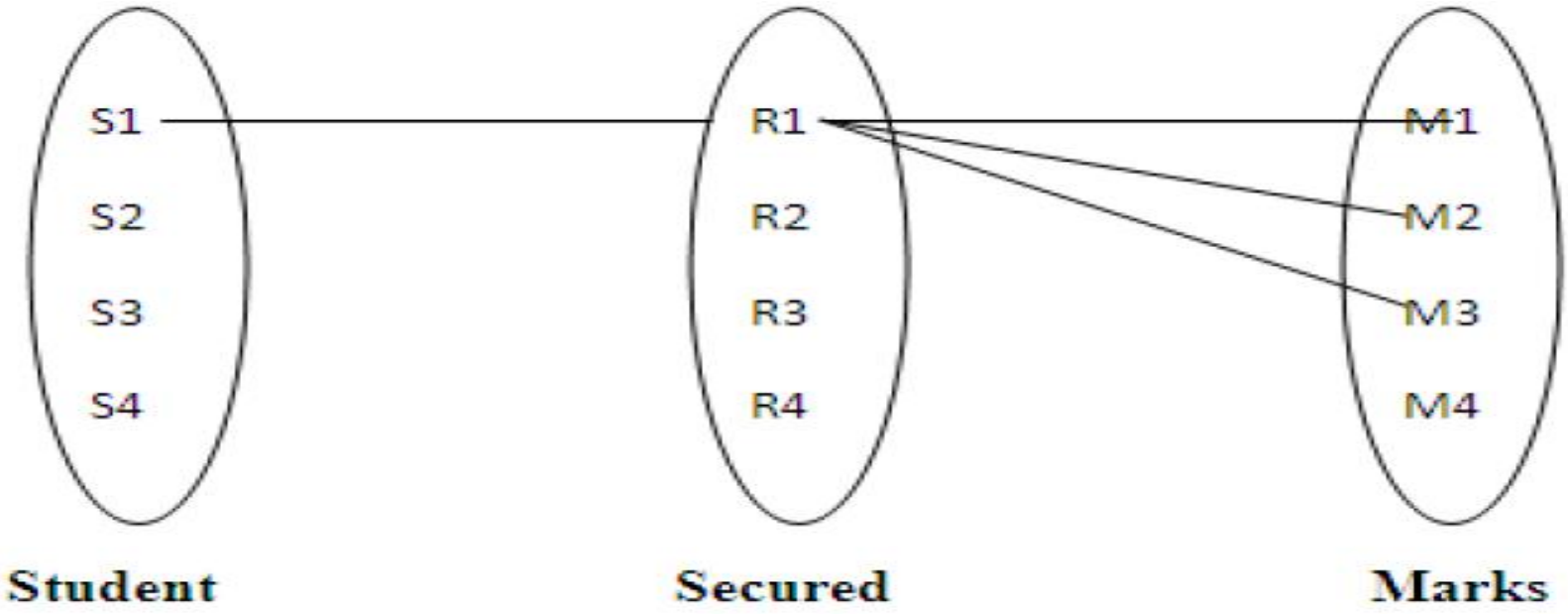
Relationship Set

- A **relationship set** is a set of relationships of the same type.



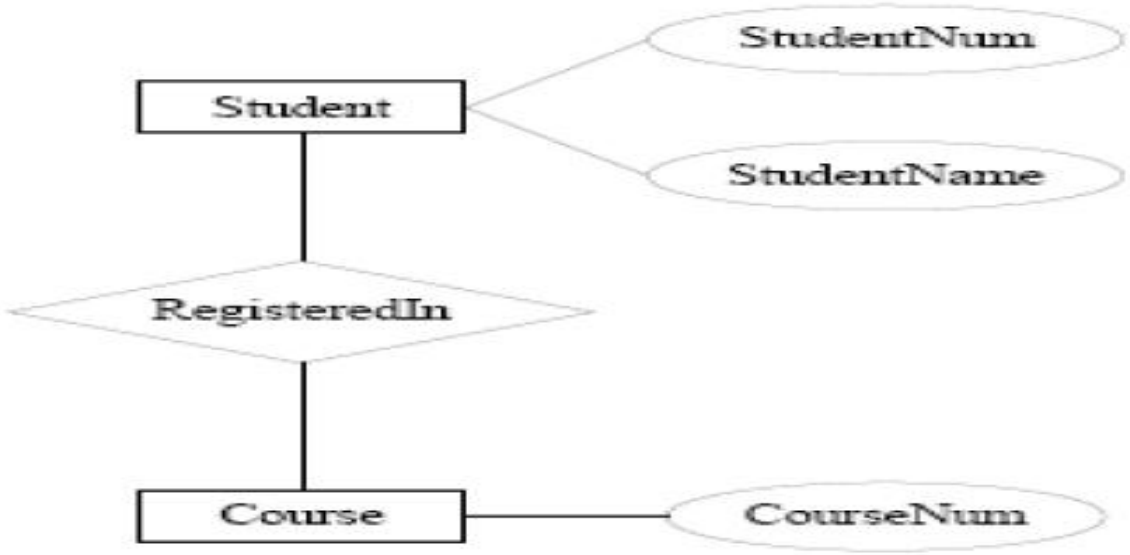
Set Representation of ER Diagram


EMPLOYEE
WORKS_FOR
DEPARTMENT
e1
e2
e3
e4
e5
e6
e7
r1
r2
r3
r4
r5
r6
r7
d1
d2
d3



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1

Graphical Representation of Relationship Sets

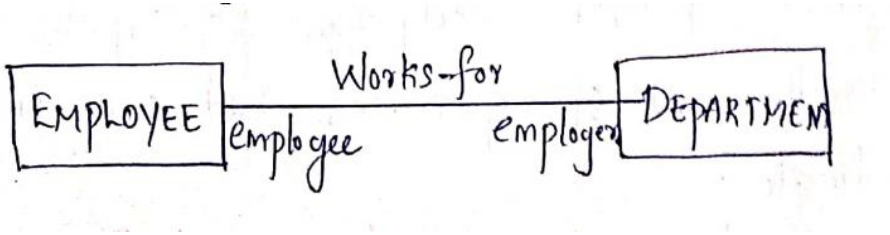


Relationship as an attribute

- A binary relationship can also be considered as an attribute
- eg, EMPLOYEE & DEPARTMENT
  - EMPLOYEE has an attribute Department where value of Department for each EMPLOYEE entity is a reference to DEPARTMENT for which EMPLOYEE works

ROLE NAMES AND RECURSSIVE RELATIONSHIPS

- Each entity type that participate in a relationship plays a particular role in relationship
- role name signifies the role that participating entity from entity type plays in each relationship instance and helps to explain what the relationship means



Recursive Relationships

- In some cases same entity type participate more than once in a relationship type in different roles
- so in this role name is essential for distingushing meaning of the role the participatig entity plays
- they are recursive relationships

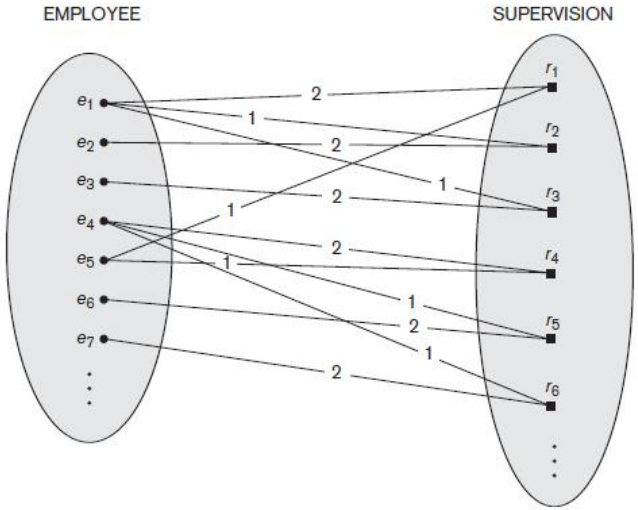


Figure 7.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

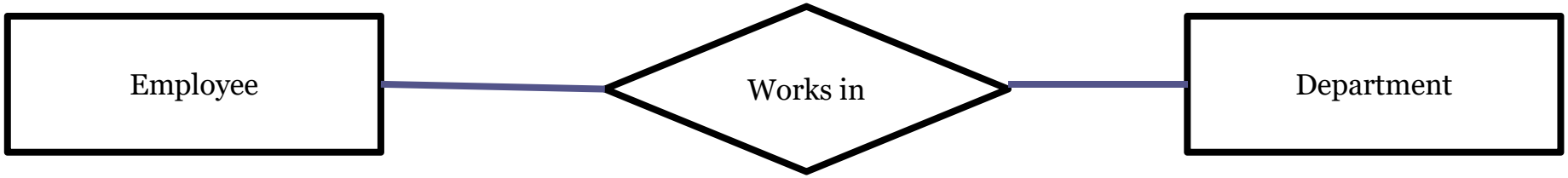
Constraints

- Relationship types usually have certain constraints. Two main types of relationship constraints:
  1. Mapping cardinalities
  2. Participation constraints

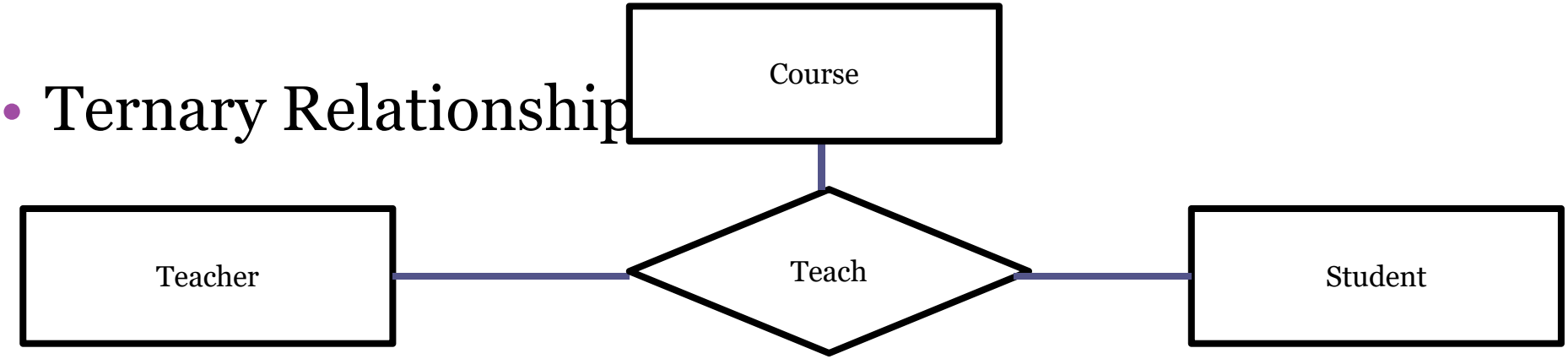
Degree of a relationship

- It is the number of entity set which are participating in a relationship
  - Unary relationship
  - Binary Relationship
  - Ternary Relationship

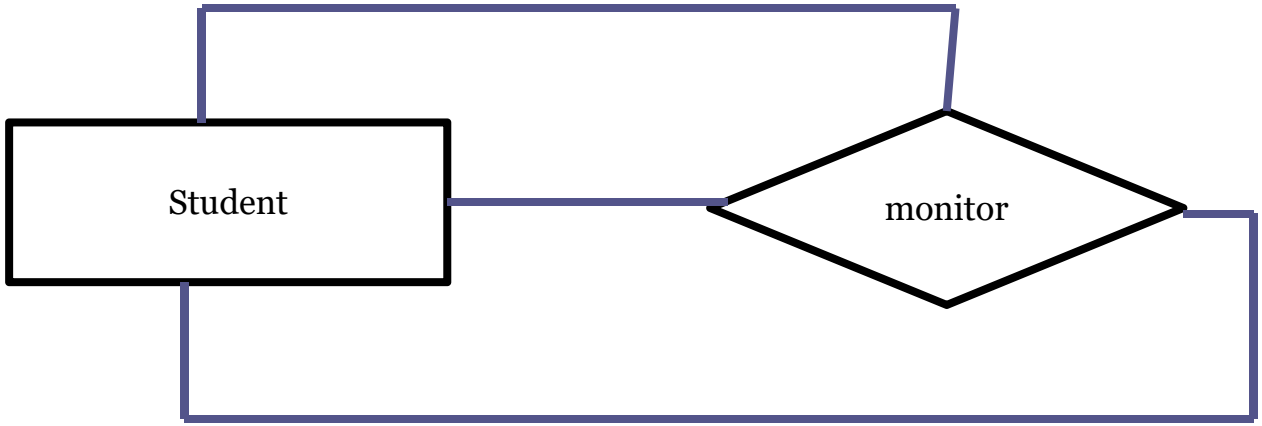
- Binary Relationship



- Ternary Relationship



- Unary Relationship



- Each relationship has
  - Name
  - Degree
  - Cardinality ratio

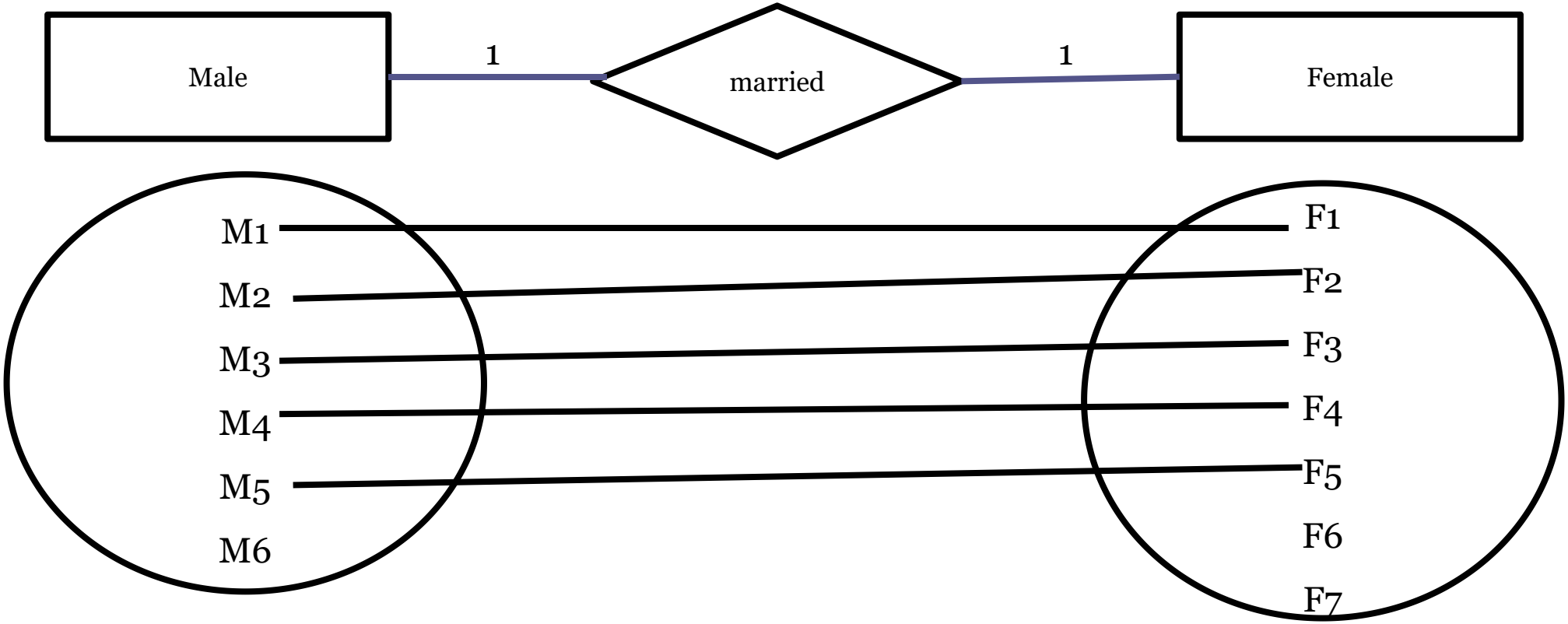
Cardinality Ratio

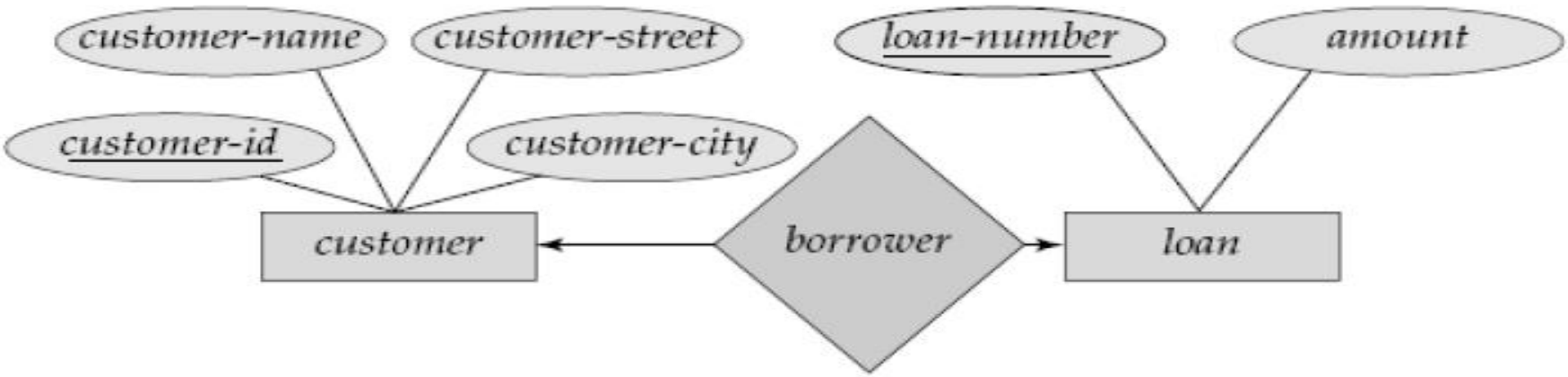
- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances to which an entity can take part in it
- It also specifies number of entities to which other entity can be related by a relationship
- Types
  - **One-to-one (1:1)**
  - **One-to-many (1: N)**
  - **Many-to-one (N: 1)**
  - **Many-to-many (M: N)**

- We express cardinality ratio by drawing
- directed line (→), signifying “one,” or an
- undirected line (—), signifying “many,”

One to One(1:1)

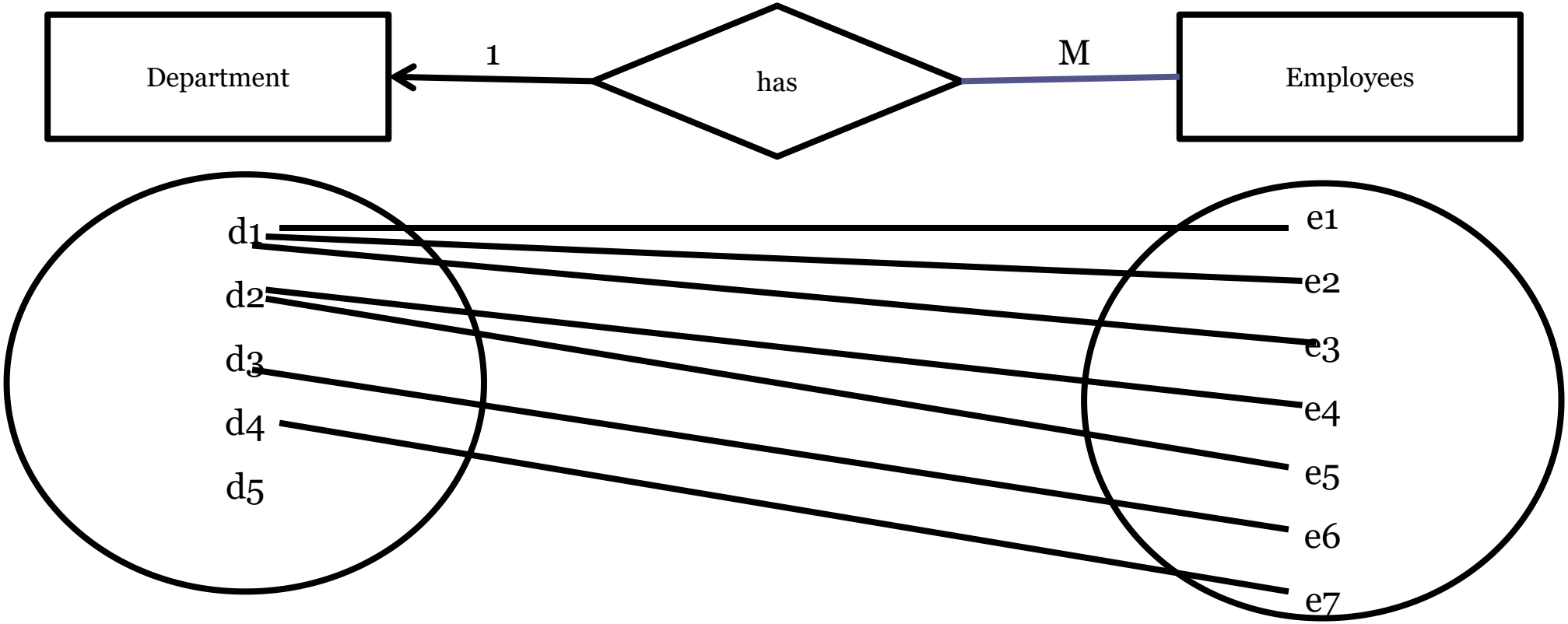
- When only a single instance of an entity is associated with single instance of other entity by a relationship
- When every entity of one entity set is related to maximum one entity of other entity set


Male
1
married
1
Female
M1
M2
M3
M4
M5
M6
F1
F2
F3
F4
F5
F6
F7

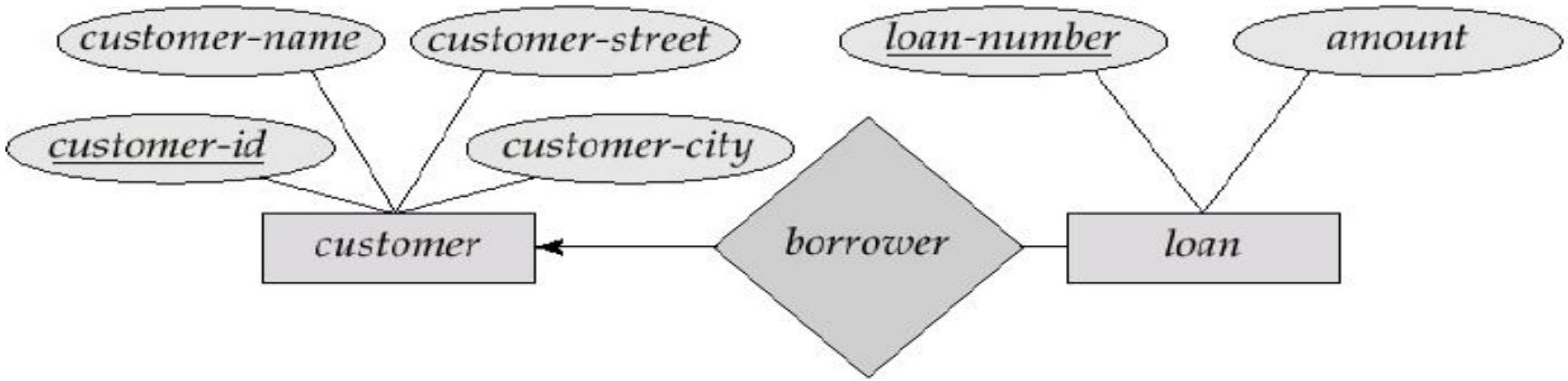

customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

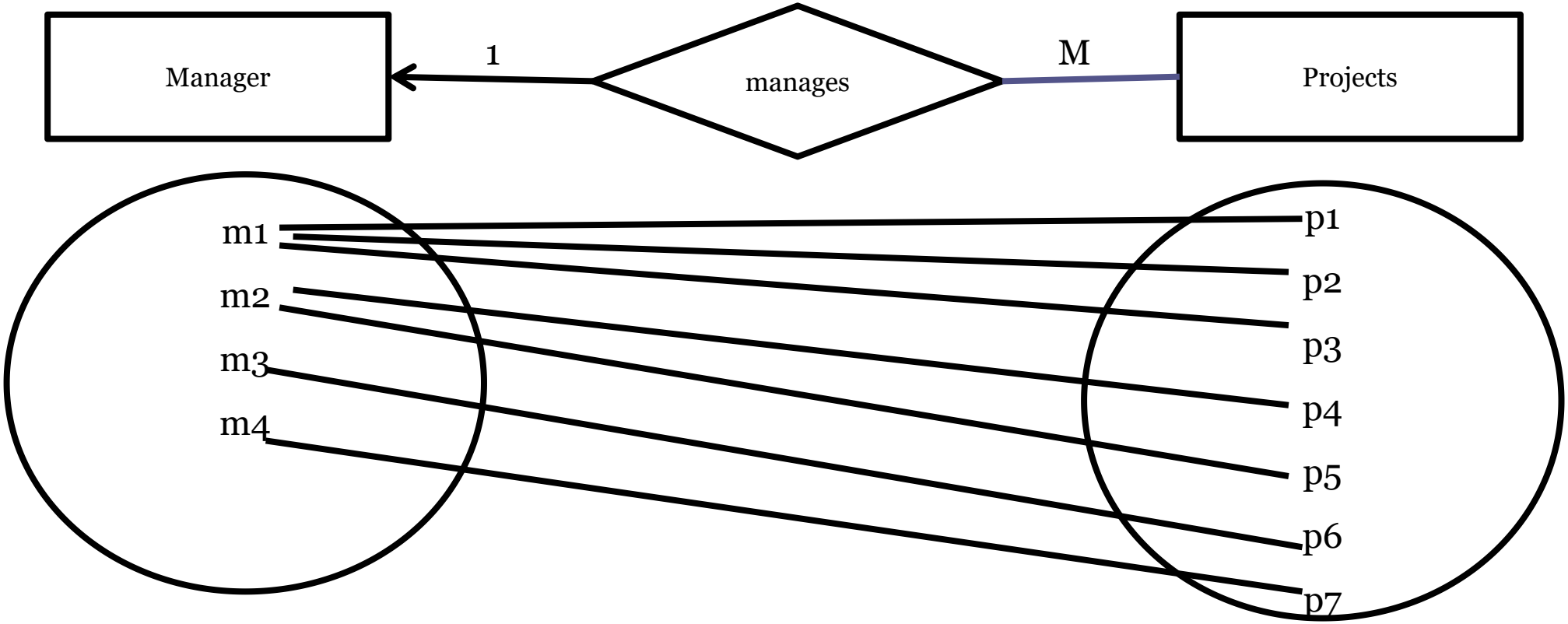
One to Many (1:M)

- When every entity of first entity set is related to at most (max) n entities of other entity set then it is one to many


Department
1
has
M
Employees
d1
d2
d3
d4
d5
e1
e2
e3
e4
e5
e6
e7

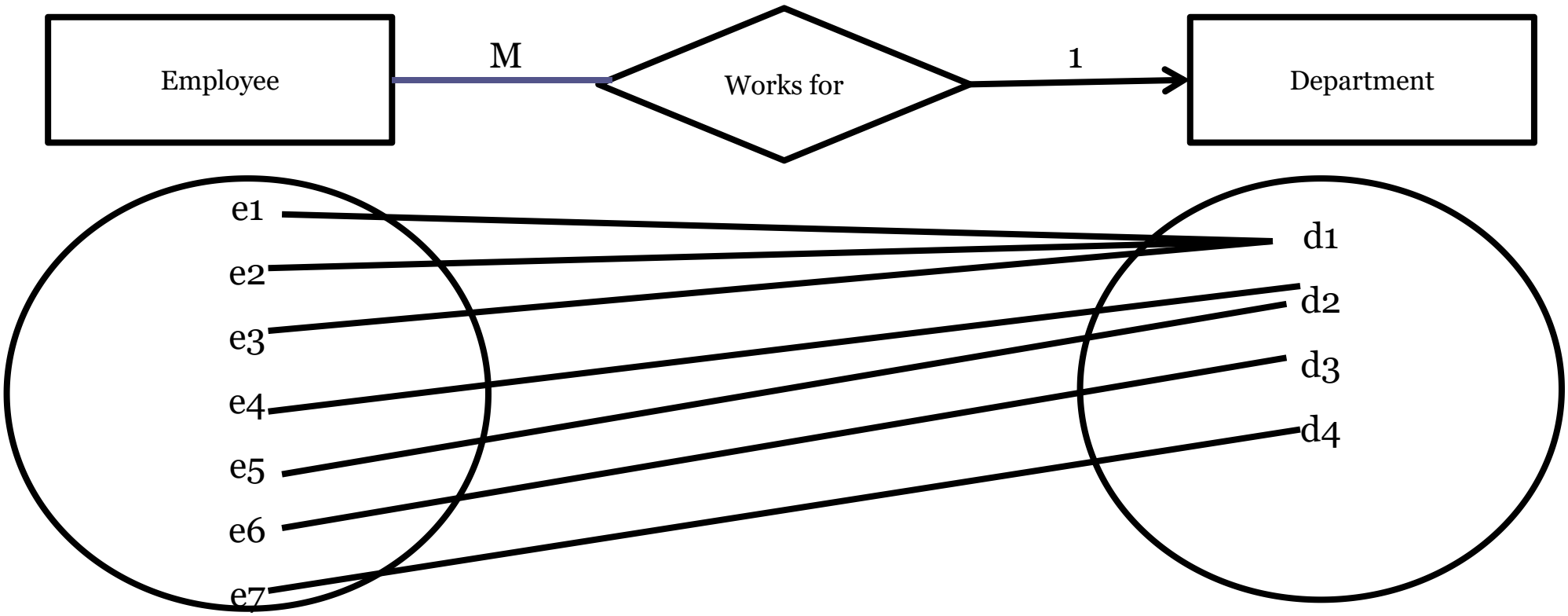
- In the one-to-many relationship a customer is associated with several loans via *borrower*



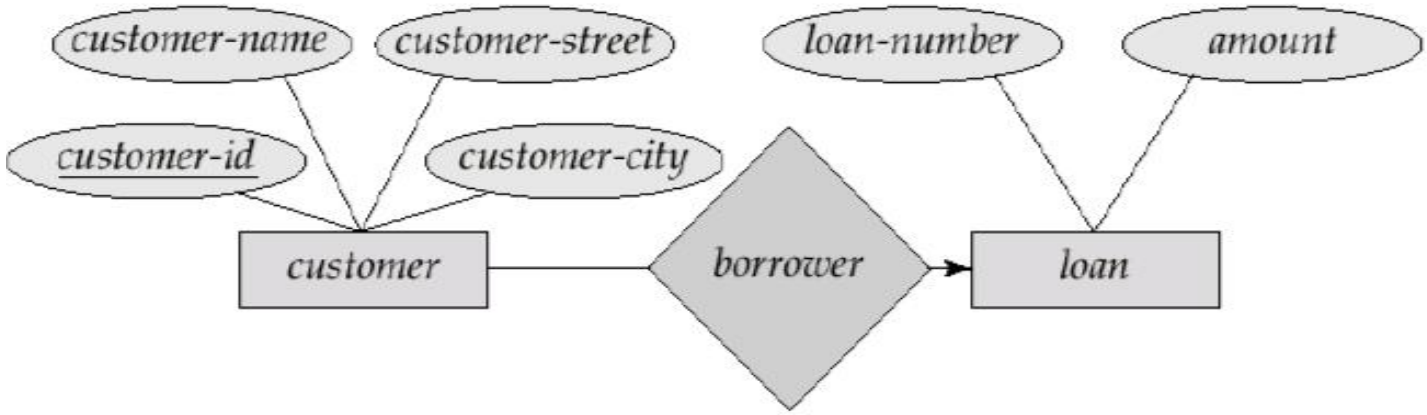

Manager
1
manages
M
Projects
m1
m2
m3
m4
p1
p2
p3
p4
p5
p6
p7

Many to One (M:1)

- When many entities of first entity set is related to 1 entity of other entity set then it is many to one

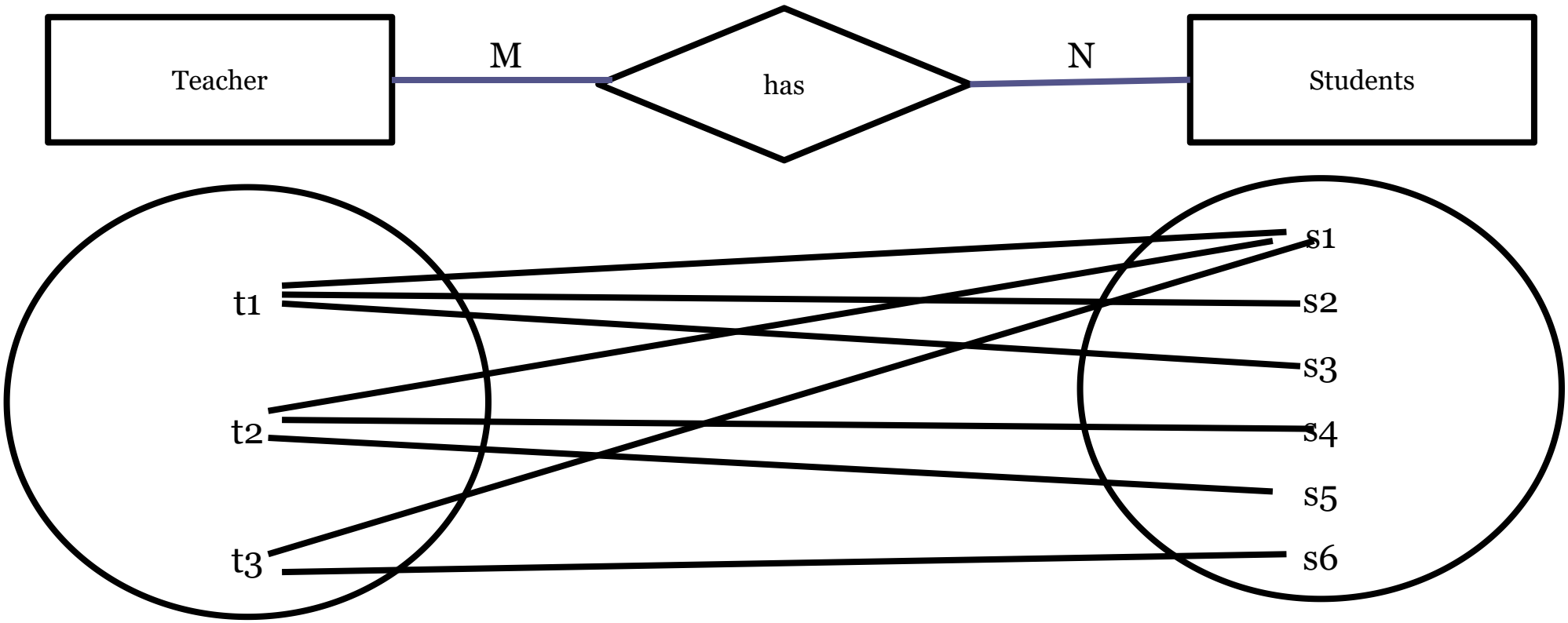

Employee
M
Works for
1
Department
e1
e2
e3
e4
e5
e6
e7
d1
d2
d3
d4

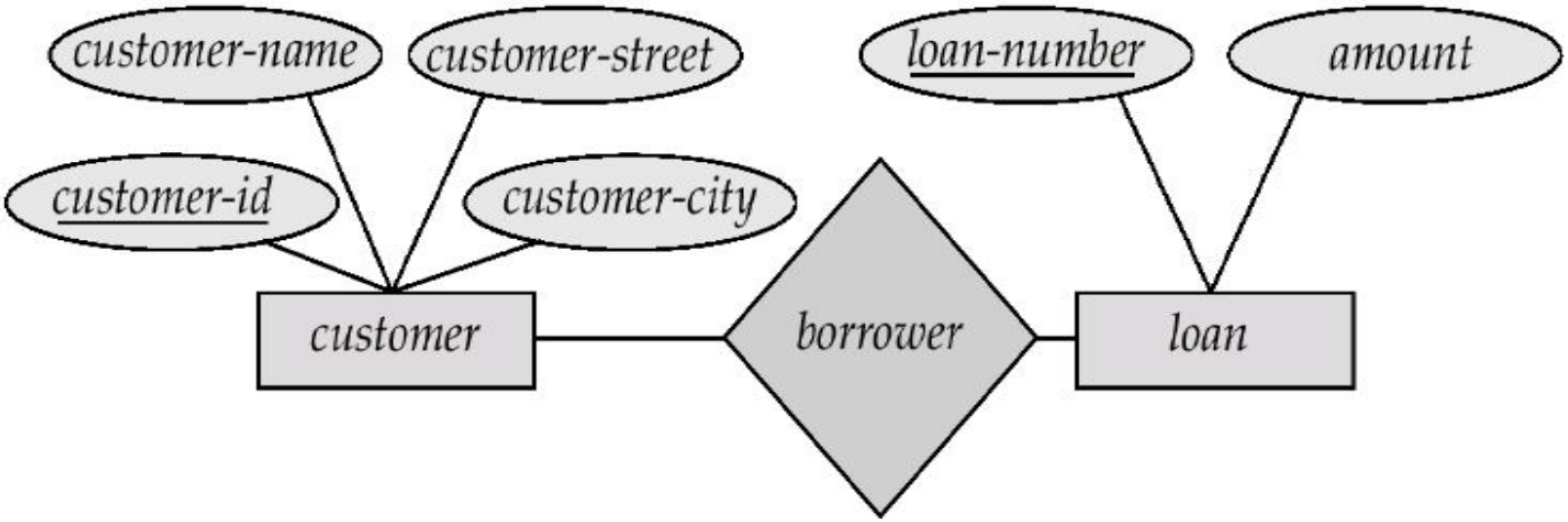
- In a many-to-one relationship a loan is associated with several customers via *borrower*.



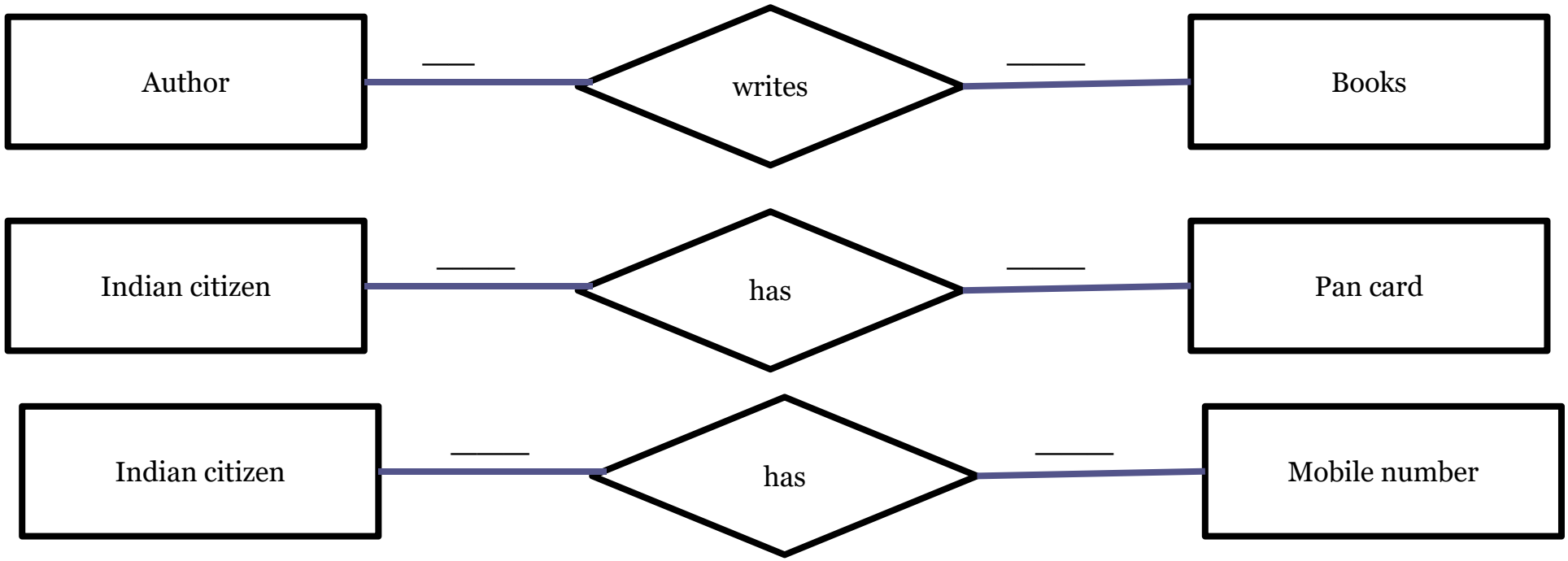
Many to Many(M:N)

- When many occurrences of one entity is related to many occurrences of another entity


Teacher
M
has
N
Students
t1
t2
t3
s1
s2
s3
s4
s5
s6


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

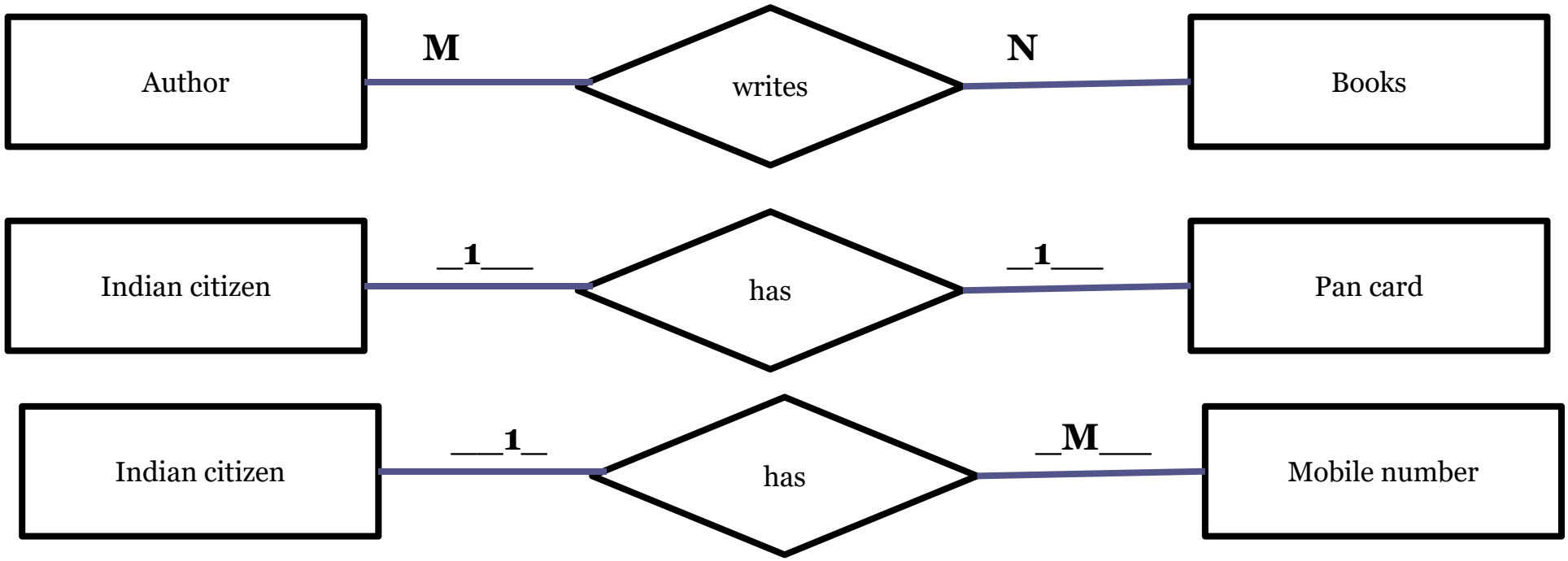
Exercise


Author
writes
Books
Indian citizen
has
Pan card
Indian citizen
has
Mobile number

Homework

- Prime minister-country
- classroom –students
- students –classroom
- customer -loan

Exercise


Author
M
writes
N
Books
Indian citizen
_1___
has
_1___
Pan card
Indian citizen
___1_
has
_M___
Mobile number

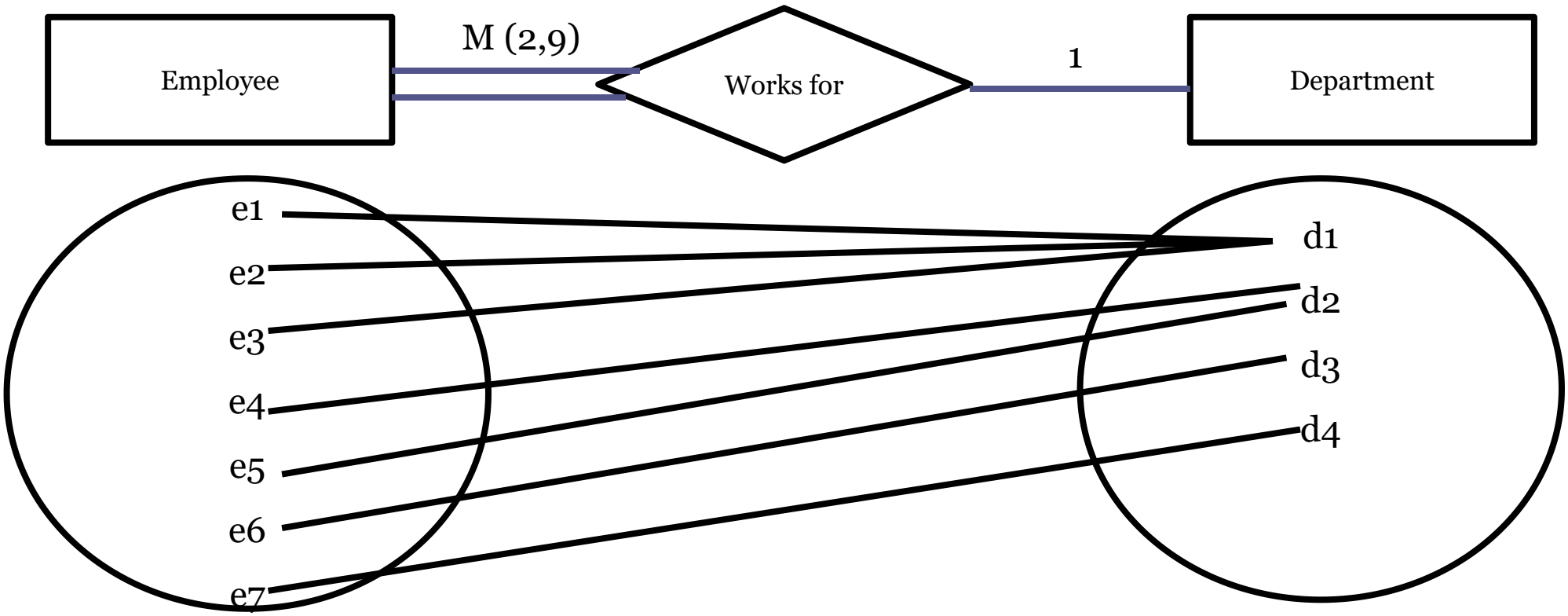
Participation constraints

- It specifies whether the existence of an entity depends on being related to another entity through relationship types
- These constraints defines max and min number of relationship instance that each entity can participate in
- Maximum cardinality
  - It defines maximum number of times an entity can participate in a relationship
- Minimum Cardinality

- There are two types of participation constraints
  1. Total Participation
  2. Partial Participation

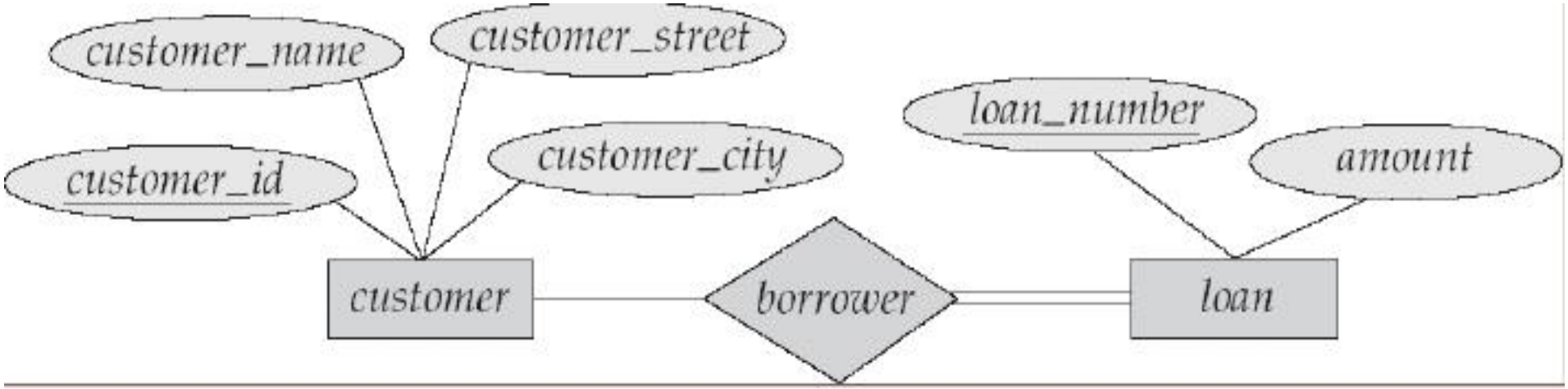
Total participation

- every entity in the entity type participates in at least one relationship in the relationship type
  - E.g. participation of loan in borrower is total every loan must have a customer associated to it via borrower
- Represented by double lines
- Minimum and maximum cardinality represented inside paranteses(m,n)
- Total participation is called existance dependency


Employee
M (2,9)
Works for
1
Department
e1
e2
e3
e4
e5
e6
e7
d1
d2
d3
d4

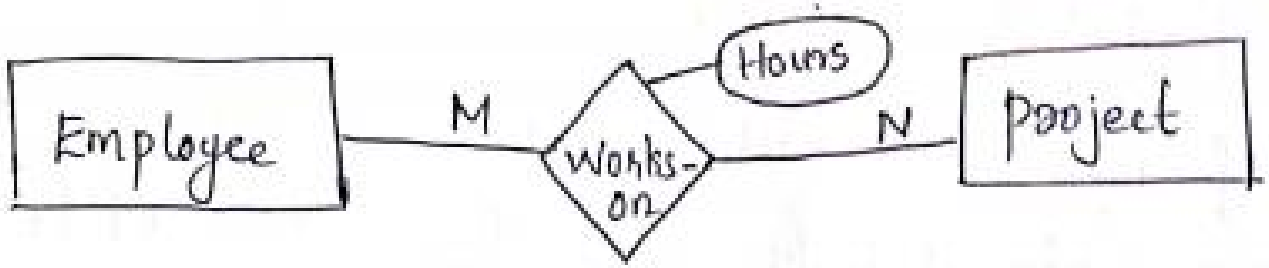
Partial participation

- Some entities may not participate in any relationship in the relationship type
  - Example: participation of customer in borrower is partial
- Represented by single line


customer_name
customer_street
customer_id
customer_city
loan_number
amount
customer
borrower
loan

Attributes of Relationship Types

- Relationship type can also have attributes similar to entity types
- Example
  - To record the number of hours/week that an employee works (le an attribu



- The attributes of 1:1 or 1:N relationship types csn be migrated to one of the participating entity types
- For M:N, some attributes may be determined by combination of participating entities in a relationship instance. Such attributed must be specified as relationship attributes

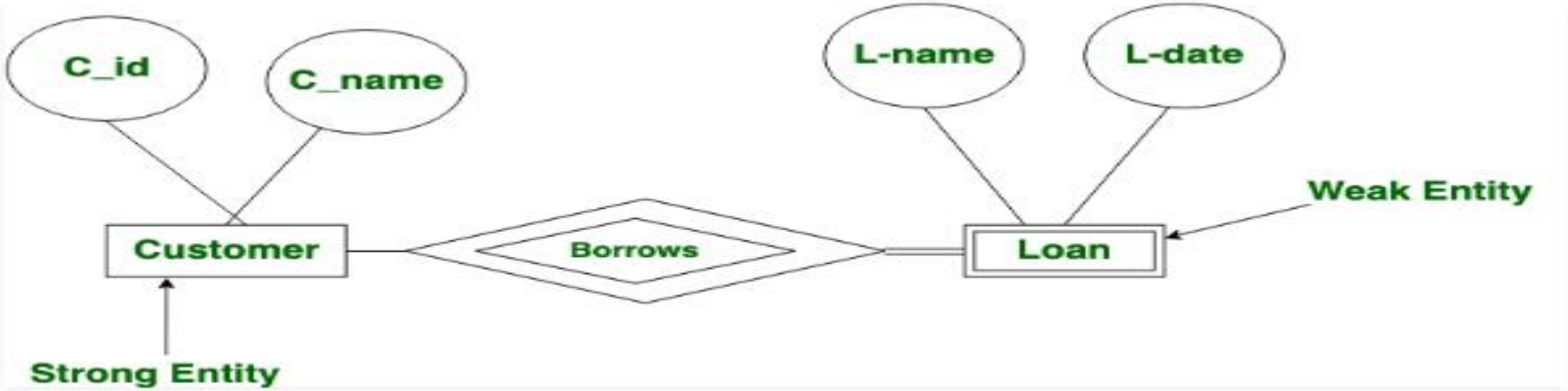
TYPES OF ENTITY TYPES

- **Strong entity type**
  - Entity types that have at least one key attribute.
  - A strong entity is not dependent of any other entity in the schema.
  - A strong entity will always have a primary key.
  - Strong entities are represented by a single rectangle.
  - The relationship of two strong entities is represented by a single diamond.
  - Various strong entities, when combined together, create a strong entity set.

- **Weak entity type**
  - Entity type that does not have any key attribute.
  - A weak entity is dependent on a strong entity to ensure the its existence.
  - Unlike a strong entity, a weak entity does not have any primary key.
  - It instead has a partial discriminator key.
  - A weak entity is represented by a double rectangle. The relation between one strong and one weak entity is represented by a **double diamond.**
-

Difference between Strong and Weak Entity:

S.NO	Strong Entity	Weak Entity
1.	Strong entity always has primary key.	While weak entity has partial discriminator key.
2.	Strong entity is not dependent of any other entity.	Weak entity is depend on strong entity.
3.	Strong entity is represented by single rectangle.	Weak entity is represented by double rectangle.
4.	Two strong entity's relationship is represented by single diamond.	While the relation between one strong and one weak entity is represented by double diamond.
5.	Strong entity have either total participation or not.	While weak entity always has total participation.

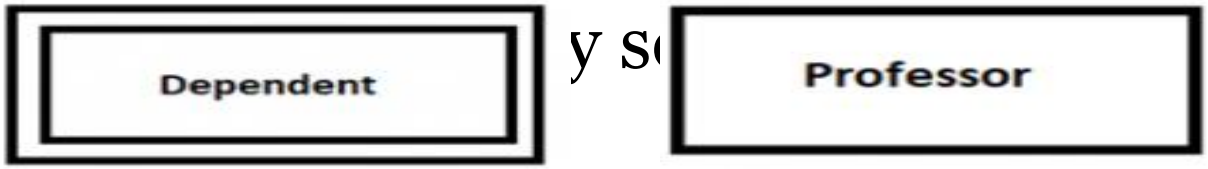

C_id
C_name
L-name
L-date
Weak Entity
Customer
Borrows
Loan
Strong Entity

- let us take an example of **Professor** entity, which is our **Strong Entity**, ith **Professor_ID** as a **Primary Key**

Professor_ID	Professor_Name	Professor_City	Professor_Salary

- The weak entity is **Professor_Dependents** entity:

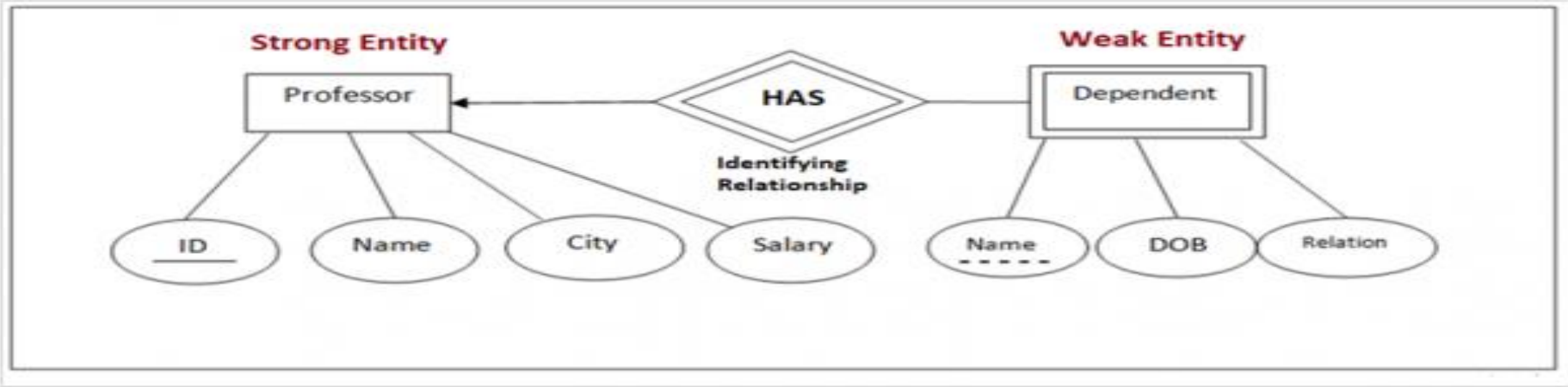
Name	DOB	Relation

- Here Dependent is a weak entity and has no primary key
- Professo 

tity

Identifying Relationship

- It links the strong and weak entity and is represented by a double diamond sign.
- Let us see with an example to link both the entities using Identifying Relationships:



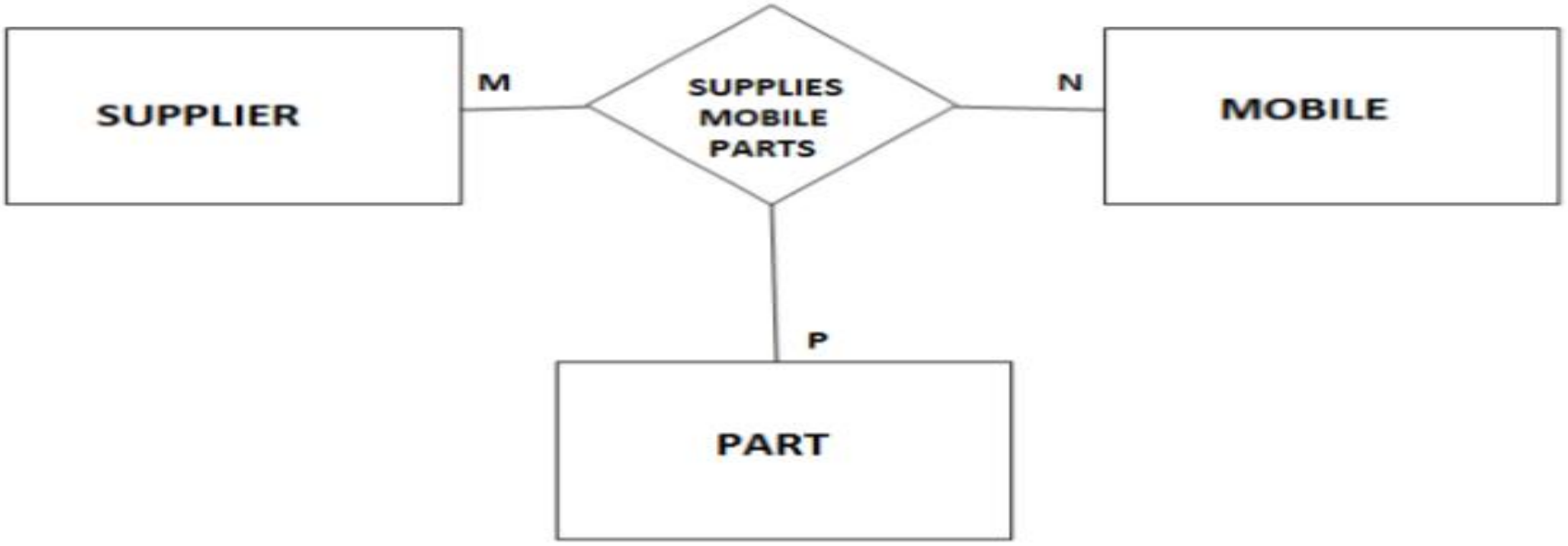
Above we saw that, Dependent Name could not exist on its own but in relationship to a Professor.

Professor	Strong Entity
Dependent	Weak Entity
Partial Key (Weak Entity)	Name
Primary Key (Strong Entity)	ID

Relationship of degree 3

- In Ternary relationship three different Entities takes part in a Relationship.
- Relationship Degree = 3
- For Example: Consider a Mobile manufacture company. Three different entities involved:
  - Mobile - Manufactured by company.
  - Part - Mobile Part which company get from Supplier.
  - Supplier - Supplier supplies Mobile parts to Company.
- Mobile, Part and Supplier will participate simultaneously in a relationship. because of this fact when we consider cardinality we need to consider it in the context of two entities simultaneously relative to third entity.

- A relationship type of degree ‘n’ will have ‘n’ edges in a ER Diagram, one connecting R to each participating entity type


SUPPLIER
M
SUPPLIES
MOBILE
PARTS
N
MOBILE
P
PART

Cardinality in Ternary Relationship

- Say for a given instance of Supplier and an Instance of Part, can that supplier supply that particular part for multiple Mobile models.

  Example – Consider a Supplier S1 that supplies a Processor P1 to the company and the uses the Processor P1 supplied by Supplier S1 in its multiple Models in that case the cardinality of Mobile relative to Supplier and Part is N (many).

- In case of Supplier's cardinality we can say for a given instance of Mobile one of its Part can be supplied by multiple Suppliers.

  Example – Consider a Mobile M1 that has a Part P1 and it is being supplied by multiple Suppliers in that case the cardinality of Supplier relative to Mobile and Part is M (many).

- Similarly, for a given instance of Supplier and an instance for Mobile does the Supplier supply multiple Parts.

  Example – Consider a Supplier S1 supplying parts for Mobile M1 like screen, Processor etc. in that case the cardinality of Part relative to Supplier and Mobile is P (many).

- Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents

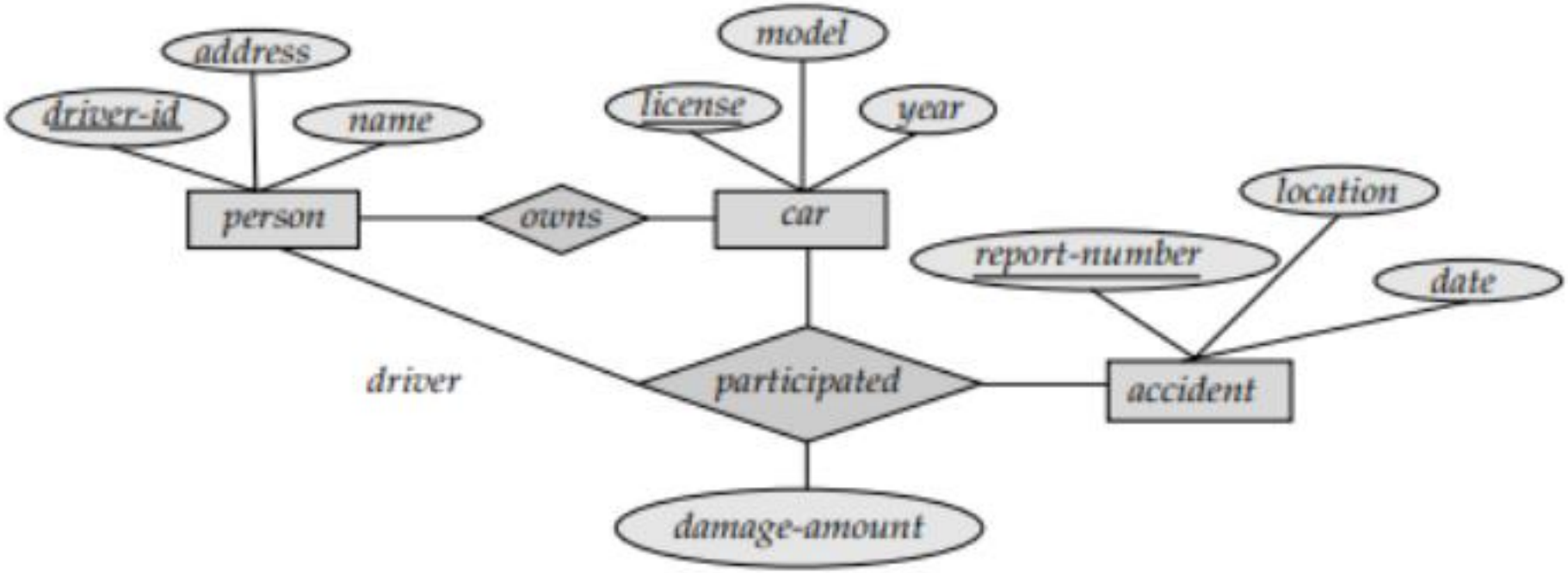


Figure 2.1 E-R diagram for a Car-insurance company.

- Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted

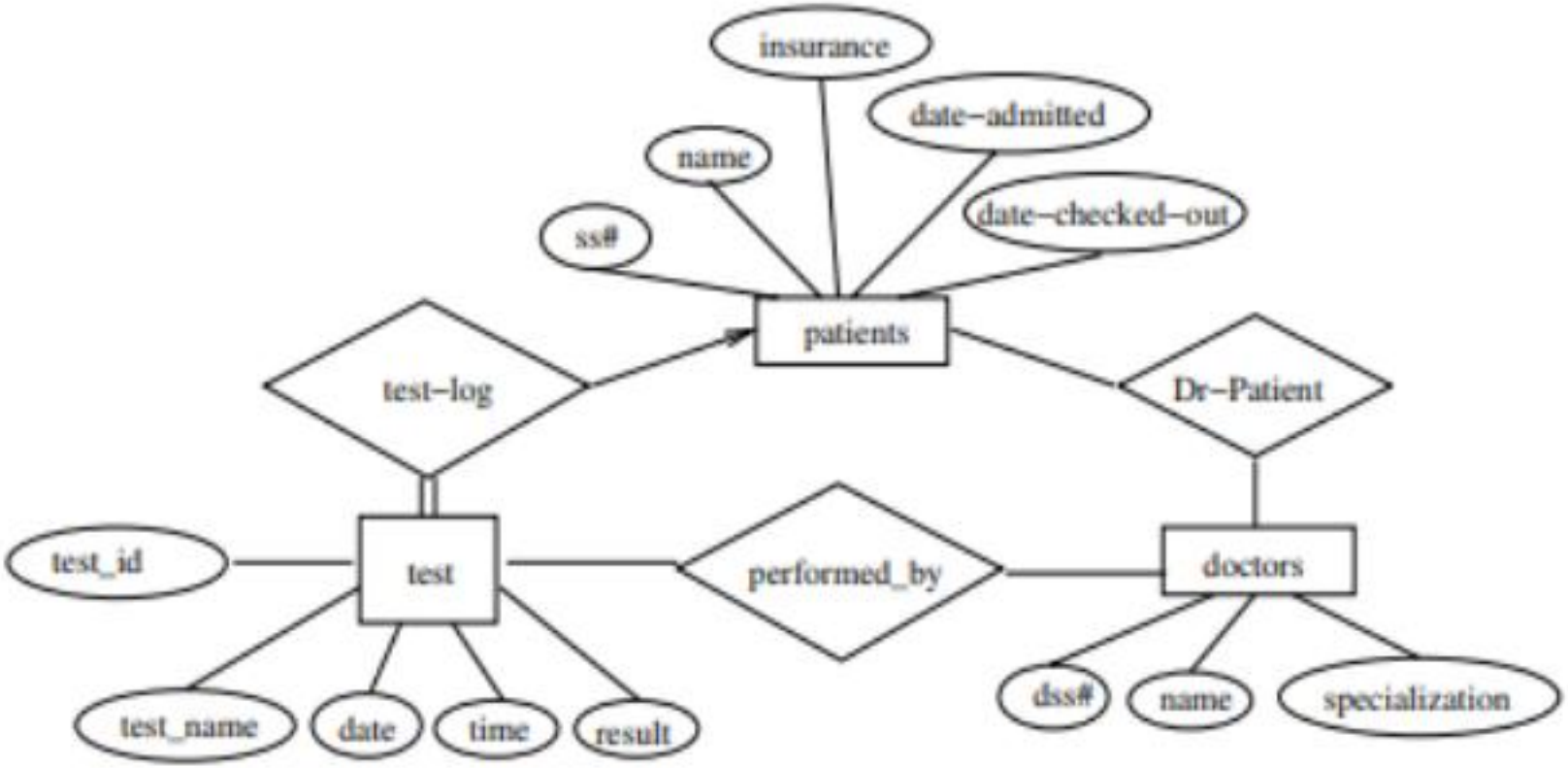


Figure 2.2 E-R diagram for a hospital.

A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram forthe registrar's office. Document all assumptions that you make about the mapping constraints.

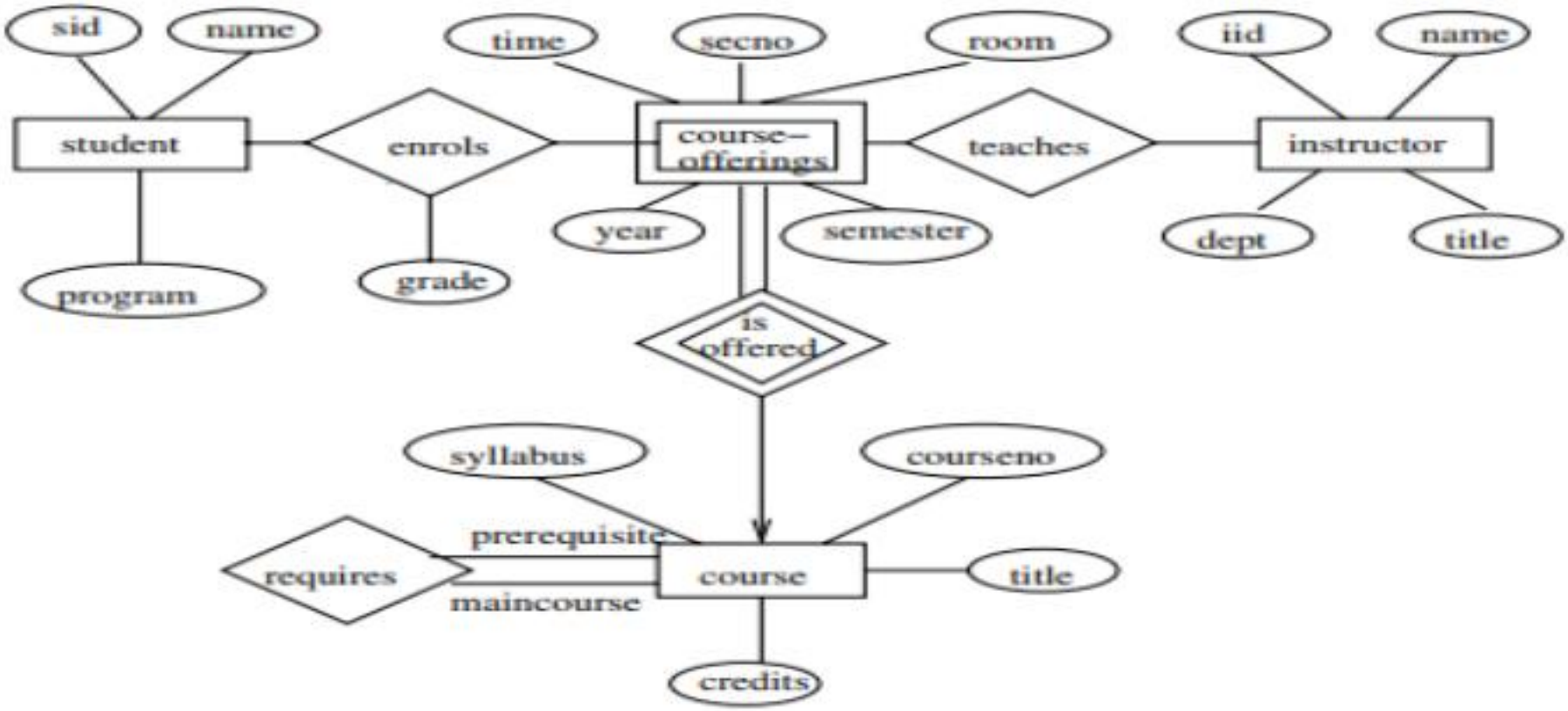


Figure 2.3 E-R diagram for a university.

- Consider a database used to record the marks that students get in different exams of different course offerings.
- Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

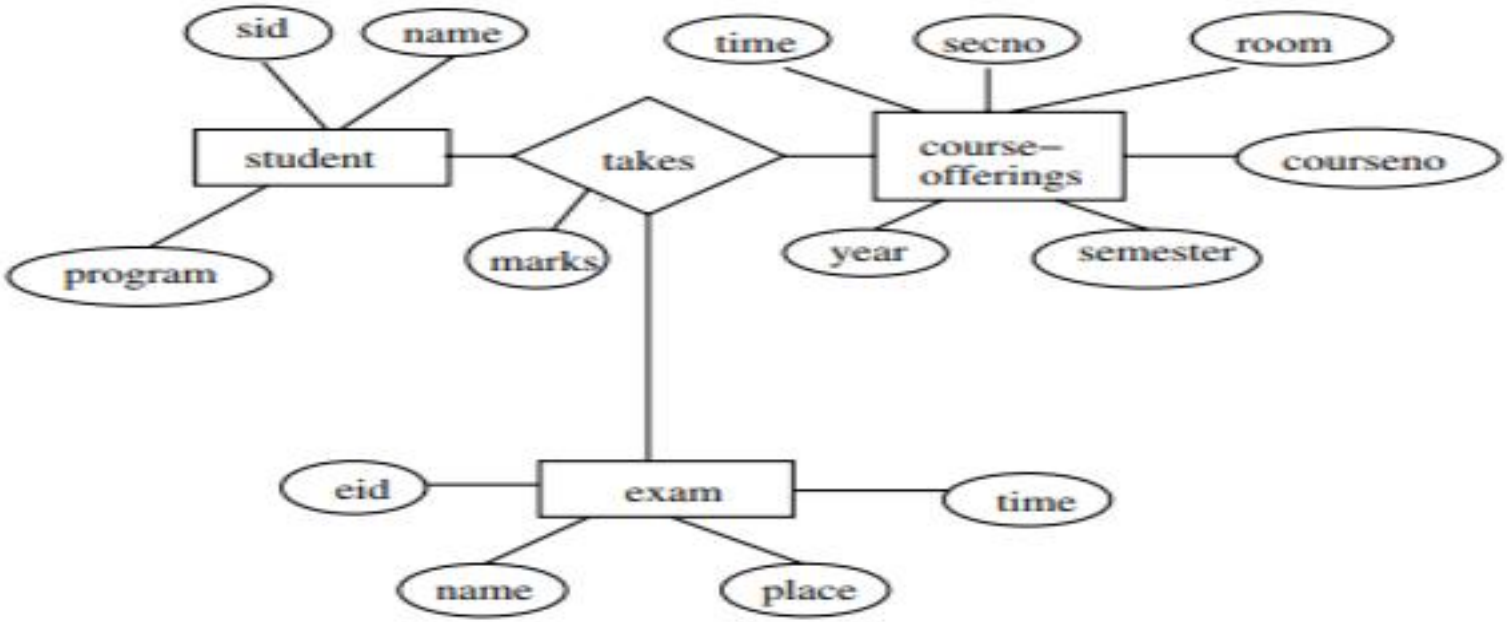
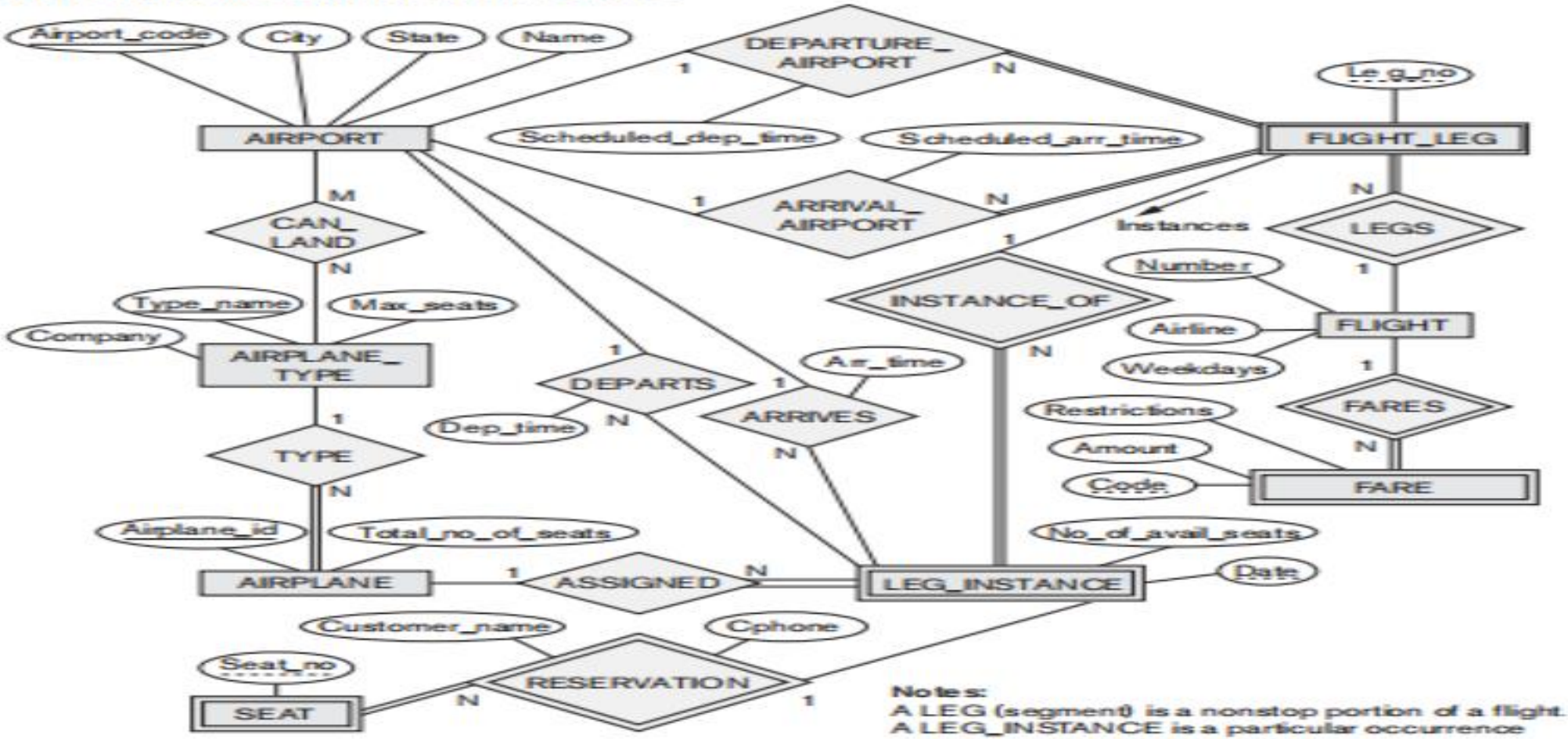


Figure 2.4 E-R diagram for marks database.

which shows a simplified schema for an airline reservations system. Extract from the ER diagram the requirements and constraints that produced this schema. Try to be as precise as possible in your requirements and constraints specification.

Figure 7.20
An ER diagram for an AIRLINE database schema.



Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.