

Chapter 1: Introduction

Basic Terms



- Data
- Information
- Database
- dbms

Data VS Information



- ❑ Data - Unprocessed information
e.g **stud name** or **mark**
- ❑ Information - processed data
- ❑ Collection of data which gives some meaning i.e. organized data
e.g **25 + 50 = 75**
- ❑ Data is converted to information

Database



- Database contains information about a particular enterprise (inter-related data).
 - A database is a collection of information or data organized and presented to serve a specific purpose
- e.g. university database contain information about
- Entities such as students, faculty, courses and classrooms
 - Relationships between entities such as student's enrollment in courses,

DBMS



- ❑ Complex software that controls the organization, storage and retrieval of data in a database.
- ❑ collection of computer programs that allow storage, modification and extraction of information from a database
- ❑ *Database+DBMS s/w=DBMS*
- ❑ *DBMS is an intermediate layer b/w program and data*
- ❑ Examples: -
 - Oracle, DB2, Microsoft Access, Microsoft SQL Server, PostgreSQL, MySQL, etc.

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
 - Collection of interrelated data
 - Set of programs to access the data
 - An environment that is both *convenient* and *efficient* to use
 - a software package designed to store and manage databases
- Database Applications:
 - Banking: transactions
 - Universities: registration, grades
 - Sales: customers, products, purchases
 - Online retailers: order tracking, customized recommendations
- Databases can be very large.
- Databases touch all aspects of our lives

University Database Example

- Application program examples
 - Add new students, instructors, and courses
 - Register students for courses, and generate class rosters
 - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems
- A File Management system is a DBMS that allows access to single files or tables at a time. In a File System, data is directly stored in set of files. It contains flat files that have no relation to other files (when only one table is stored in single file, then this file is known as flat file).

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
 - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
 - Need to write a new program to carry out each new task
- Data isolation
 - Multiple files and formats
- Integrity problems
 - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
 - Concurrent access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
 - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

What is a database system?

Computerized record keeping system

Single user / Multi user system

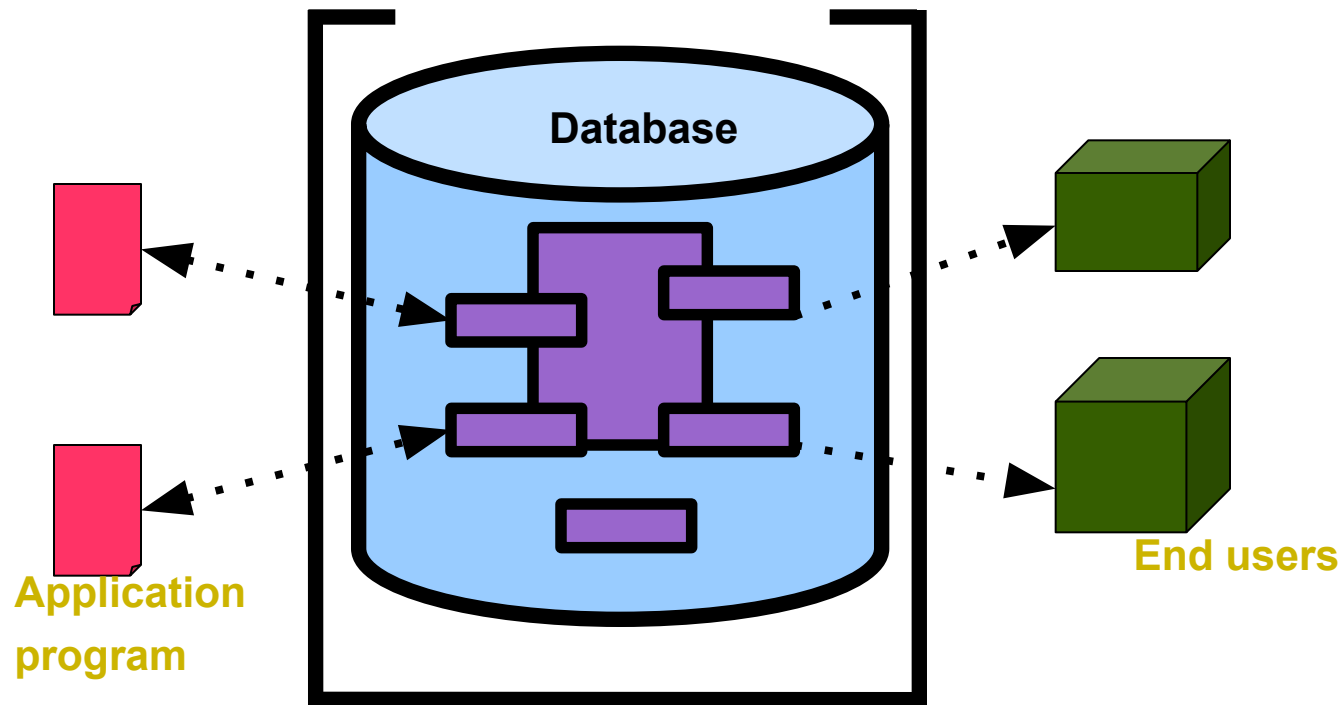
Four major components

Data

Hardware

Software

Users



Types of Databases

Single user:

✓ *Supports only one user at a time*

Desktop:

✓ *Single -user database running only one desktop.*

Multi user:

✓ *Supports multiple users at a time.*

Workgroup:

✓ *Multi-user database that supports a small group of users or a single department*

Enterprise:

✓ *Multi-user database that supports a large group of users or an entire organization*

Types of Databases



- Can be classified by location:

- **Centralized**

- ✓ *Supports data located at a single site*

- **Distributed**

- ✓ *Supports data distributed across several sites*

Types of Databases

Can be classified by use:

▣ Transactional (or production):

- ✓ *Supports a company's day-to-day operations*

▣ Data warehouse:

- ✓ *Stores data used to generate information required to make tactical or strategic decisions*

- ✓ *Often used to store historical data*

- ✓ *Structure is quite different*

Advantages of DBMS



- **Data independence**: Application programs should be as independent as possible from details of data representation and storage.
- The DBMS can provide an abstract view of the data to insulate application code from such details.
- **Efficient data access**: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.



- **Data integrity and security**: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data.
- For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce *access controls that govern* what data is visible to different classes of users.



- **Data administration:** When several users share the data, centralizing the administration of data can offer significant improvements.
- Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.

- **Concurrent access and crash recovery:** A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced application development time:** Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This in conjunction with the high-level interface to the data, facilitates quick development of applications.

Describing and storing data in DBMS



Data Models

- A collection of tools for describing
 - Data
 - Data relationships
 - Data semantics
 - Data constraints
- A way to describe the design of a database at the physical, logical and view levels.
- A data model is a collection of high level data description constructs that hide many low level storage details.
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
 - Network model
 - Hierarchical model

The relational model



- **Relation-set of records.**
- A description of data in terms of a data model is called a **schema**.
- In the relational model, the schema for a relation specifies its name, the name of each field (or attribute or column), and the type of each field.
- Eg: Student relation with five fields and corresponding data type

Students(sid: string, name: string, login: string, age: integer, gpa: real)

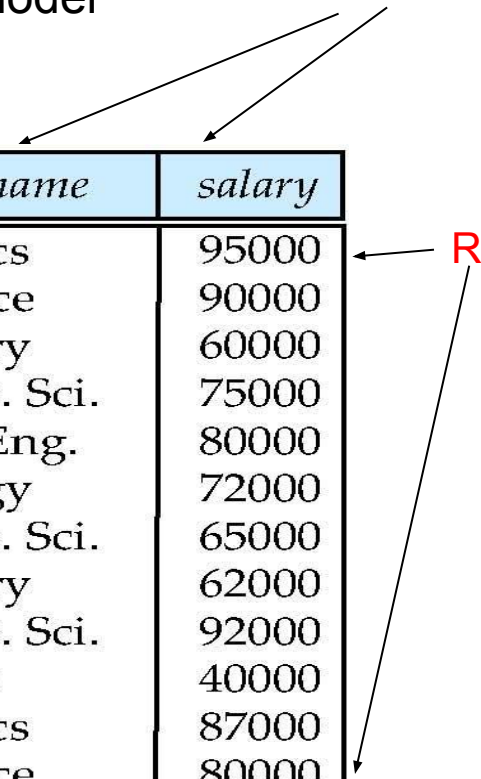


- In this model, the data is organized into tables (ie., rows and columns). These tables are called relations.
- A row in a table represents a relationship among a set of values.
- Rows of relations are generally referred to as *tuples* and the columns are usually referred to as *attributes or fields*.
- No two tuples are identical and their ordering is not important.
- A relationship is represented by combining the primary key of the relations.
- various operations can be performed on relations such as insert new tuples, delete tuples and modify tuples.
- There are several languages for expressing these operations. One such language is *relational query language*.

Relational Model

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

Columns / attributes / fields



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

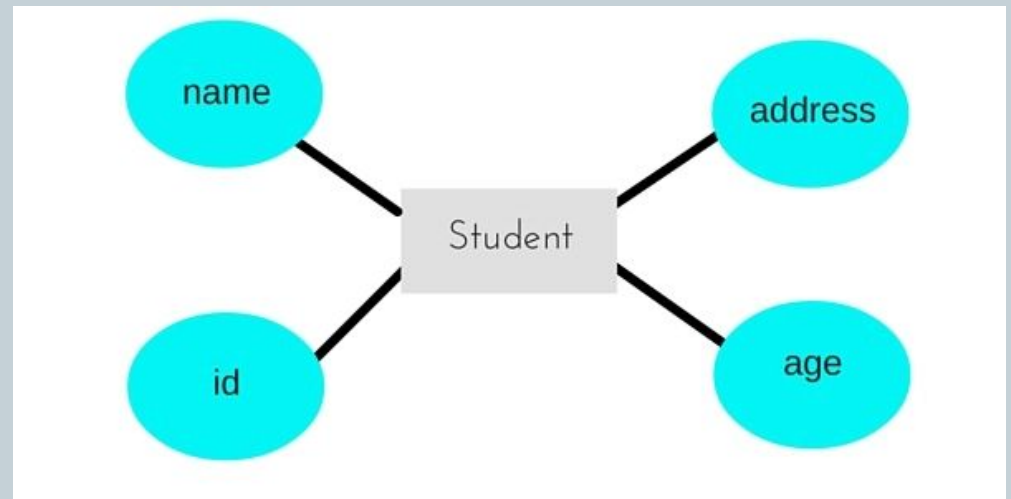
Rows / tuples

(a) The *instructor* table

Entity-relationship Model



- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.
- Different entities are related using relationships.
- This model is good to design a database, which can then be turned into tables in relational model
- Eg:
- **Entity**- Student
- **Attributes** – id,ame,age,address



Object-Based Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
 - Extend the relational data model by including object orientation and constructs to deal with added data types including encapsulation, methods or functions and object identity.
 - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Object-relational data model combines features of object-oriented data model and object-relational data model.

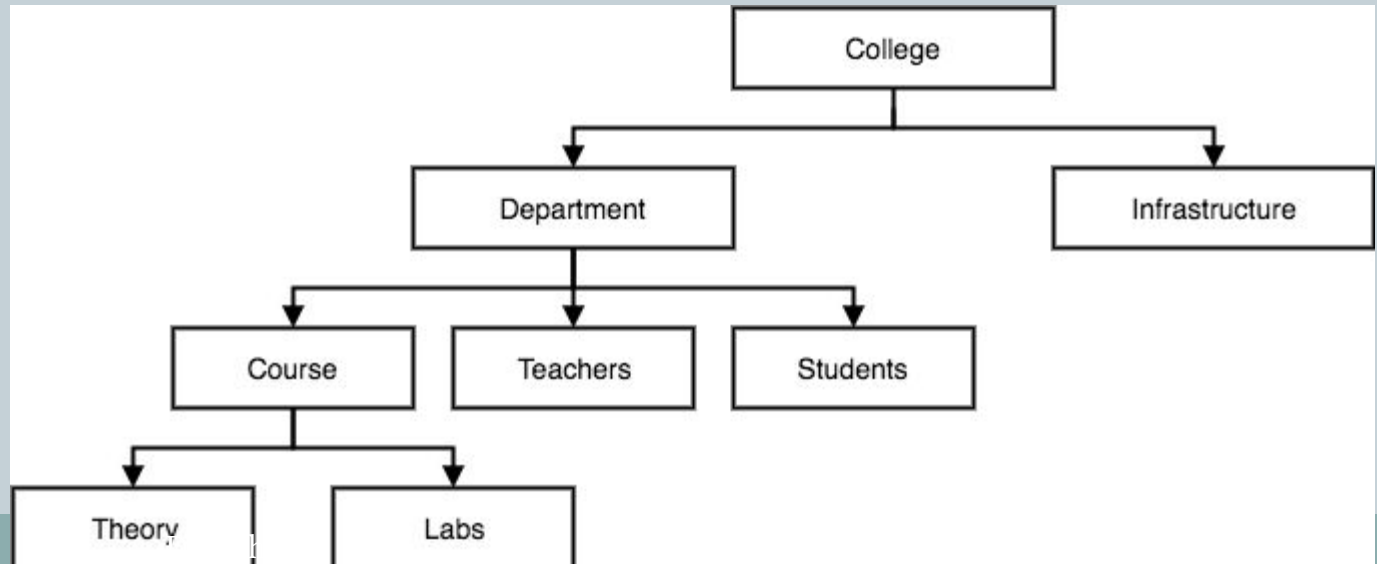
Semistructured Data Model

- Permits specification of data where individual items of the same type may have different sets of attributes.
- **XML: Extensible Markup Language is widely used to represent semistructured data.**

Hierarchical Model



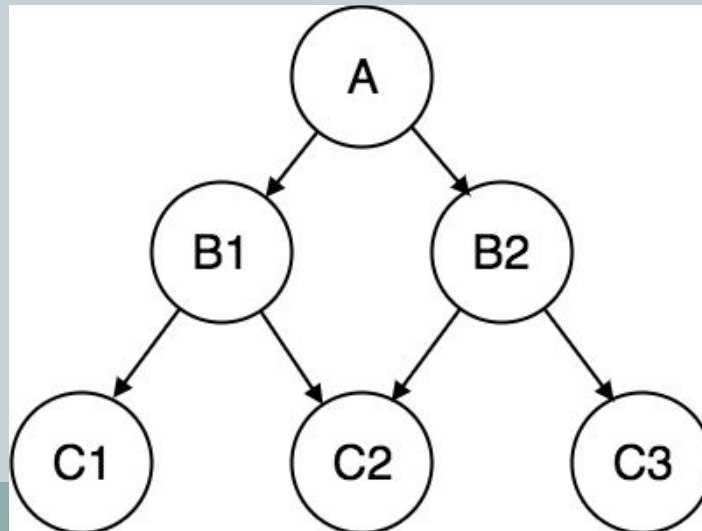
- In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.
- In this model, a child node will only have a single parent node.
- This model efficiently describes many real-world relationships like index of a book, recipes etc.



Network Model



- This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.
- In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.
- This was the most widely used database model, before Relational Model was introduced.



Instances and Schemas



□ Schema

- Overall design of the data base
- The logical structure of the database.

✓ **Instance** – The actual content of the database at a particular point of time.

✓ collection of information stored in the database at a particular moment.

Data Abstraction

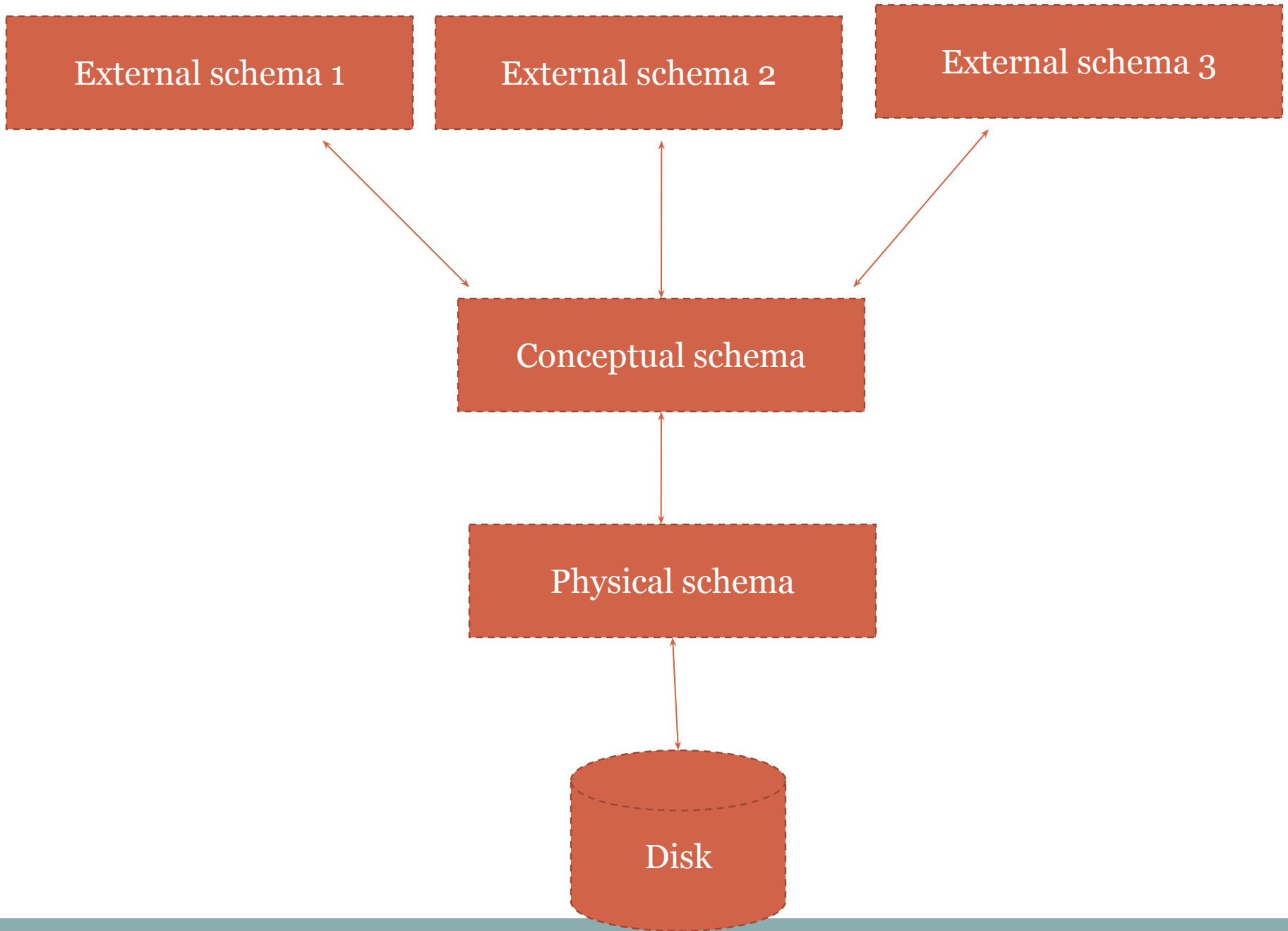


- ❑ To provide users with an **abstract view of the data**
i.e. the system hides certain details of how the data are stored and maintained. This is the major purpose of DBMS.
- Since many database users are not computer trained, the Complexity is hidden from users through several levels of abstraction.
- To simplify users interaction with the system.

Different levels of Abstraction



- Physical level / Internal level
- Conceptual level / logical level
- View level / External level





- Depending on different levels of abstraction, DBMS has several schemas
 - Lowest level – Physical schema
 - Intermediate level – Logical schema
 - Highest level – External schema

Physical level



- Lowest level of abstraction
- Describes how the data are stored
- Gives the description of the physical storage structure of the database
- How the data is actually stored on the secondary storage devices like tapes and disks.
- What file organization to use to store the relations and create auxiliary data structures called indexes to speed up data retrieval operations

Conceptual level



- Hides the details of physical storage
- Describes what data are stored
- Describes the relationship among data
- Describes entities, data types, relationships, constraints
- Describes the structure of the entire database
- Changes made in this level does not affect the external level.
- **Conceptual schema:**
 - Students(sid: string, name: string, login: string, age: integer, gpa:real)
 - Courses(cid: string, cname:string, credits:integer)
 - Enrolled(sid:string, cid:string, grade:string)

View level



- Highest level of abstraction
- Describes part of the database for a particular group of users
- Can be many different views of a database
e.g. Tellers in bank get a view of customer accounts but not of payroll data

Data Independence



- The ability to modify a schema definition in one level without affecting a schema definition in the next higher level.
- Two types of data independence
- Physical data independence
- Logical data independence



- **Physical Data independence**

- The ability to modify the physical schema without changing the conceptual schema
- e.g. DBA decides that it is more efficient to store the data in a B+ tree instead of a heap, so he changes the physical schema

- **Logical data independence**

- The ability to modify the logical schema without causing any changes to the external schema
- e.g. columns could be added to tables without changing the existing external schema

Database languages



- DDL to specify database schema(both physical and conceptual schema)
- DML to express database queries and updates(to create, modify or retrieve data from database)
- DDL& DML form parts of a single database language, such as SQL language

Data Definition Language (DDL)



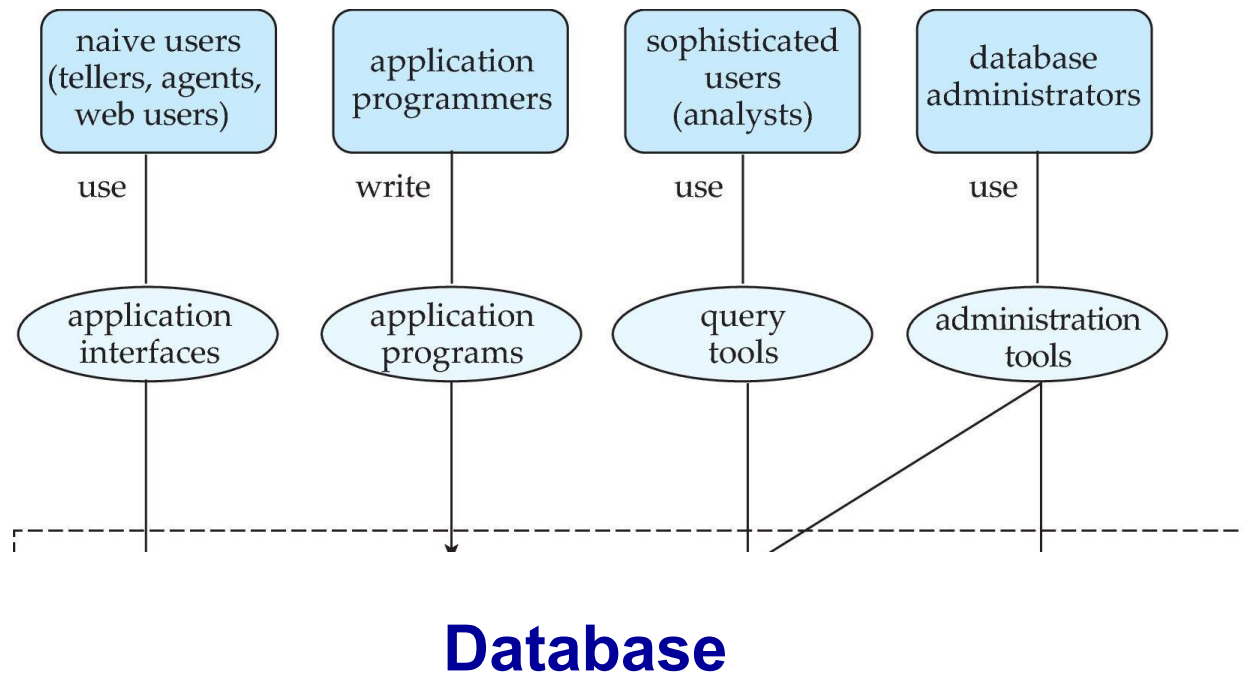
- Specification notation for defining the database schema.
Eg: - create table account(account-number char(10),balance integer)
- Execution of DDL statement creates that table also it updates a special set of tables called **data dictionary**
- Data dictionary contains metadata (ie. data about data)
 - ✓ *Eg: - Database schema*
 - ✓ *Database system consult the data dictionary before reading or modifying actual data.*

Data Manipulation Language(DML)



- Language that enables users to access or manipulate data.
 - Retrieval of information stored in database
 - Insertion of new information stored in database
 - Deletion of information from the database
 - Modification of information stored in database
- DML also known as query language
- Basically there are two types:
 - ✓ **Procedural DML** – *require a user to specify what data are needed and how to get those data*
 - ✓ **Declarative DML (Non procedural DML)** – *require a user to specify what data are needed without specifying how to get that data. Eg:- SQL*
 - ✓ **SQL is the most widely used query language**

Database Users and Administrators



People who work with a database can be categorized as database users and database administrators

Types of database users

- **Naive users**

unsophisticated users interacting with system by invoking one of the application programs that 've been written previously.

e.g bank teller transfers \$50 from account A to B by invoking a program called 'transfer'

- **Application programmers**

computer professionals who write application programs like 'transfer' in DML



- **Sophisticated users**

interact without writing programs. They form their requests in query language.

- **Specialized users**

sophisticated users who write specialized database applications.

E.g CAD systems and Expert systems

Database Administrator

A person who has a central control over the system is called a DBA

Functions of DBA

Design of the Conceptual and Physiscal schema (Schema Definition)

- DBA creates the original database schema by executing data definition statements in DDL.
- Based on what data users needs and how it is likely to be used, DBA must design both physical and conceptual schema.

● Security and authorization (Granting of authorization for data access)

DBA grants different types of authorization by which DBA regulates which parts of the database various users can access

● Data availability and recovery from failures

- After crash or failure, database must be restored to consistent state.
- Back up data periodically
- Maintains log of system activity

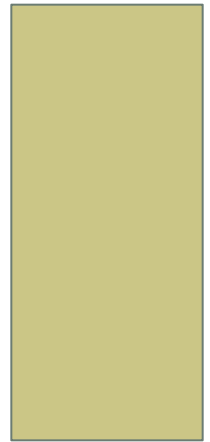
● Database tuning (Routine maintenance)

○ Schema and physical organization modification

DBA makes changes to the schema and physical organization to reflect the changing needs of the organization

- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations
- Monitoring jobs running on the database and ensuring its performance

COMPONENTS OF DBMS



MAJOR COMPONENTS OF A DBMS

1. Transaction management:
2. Concurrency Control:
3. Recovery Management:
4. Security Management:
5. Language Interface:
6. Storage Management:

TRANSACTION MANAGEMENT

- A transaction is a sequence of database operations that represents a logical unit of work and that accesses a database and transforms it from one state to another.
- For e.g. consider a database that holds information about airline reservations
 - Several travel agents look up information about available seats on various flights and make new seat reservation
 - When several users access a database concurrently, DBMS must order their requests carefully to avoid conflicts.
 - DBMS must protect users from the effects of system failures by ensuring that all data is restored to a consistent state

CONCURRENCY CONTROL

- This is the database management activity of coordinating the actions of database manipulation processes that operate concurrently that access shared data and potentially interfere with one another
- **Recovery Management**
- This in a database ensures the aborted or failed transactions (system crash) create no adverse effects on the database or the other transactions and should be restored to consistent state after crash.
- DBMS must ensure that the changes made by such incomplete transactions are removed from the database

- DBMS maintains a log of all writes to the database. Each write action must be recorded in the log before the corresponding change is reflected in the database itself. Property is called Write-Ahead Log or WAL
- **Security Management:** Refers to the protection of data against un-authorized access
- **Language Interface:** The DBMS provides support languages for definition and manipulation of data in the database.

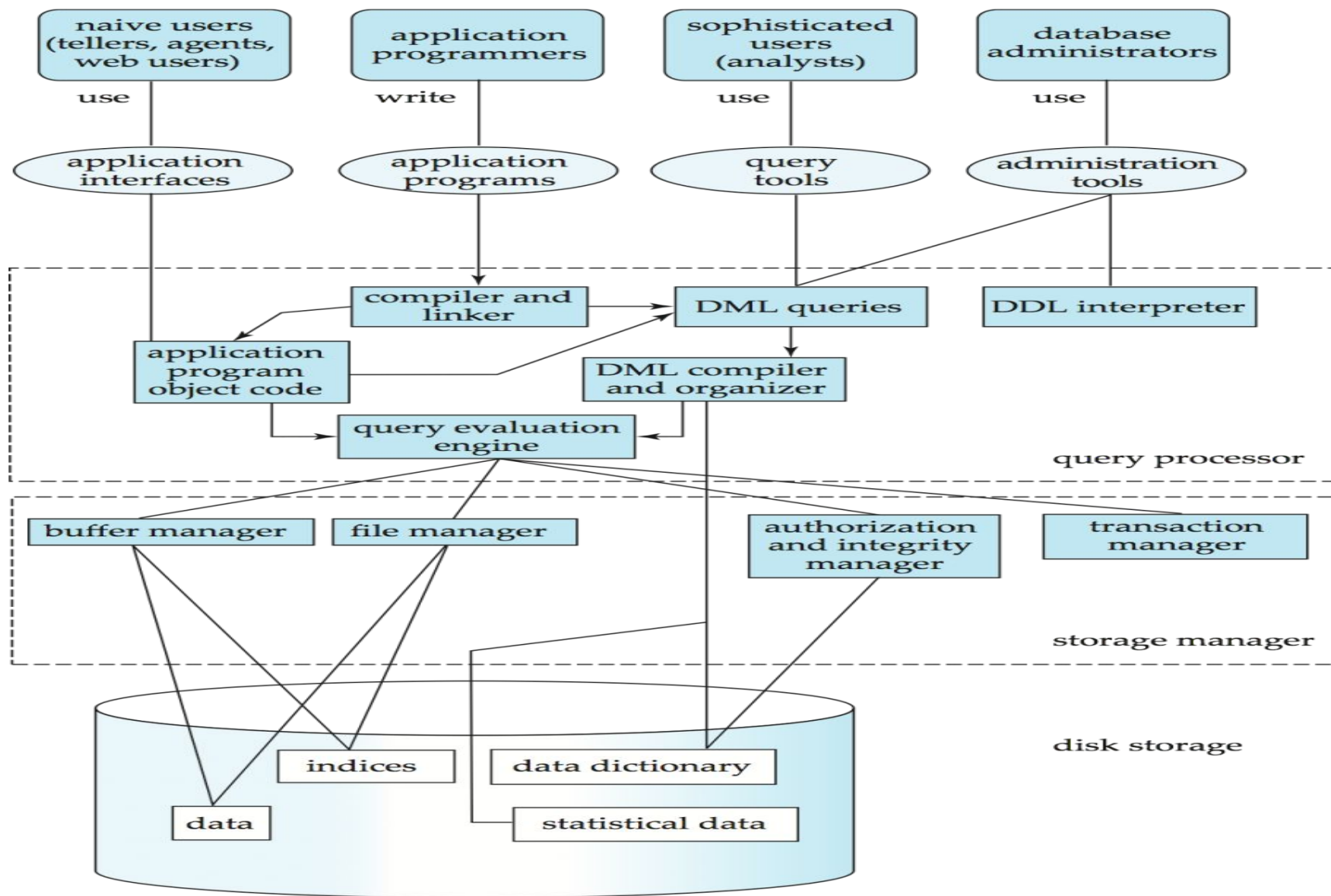
Storage Management: The DBMS provides a mechanism for management of permanent storage of the data.

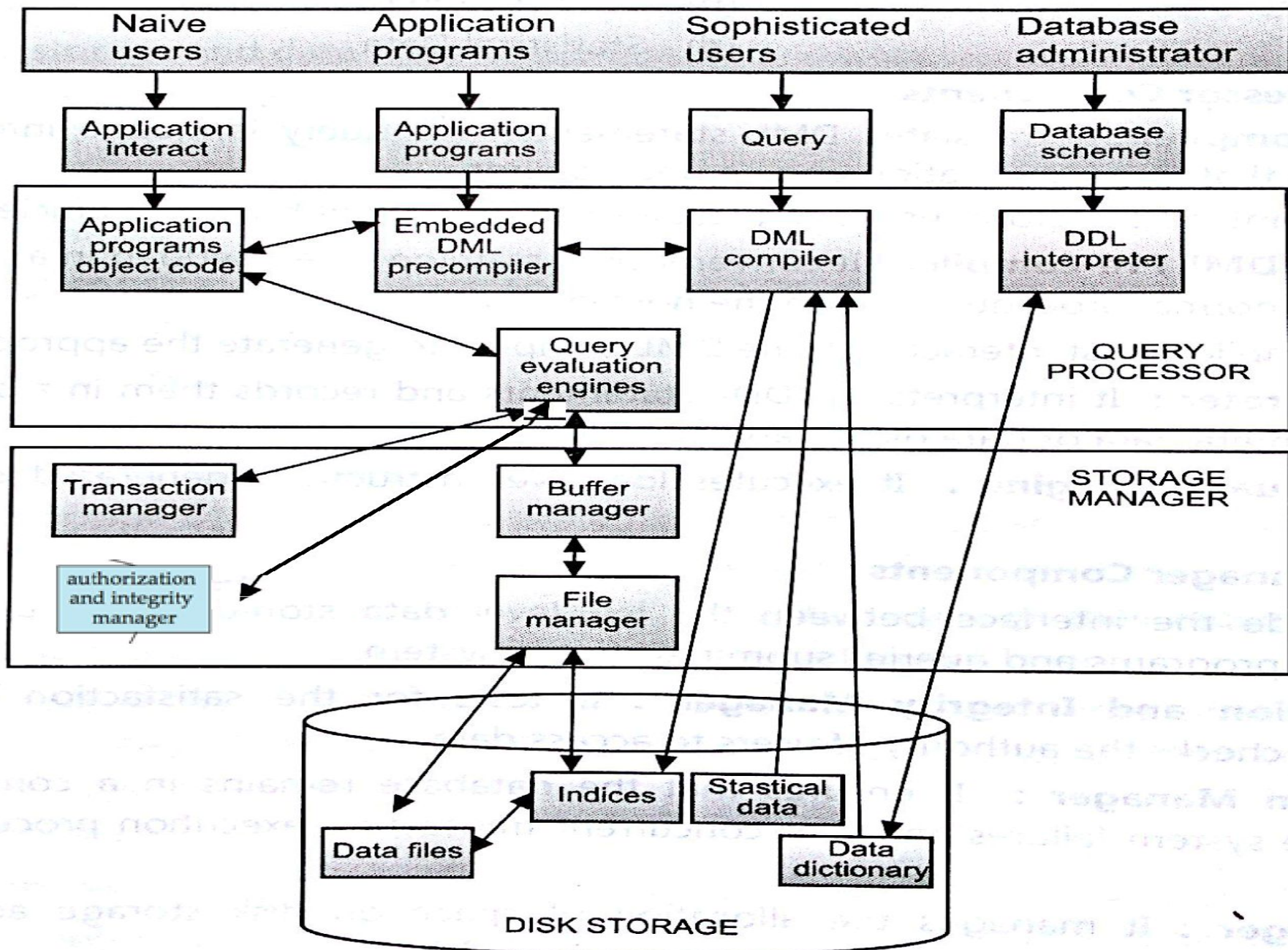
- Interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- Storage manager is responsible for storing, retrieving and updating data in the database.

OVERALL SYSTEM STRUCTURE

- DBMS is divided into modules that deals with each of the responsibilities of the overall system.
- Functional components of a DBMS are
 - ❑ Query processor components
 - ❑ Storage manager components

Database System Internals / DBMS Architecture





Query Processor components

- DML compiler
- Embedded DML pre compiler
- DDL interpreter
- Query evaluation engine

Storage manager components

- Authorization and integrity manager
- Transaction manager
- File manager
- Buffer manager

Query processor components

- **DML compiler**

Translates DML statements in a query language into low level instructions that the query evaluation engine understands.

- **Embedded DML precompiler**

- **Converts DML statements embedded in an application program to normal procedure calls.**
- **Interact with the DML compiler to generate the appropriate code**

Query processor components

- DDL interpreter

Interprets DDL statements and records the definition in the data dictionary

- Query evaluation engine

Executes low-level instructions generated by the DML compiler

Storage manager components

- Authorization & integrity manager

Tests for the satisfaction of integrity constraints and checks the authority of users to access data.

- Transaction manager

Ensures that the database remains in a consistent state despite system failures.

Storage manager components

- File manager

Manages the allocation of space on disk storage and the data structures used to represent information stored on disks.

- Buffer manager

Responsible for fetching data from disk storage into main memory and deciding what data to cache in memory.

SEVERAL DATA STRUCTURES REQUIRED FOR PHYSICAL SYSTEM IMPLEMENTATION

- **Data files**- which store the database itself
- **Data dictionary**- stores metadata about the structure of the database
- **Indices**- provides fast access to data items that hold particular values.
- **Statistical data**- stores statistical information about the data in the database.

Entity-Relationship Model

ADBS
Dr. Rahul Shajan

SJCET



MODELING

- 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



Key Attributes of an entity type

- A **key attribute** is a minimal set of attributes of an entity set, which **uniquely identifies** an entity in an entity set.
- A key may be a single attribute or may be more than one attribute.

Example:

- * For the student entity the **RegNo** can be the key attribute.
- * For a person entity the **SSN** can be the key attribute.
- Some times **key** may be formed by the **combination of several attributes** – a composite attribute. (ie., the combination of the attribute values must be distinct for each individual entity.

Example: The **registration no. for a vehicle** with two two simple attributes, i.e., state number. and registration number

Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
 - *Customer_id* is candidate key of *customer*
 - *account_number* is candidate key of *account*
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.

Value Sets (Domains) of Attributes

Value Sets (Domains) of Attributes:

- A **set of values** that may be assigned to the attributes of each individual entity, in an entity set is called the **value set or domain**.

Example:

- * For employee entity, if age limit is 20 to 58 then the value set (domain) of attribute age consists of **integer** from 20 to 58.

Age: Domain [20 – 58]

- * The value set for name attribute is a set of alphabets and some special characters.

Name: Domain [a – z], [A – Z], blank space, dot

ENTITY SETS *CUSTOMER* AND *LOAN*

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

321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

ATTRIBUTES

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

Example:

*customer = (customer-id, customer-name,
customer-street, customer-city)*
loan = (loan-number, amount)

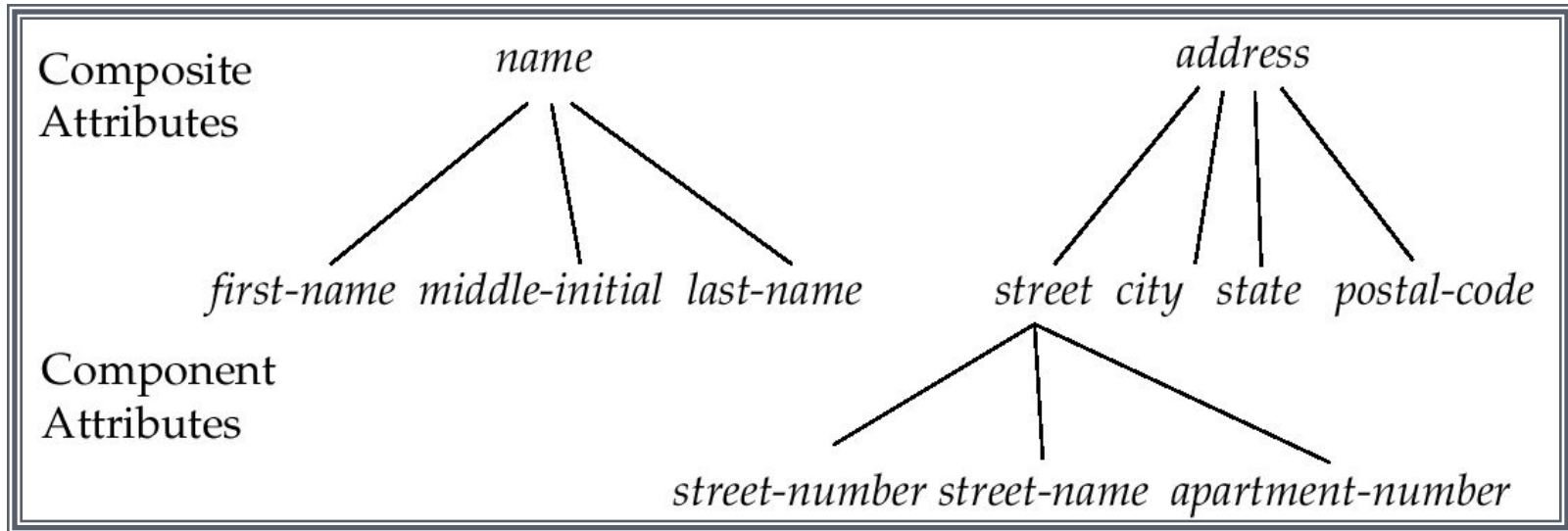
- **Domain** – the set of permitted values for each attribute
- **Attribute types**:
 - *Simple* and *composite* attributes.
 - *Single-valued* and *multi-valued* attributes
 - *Derived* attributes



- Simple attributes: Attributes which are not divided into sub parts.it is called atomic attributes
- Eg: age
- Composite attributes: Attributes which can be divided into subparts, helps us to group together related attributes.
- Eg: Name- First name, Middle initial, Last name



COMPOSITE ATTRIBUTES



Name, Address are the composite attributes.

First name, middle initial, last name, street, city, state, postal code are component attributes.

Street no, street name, apartment number are component attributes.

- Single valued attribute: Attributes having a single value for a particular entity.
- Eg: loan number attribute for a specific loan entity refers to only one loan number.
- Eg: Date of birth
- Multivalued attribute: Attributes having a set of values for a specific entity.
- Eg: an emp entity set with the attribute phno
- entity student can have multiple values for the hobby attribute- reading, listening music etc
- Derived attributes: Values for this type of attributes can be derived from the value of another other related attributes or entities.



- e.g. entity set customer has an attribute age, which indicates the customer's age. If the customer entity set also has an attribute date-of-birth. Age is calculated from date-of-birth and the current date. Thus age is a derived attribute.
- In this case dob may be referred to as **base attribute** or **stored attribute**.
- **Stored attribute**: the value of derived attribute is not stored but is computed when required.
- **Null attribute**: Attribute can be null. A null value is used when an entity does not have a value for an attribute.
- Null also indicates attribute value is unknown.



RELATIONSHIP SETS

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

Example: Book is published by a particular publisher.

Relationship set is the set of all relationship of the same type.

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets.
- E_1, E_2, \dots, E_n are *entity sets* then *relationship set* R is *defined as*

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

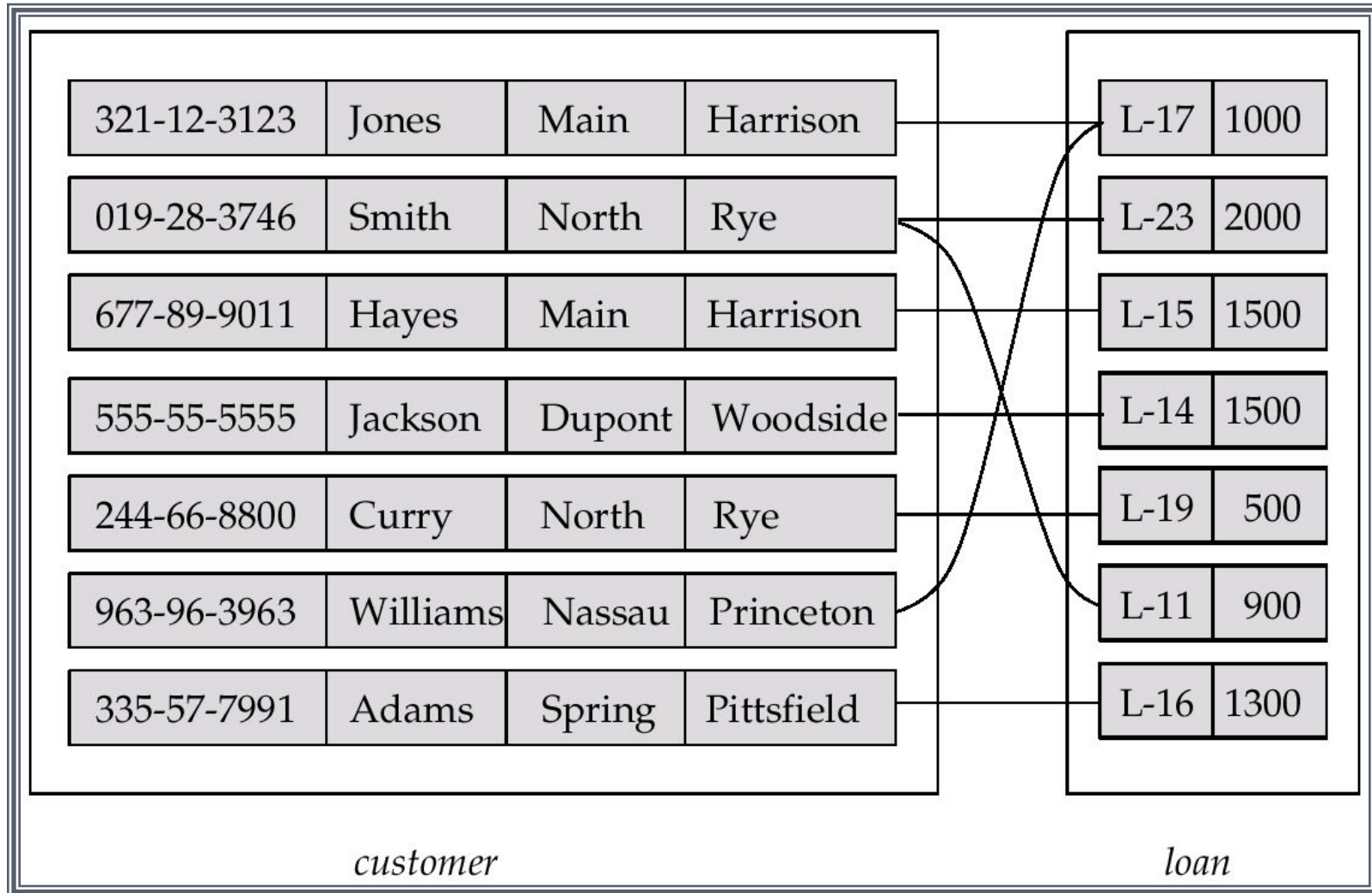
where (e_1, e_2, \dots, e_n) is a relationship

- Example:

$$(\text{Hayes}, \text{A-102}) \in \text{depositor}$$

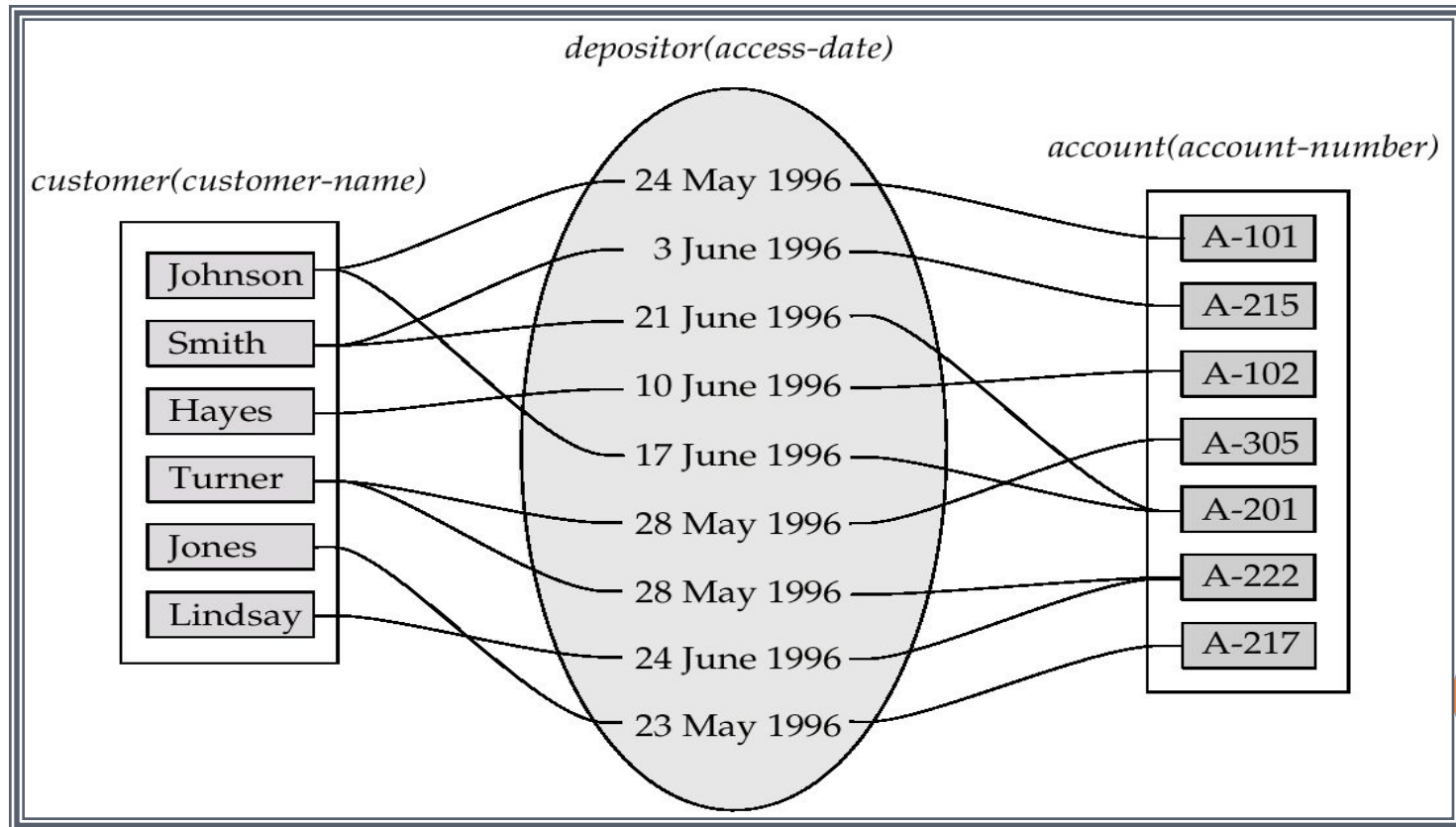


RELATIONSHIP SET *BORROWER*



RELATIONSHIP SETS (CONT.)

- A relationship may also have attribute called **descriptive attribute**.
- An *attribute* can also be property of a relationship set.
- For instance, the **depositor** relationship set between entity sets **customer** and **account** may have the attribute **access-date**

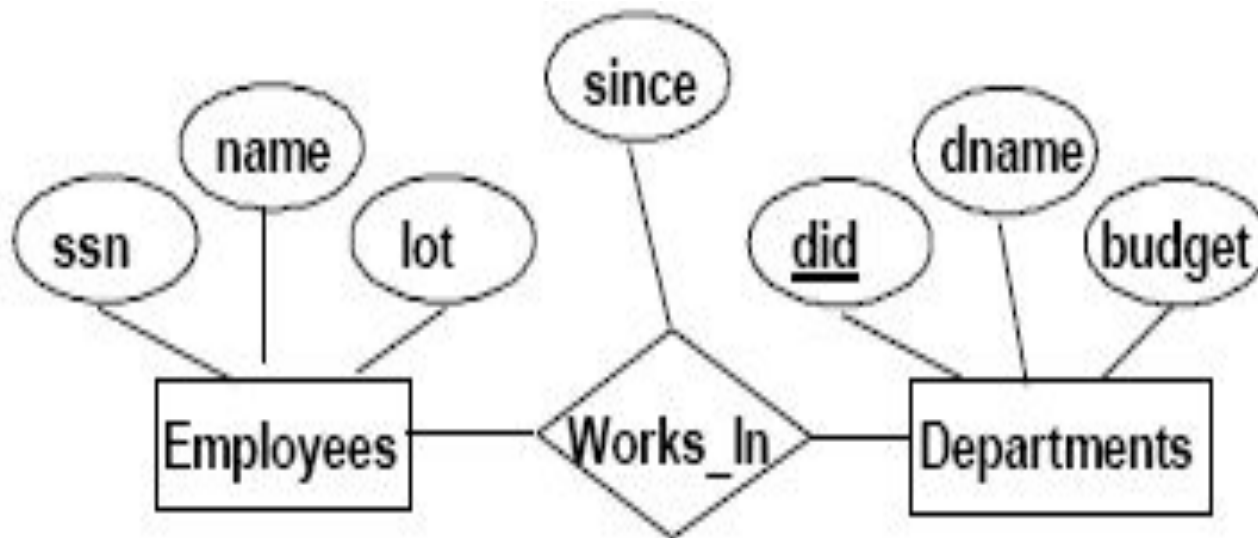


DEGREE OF A RELATIONSHIP SET

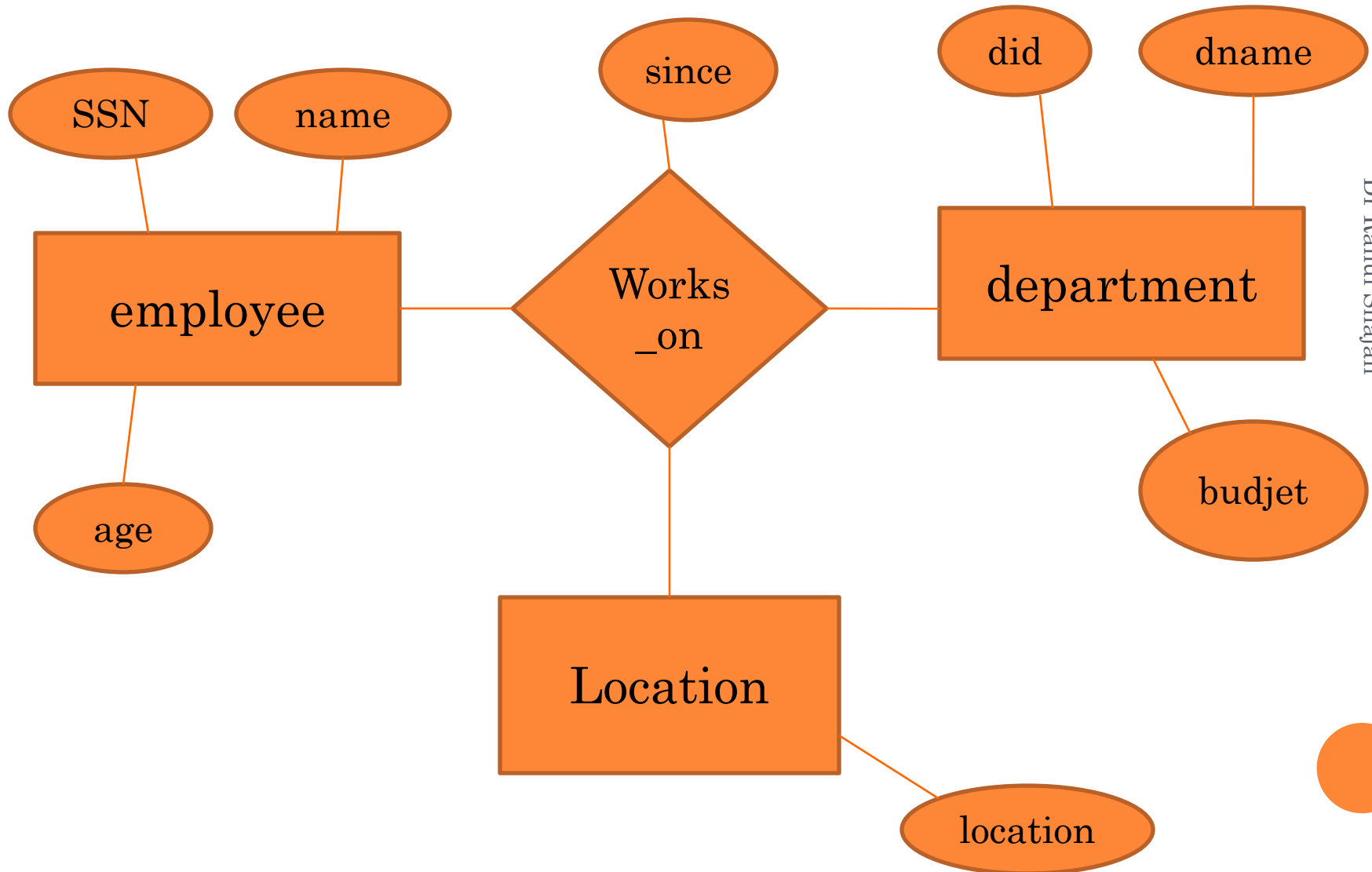
- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are **binary** (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
 - E.g. Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee*, *job* and *branch*
- Relationships between more than two entity sets are rare. Most relationships are binary.
- Relationship types of **degree 3** are called **ternary** and of **degree n** are called **n-ary**



E.G. BINARY RELATIONSHIP



E.G. TERNARY RELATIONSHIP



ER DIAGRAM

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities.

An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

Entities

Relationships

Attributes



- ❑ **Rectangles** represent entity sets. entity written in upper case, where as the attribute name is written in lower case.
- ❑ **Diamonds** represent relationship sets.
- ❑ **Lines** link attributes to entity sets and entity sets to relationship sets.
- ❑ **Underline** indicates primary key attributes
- ❑ **Ellipses** represent attributes
 - ✓ ***Double ellipses represent multivalued attributes.***
 - ✓ ***Dashed ellipses denote derived attributes.***



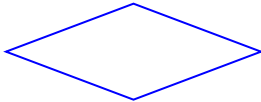
ERD symbols



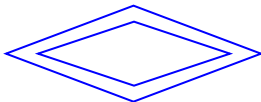
Entity



Weak entity



Relationship



Identifying relationship



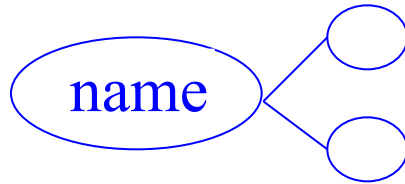
Attribute



Key attribute



ERD symbols



Composite attribute



Derived attribute



Mutlti-valued attribute



Partial participation



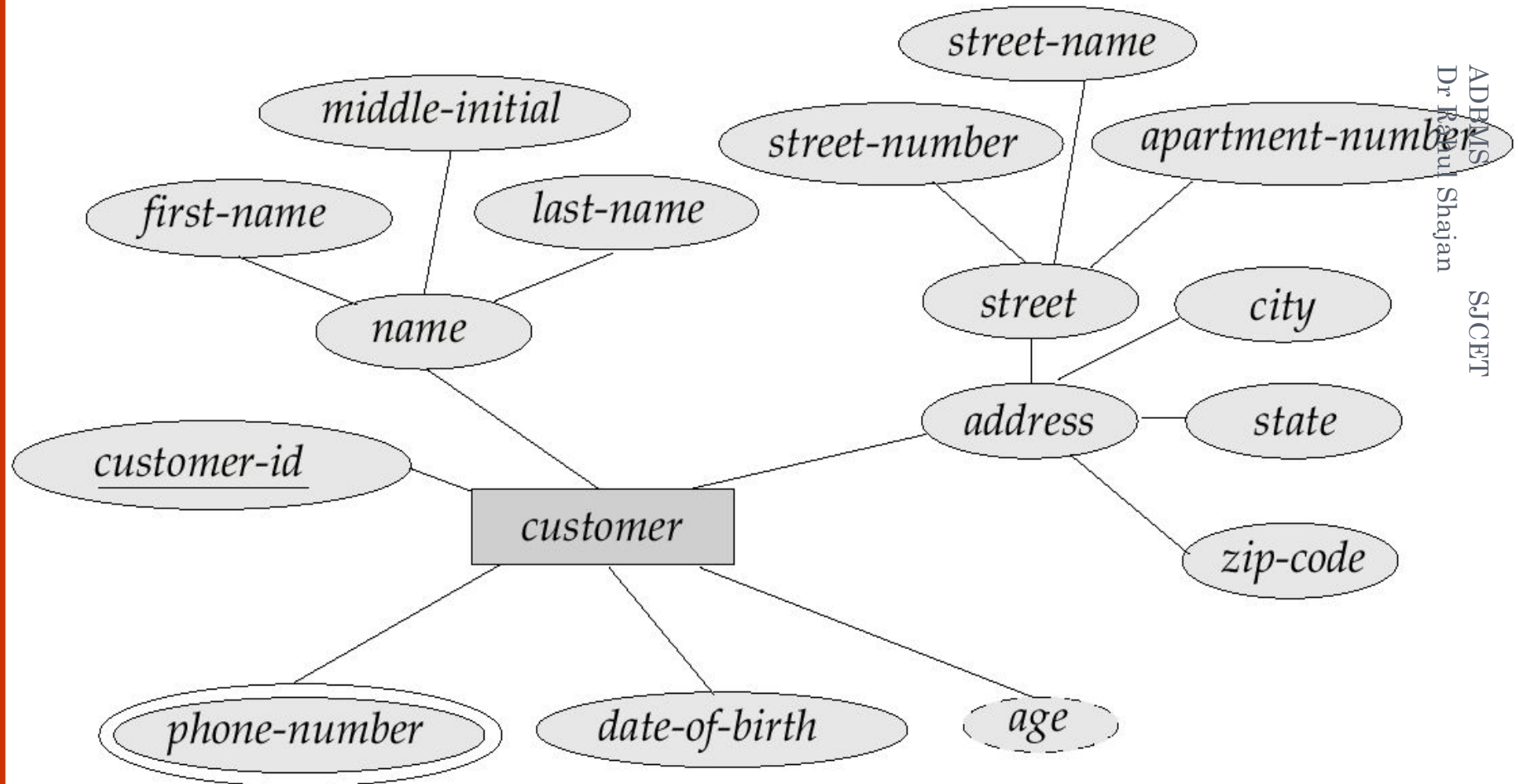
Total participation

1 n N

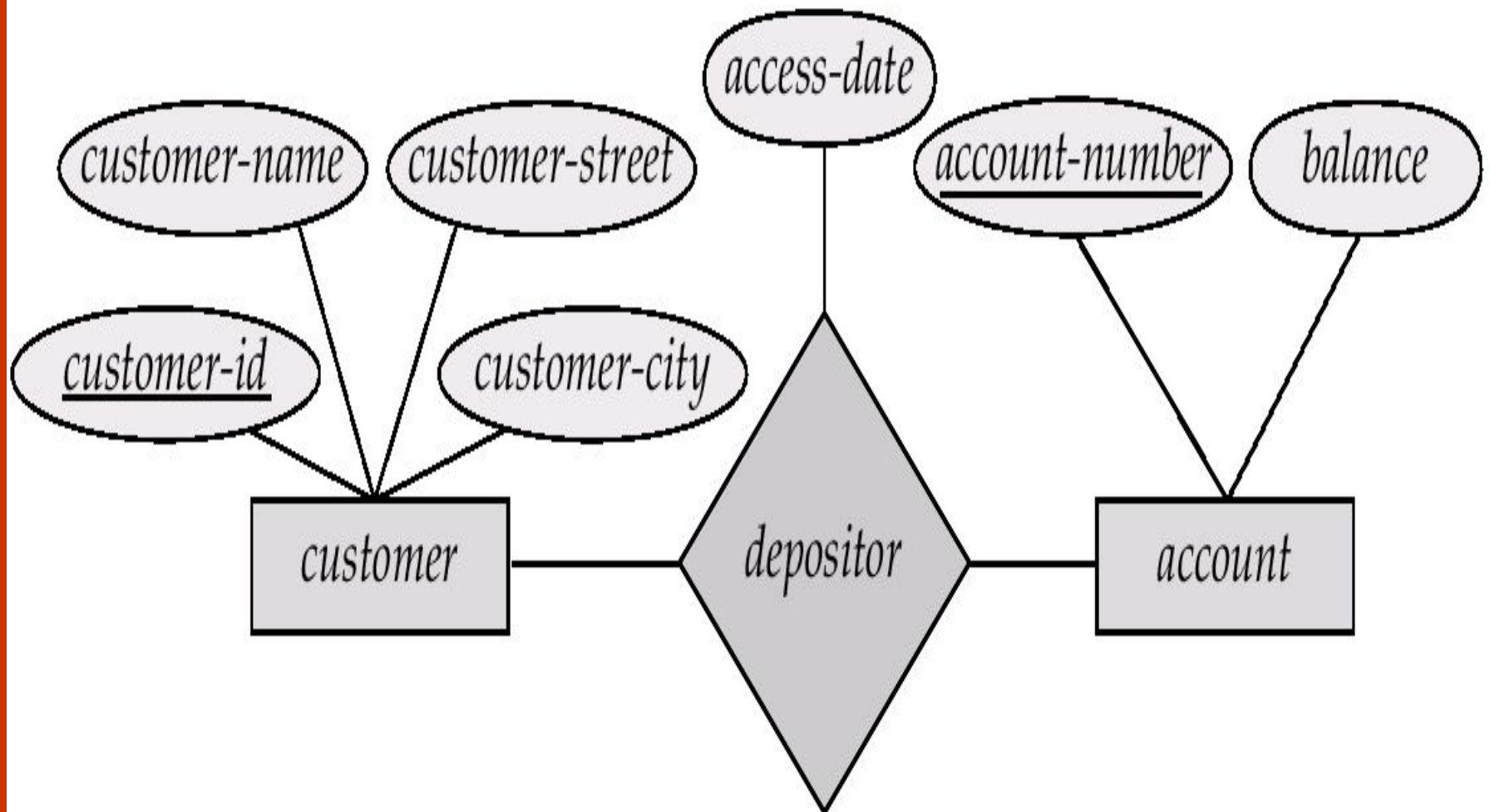
Cardinality



ER diagram with composite, Multivalued and Derived attributes



EG: Relationship sets with attributes



Additional features of ER model



MAPPING CARDINALITIES

- 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



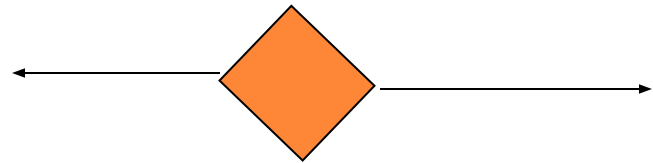
CARDINALITY CONSTRAINTS

- We express cardinality constraints by drawing either a directed line (\rightarrow), signifying “one,” or an undirected line ($—$), signifying “many,” between the relationship set and the entity set.

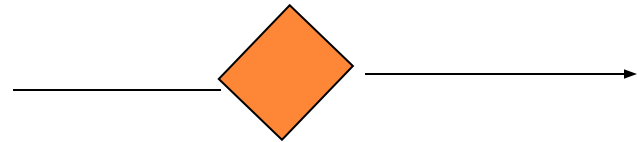


MAPPING CARDINALITY

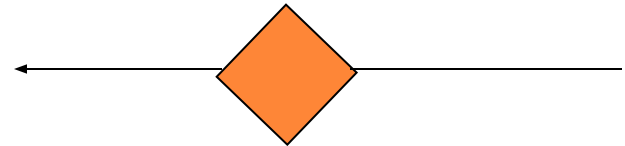
One to
One



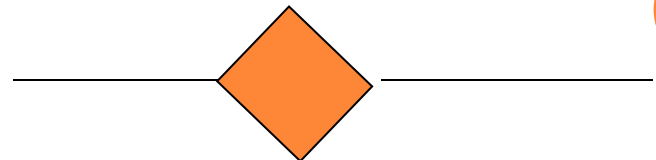
Many to
one

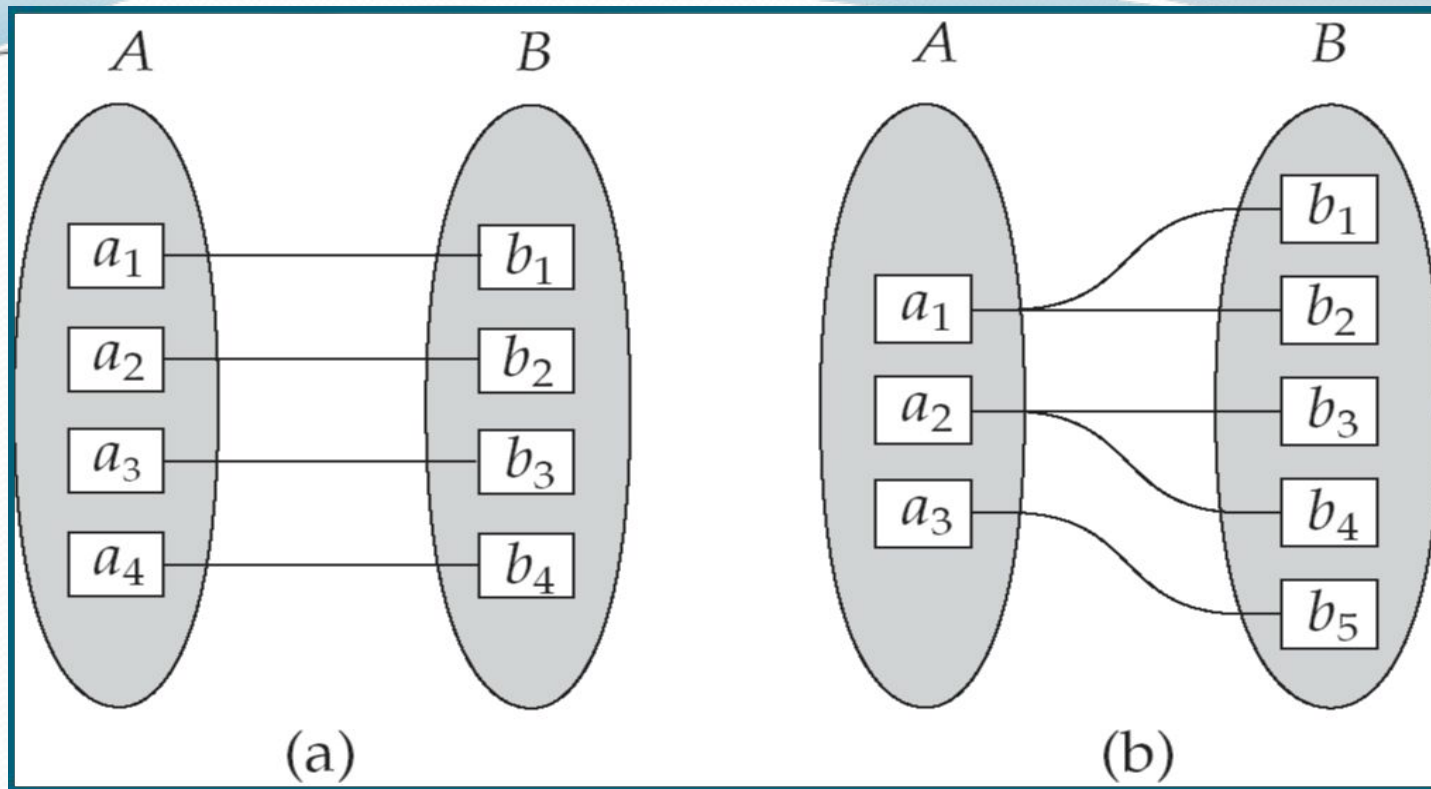


One to
many



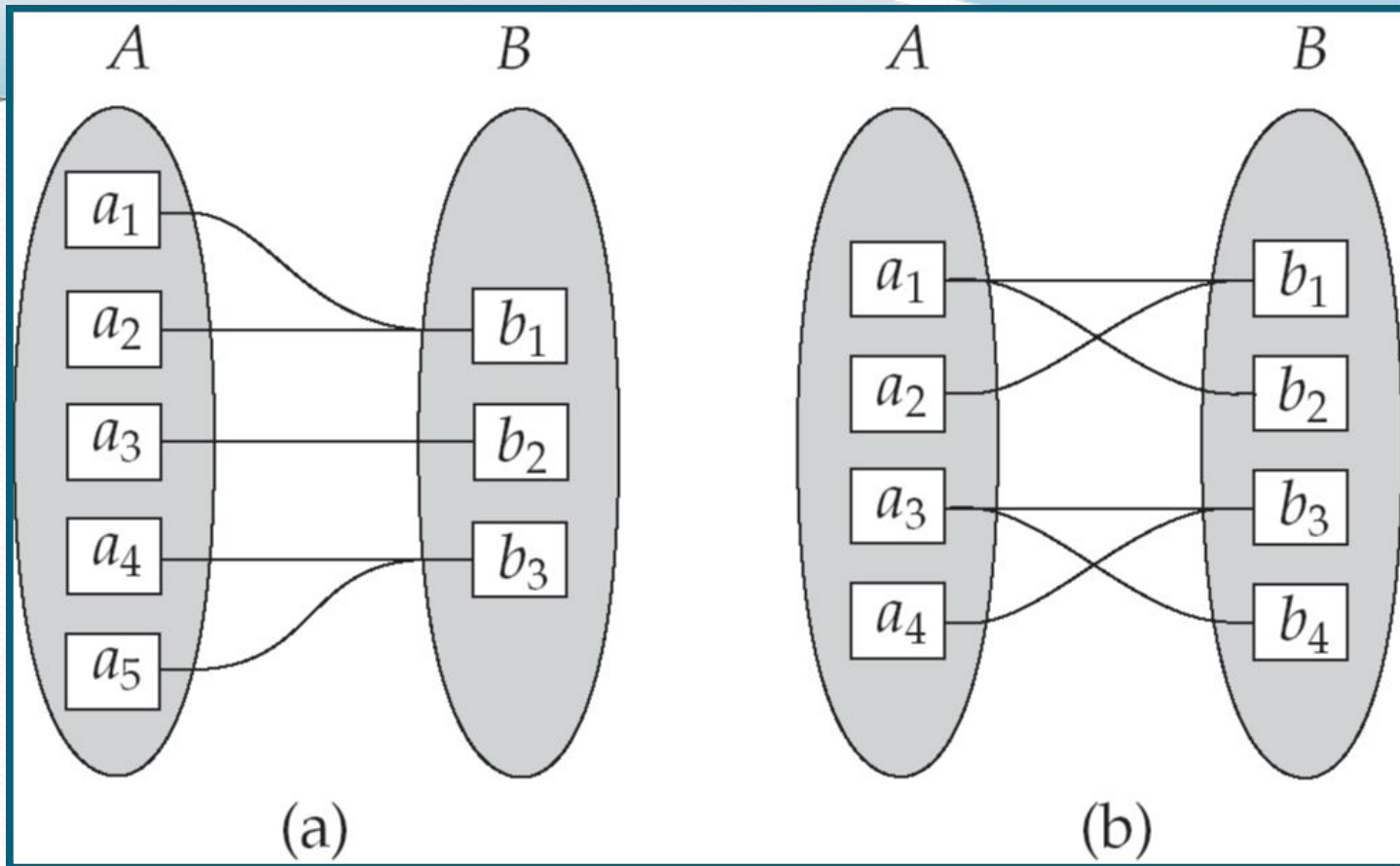
Many to
many





One to One: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

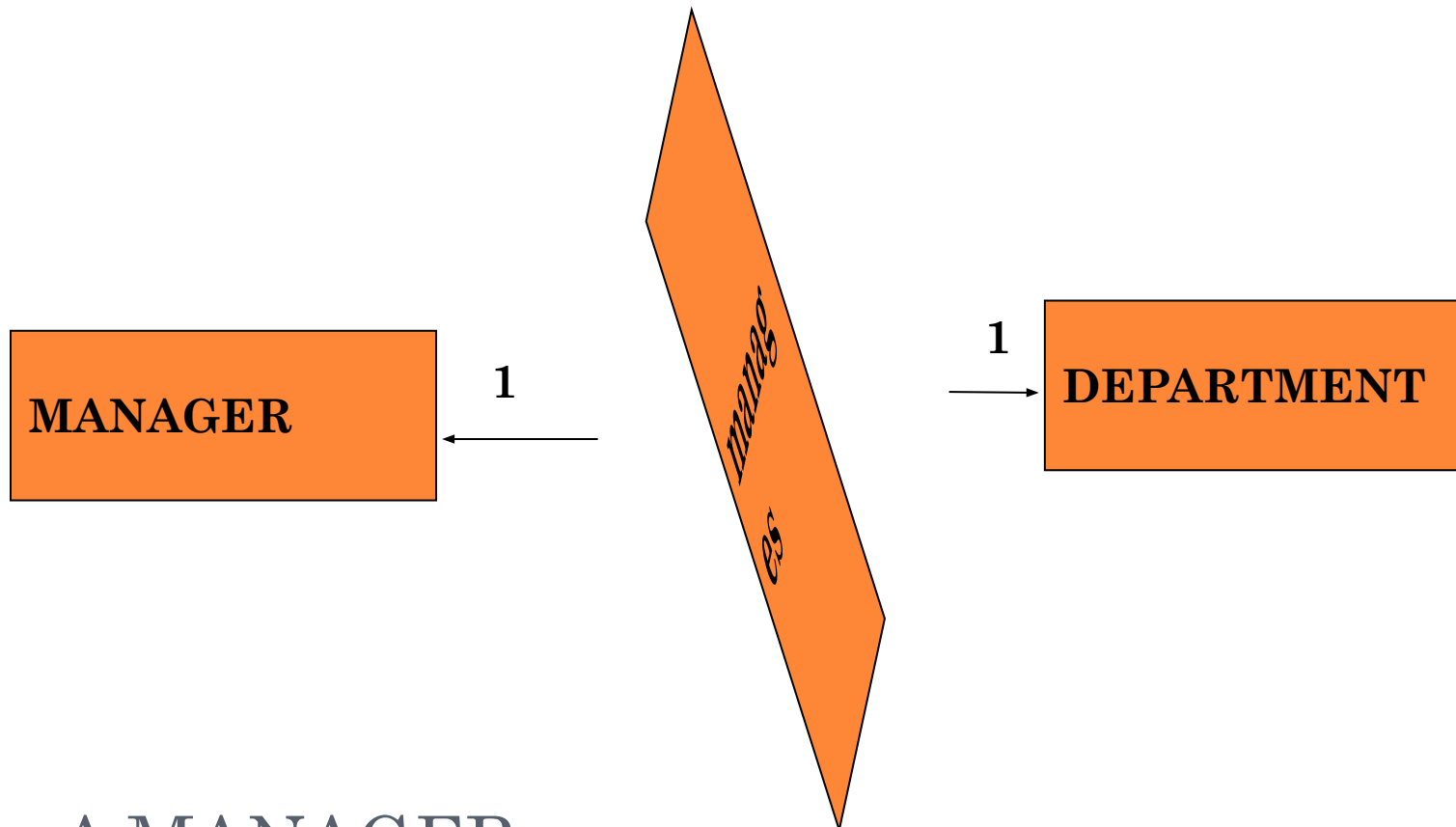
One to Many: An entity in A is associated with any number of entities in B. An entity in B, however can be associated with at most one entity in A.



Many to One: An entity in A is associated with at most one entity in B, and an entity in B is associated with any number of entities in A.

Many to Many: An entity in A is associated with any number of entities in B. and n entity in B, is associated with any number of entities in A.

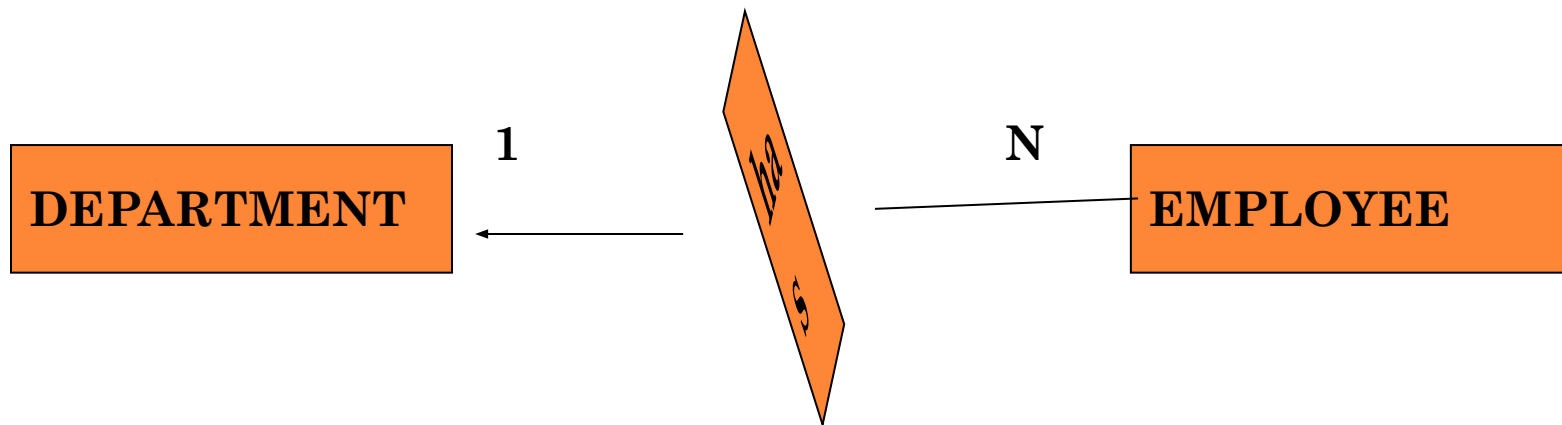
ONE TO ONE RELATIONSHIP



A **MANAGER** IN THE COMPANY MANAGES A SINGLE **DEPARTMENT**



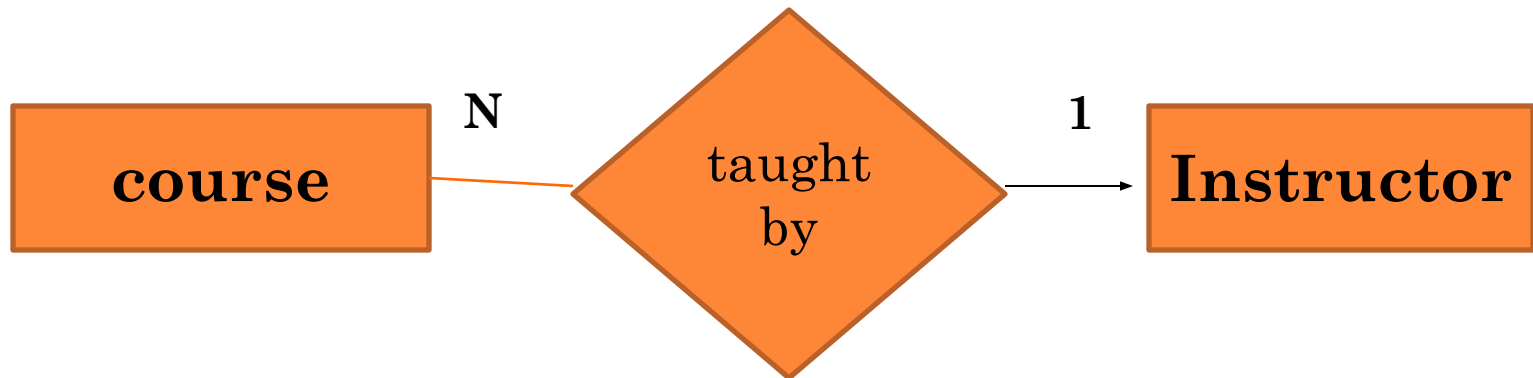
ONE TO MANY RELATIONSHIP



A DEPARTMENT CAN HAVE MORE THAN ONE EMPLOYEE



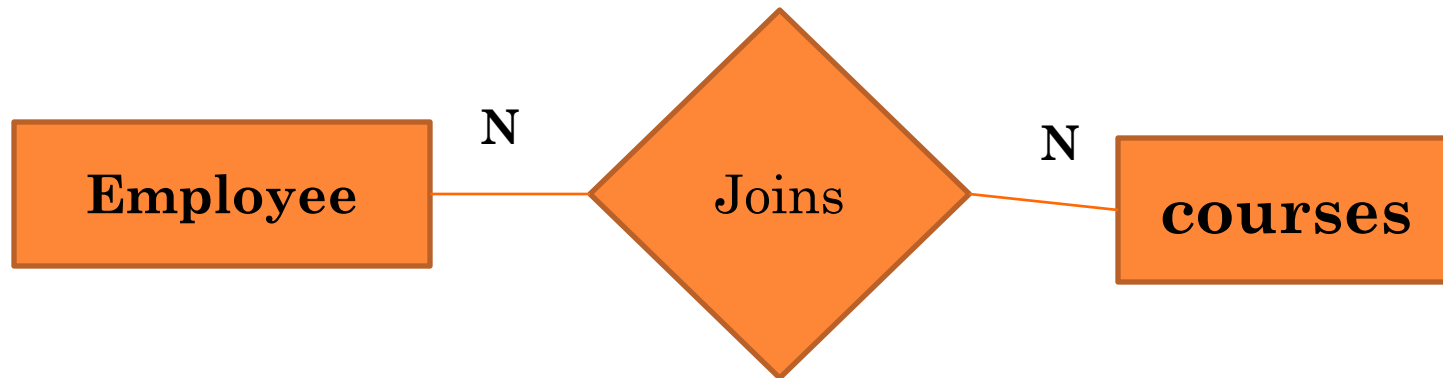
MANY TO ONE RELATIONSHIP



N COURSES CAN BE TAUGHT BY ONE INSTRUCTOR



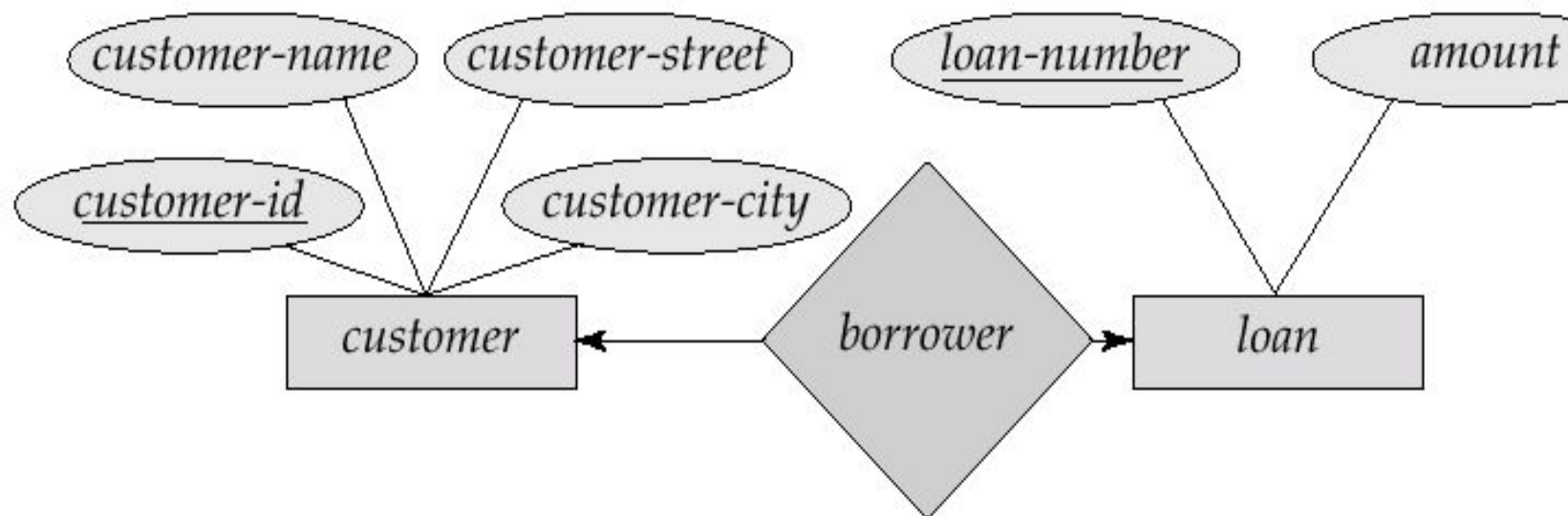
MANY TO MANY RELATIONSHIP



EACH EMPLOYEE CAN JOIN FOR MORE THAN ONE COURSE

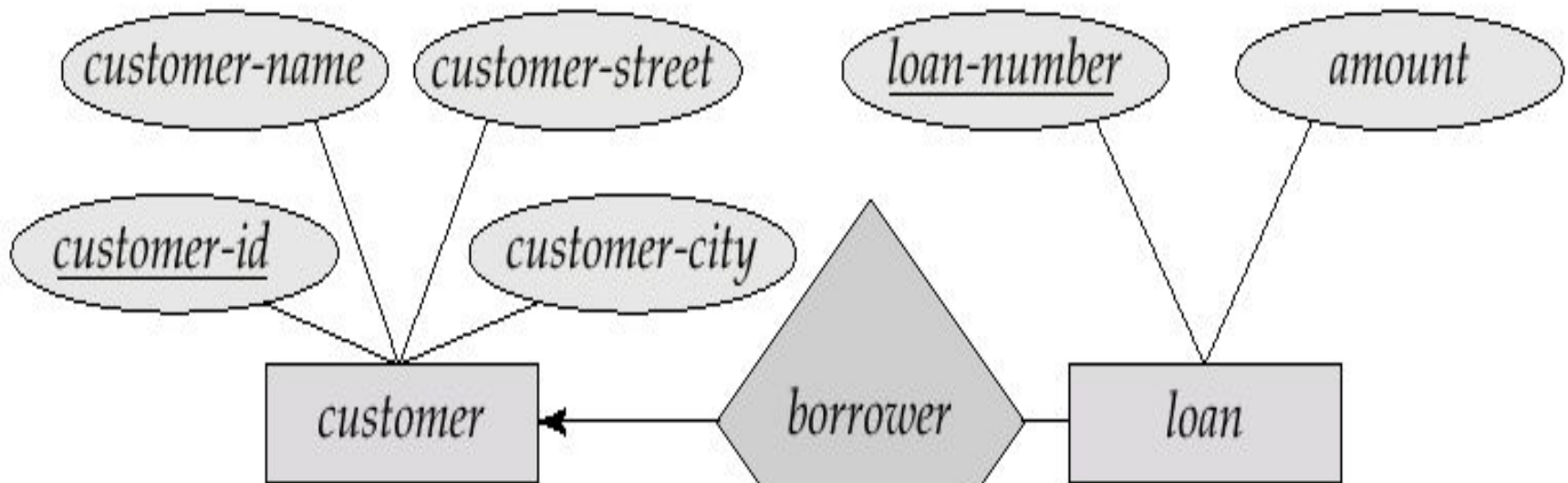
ONE TO ONE RELATIONSHIP

- A customer is associated with at most one loan via the relationship *borrower*
- A loan is associated with at most one customer via *borrower*



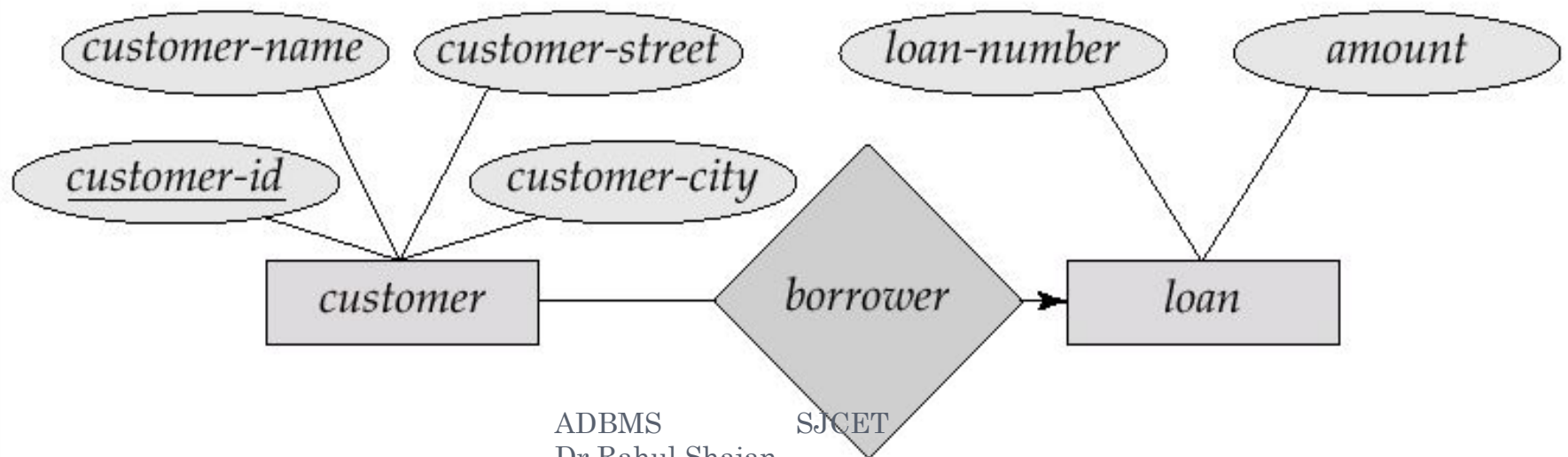
ONE TO MANY RELATIONSHIP

- In the one-to-many relationship a loan is associated with at most one customer via *borrower*, a customer is associated with several (including 0) loans via *borrower*



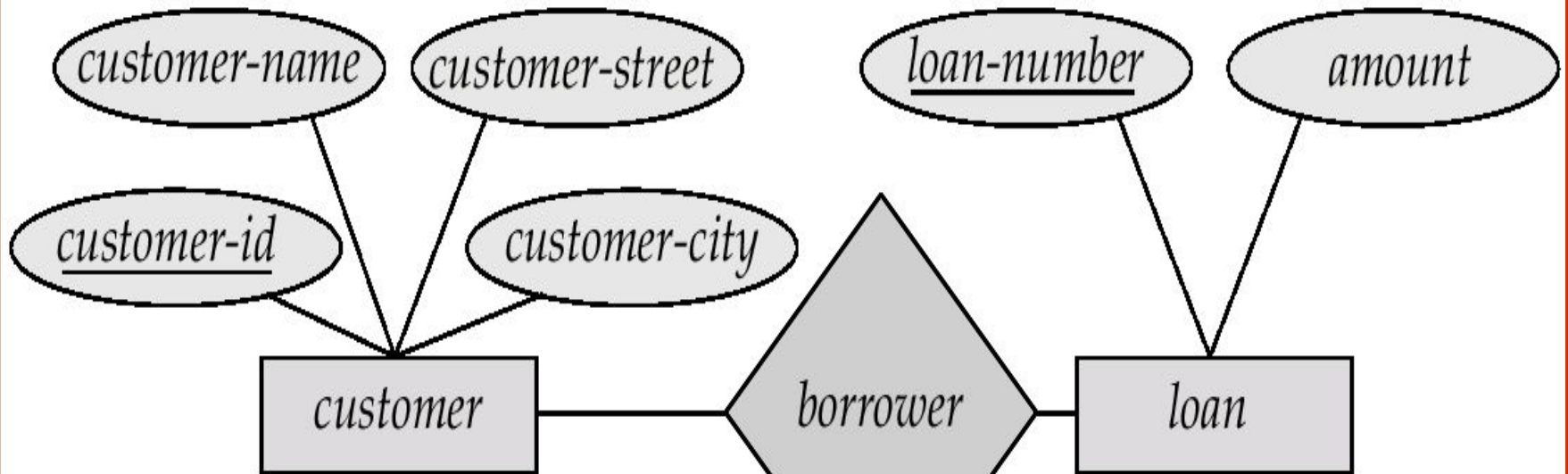
MANY TO ONE RELATIONSHIP

- In a many-to-one relationship a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*



MANY TO MANY RELATIONSHIP

- A customer is associated with several (possibly 0) loans via borrower and A loan is associated with several (possibly 0) customers via borrower



PARTICIPATION OF ENTITY SET IN A RELATIONSHIP SET

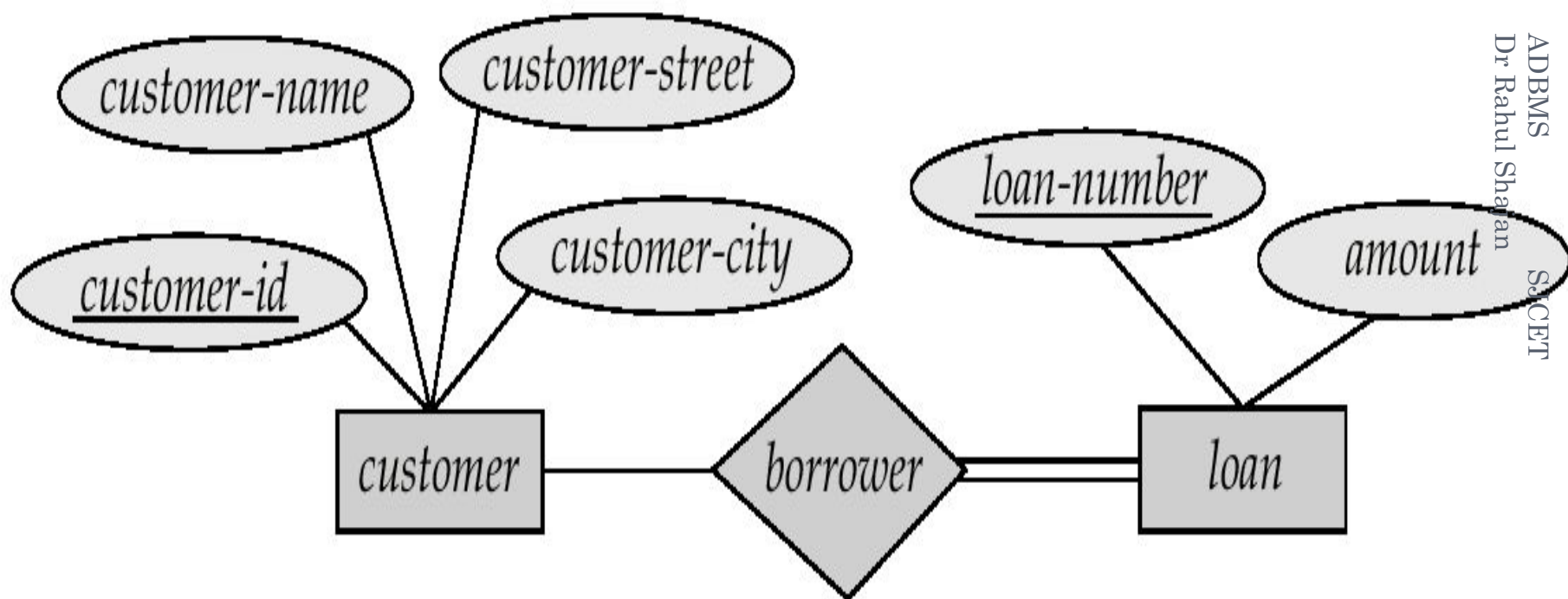
□ **Total participation** (indicated by double line) every entity in the entity set participates in at least one relationship in the relationship set

✓ *E.g. participation of loan in borrower is total*

✓ *Every loan must have a customer associated to it via borrower*

□ **Partial participation** some entities may not participate in any relationship in the relationship set

✓ *E.g. participation of customer in borrower is partial*



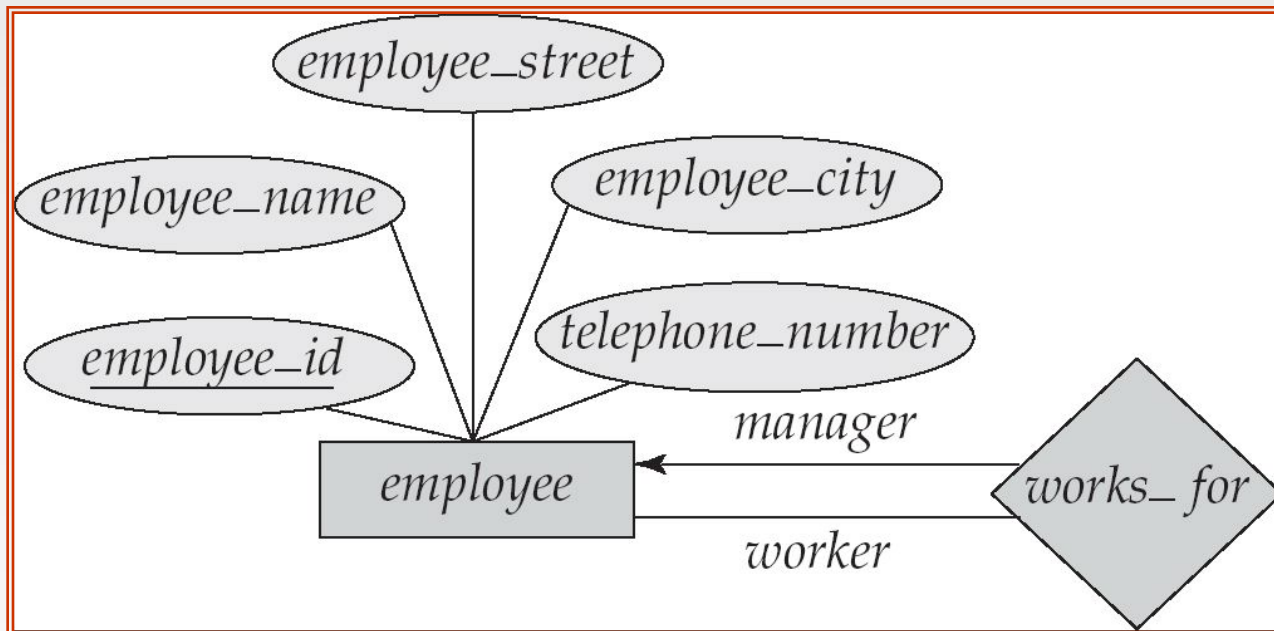
ROLES

- A relationship can be between entities of one entity set; the two entities in such a relationship are in different **roles**.
 - E.g., Two employees in the employee entity set are involved in a works_for relationship. One has a role of “**manager**” and the other “**worker**”.

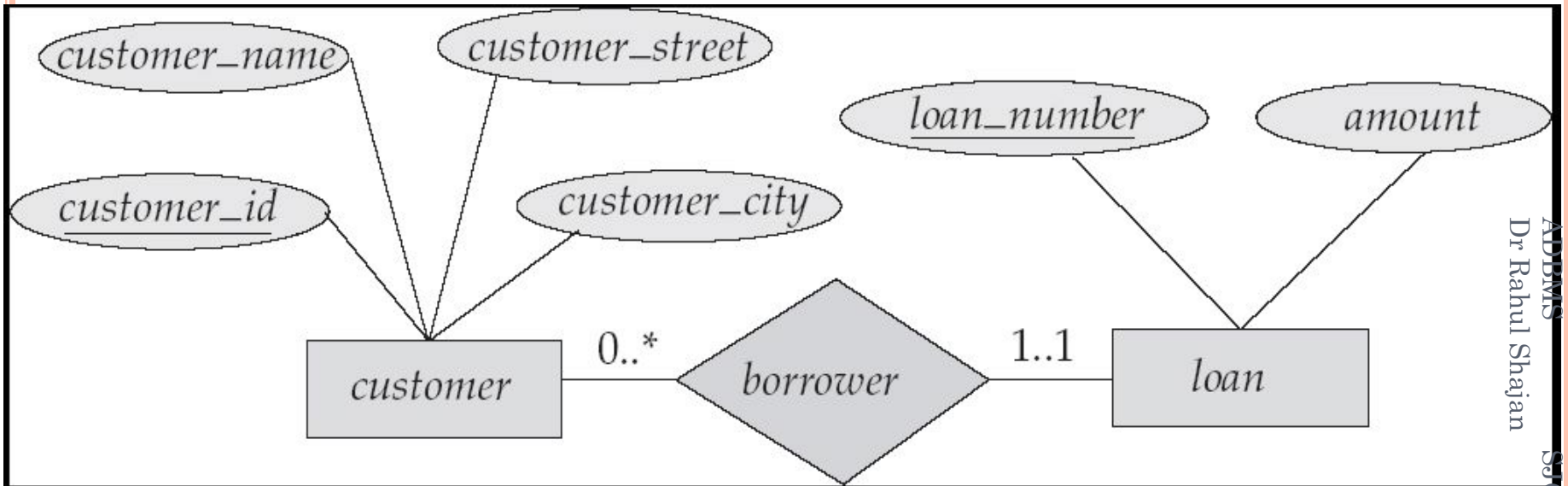


Roles

- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship



ALTERNATIVE NOTATION FOR CARDINALITY LIMITS



ADAMS
Dr Rahul Shajan
SECRET

An edge between entity set and relationship can have **minimum** and **maximum** cardinality shown in form of $l..h$,
 l minimum , h maximum

Minimum 1- indicates total participation

Maximum 1-indicates entity participate in at most 1 relationship

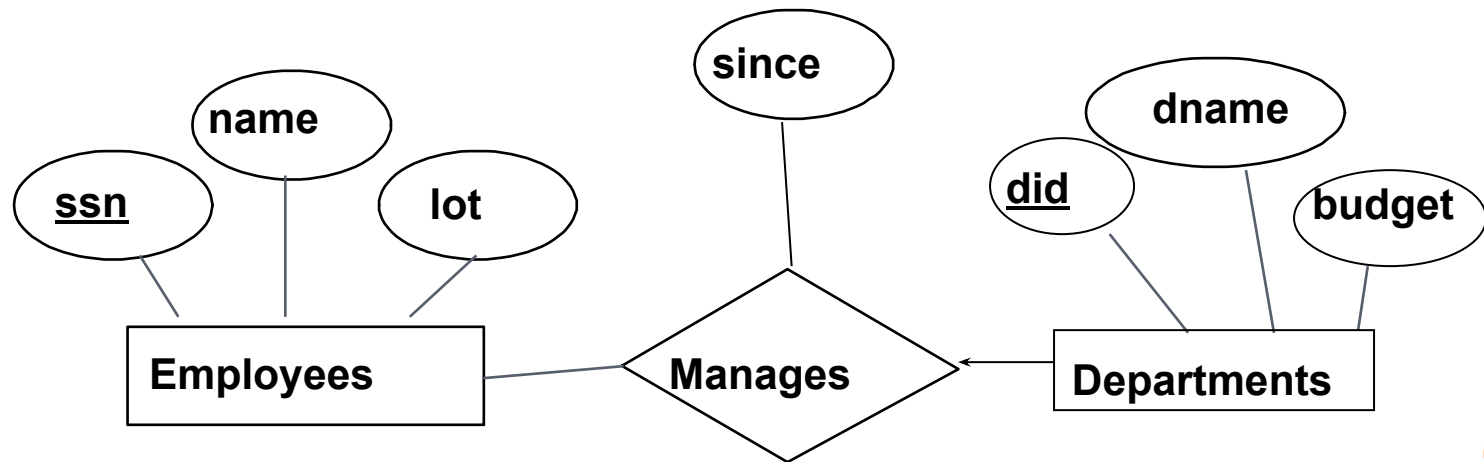
Maximum value * indicates no limit



KEY CONSTRAINT

- A key constraint between an entity set S and a relationship set restricts instances of the relationship set by requiring that each entity of S participate in at most one relationship

Eg: each dept has at most one manager, according to the key constraint on Manages.



Indicated by using an arrow from departments to manages

- The restriction that each department has at most one manager is an example of a **key constraint**, and it implies that each Departments entity appears in at most one Manages relationship in any allowable instance of Manages. **This restriction is indicated in the ER diagram by using an arrow from Departments to Manages.**



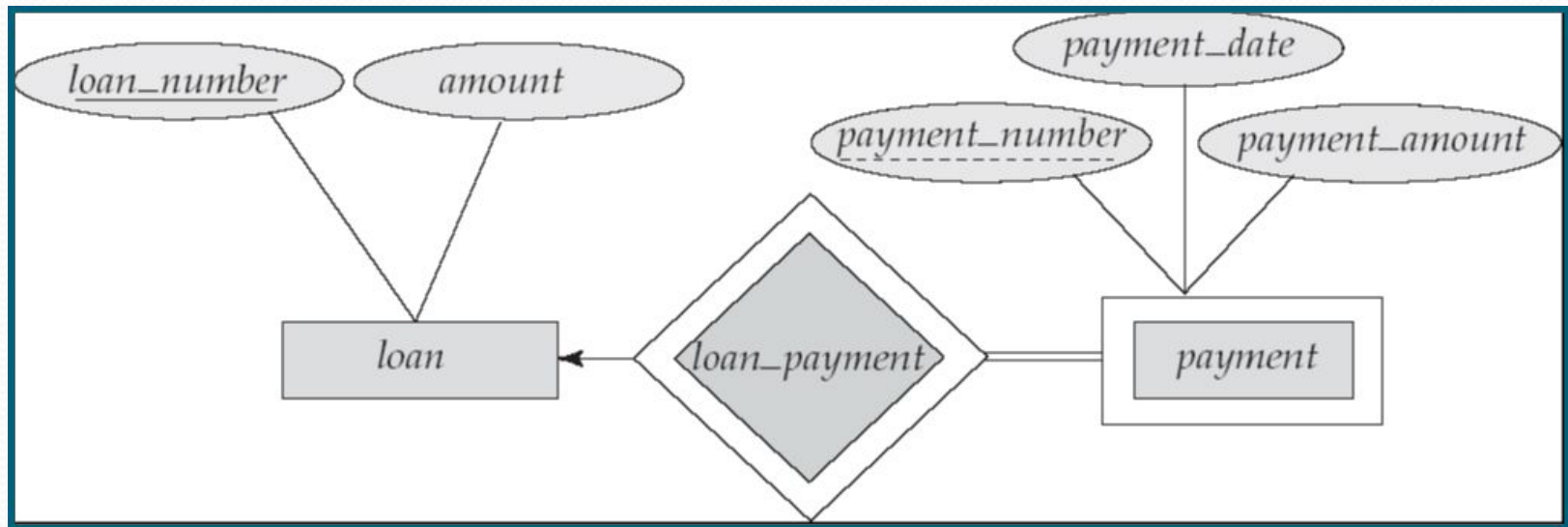
Weak entity set

- An entity set that does not have a primary key is referred to as a weak entity s.
- An entity set that has a primary key is termed strong entity set.
- .
- The existence of a weak entity set depends on the existence of a Strong entity set
 - Identifying weak entity set is represented with double-line rectangle.
 - Identifying relationship is the relationship between Strong and weak entity sets and it is represented by 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.

Identifying relationship

- We represent **weak entity sets by double rectangles**.
- We underline the discriminator of a weak entity set with a dashed line.

payment_number – discriminator of the *payment* entity set
Primary key for *payment* – (*loan_number*, *payment_number*)



Additional features of ER model



Class Hierarchies-Specialization & Generalization



Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entities to make further higher level entity. It's more like Superclass and Subclass system, but the only difference is the approach, which is bottom-up. Hence, entities are combined to form a more generalised entity, in other words, sub-classes are combined to form a super-class.

Specialization

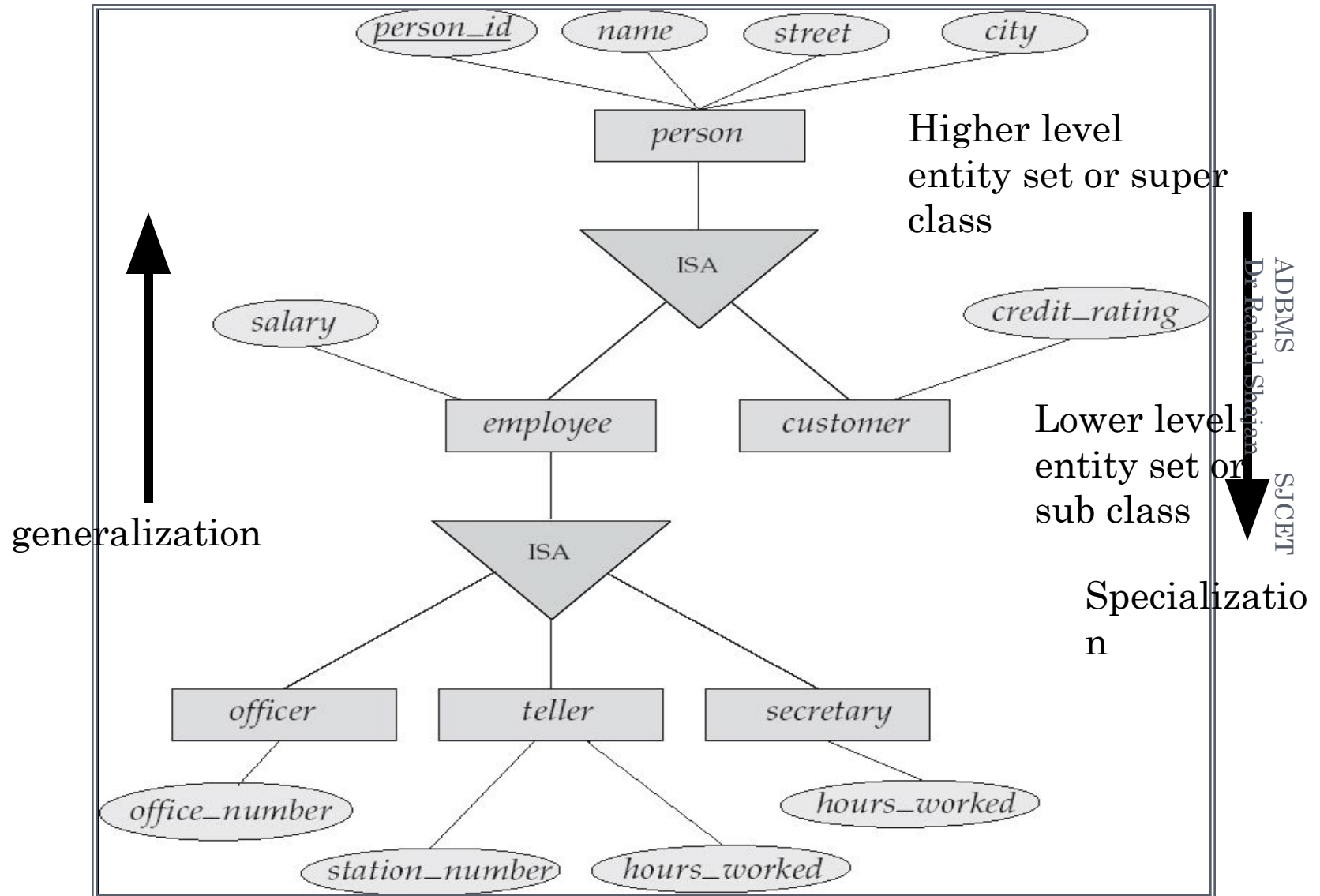
An entity set may include **sub groupings of entities** that are distinct in some way from other entities. (i.e a subset of entities within an entity set may have attributes **that are not shared** by all entities in the entity set.)

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, a higher level entity may not have any lower-level entity sets, it's possible.

Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the super class VEHICLE.

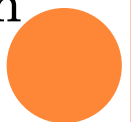
- ❑ We can view {CAR, TRUCK} as a **specialization** of VEHICLE
- ❑ Alternatively, we can view VEHICLE as a **generalization** of CAR and TRUCK

SPECIALIZATION / GENERALIZATION EXAMPLE



DESIGN CONSTRAINTS ON A SPECIALIZATION/GENERALIZATION

- Constraint on which entities can be members of a given lower-level entity set.
 - condition-defined
- Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
- If all subclasses in a specialization have membership condition on same attribute of the superclass, specialization is called an attribute-defined specialization
 - Attribute is called the defining attribute of the specialization
 - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE
- If no condition determines membership, the subclass is called user-defined
 - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
 - Membership in the subclass is specified individually for each entity in the superclass by the user



Constraints on Specialization and Generalization (4)

- Disjointness Constraint:
 - Specifies that the subclasses of the specialization must be *disjoint*:
 - an entity can be a member of at most one of the subclasses of the specialization
 - Specified by **d** in EER diagram
 - Noted in E-R diagram by writing *disjoint* next to the ISA triangle
 - If not disjoint, specialization is *overlapping*:
 - that is the same entity may be a member of more than one subclass of the specialization
 - Specified by **o** in EER diagram

Constraints on Specialization and Generalization (5)

■ Completeness Constraint:

- *Total* specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization
- Shown in EER diagrams by a **double line**
- *Partial* allows an entity not to belong to any of the subclasses
- Shown in EER diagrams by a single line

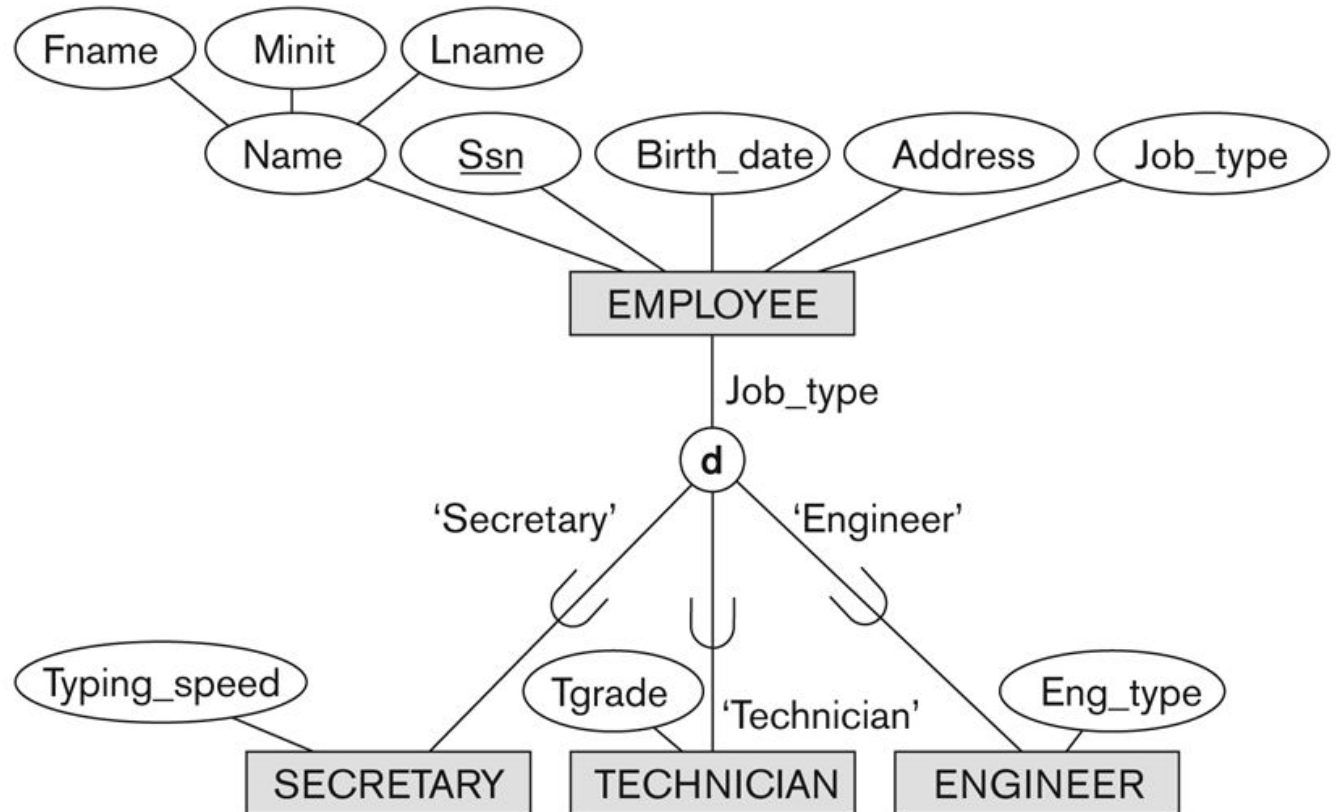
Constraints on Specialization and Generalization (6)

- Hence, we have four types of specialization/generalization:
 - Disjoint, total
 - Disjoint, partial
 - Overlapping, total
 - Overlapping, partial
- Note: Generalization usually is total because the superclass is derived from the subclasses.

Example of disjoint partial Specialization

Figure 4.4

EER diagram notation for an attribute-defined specialization on Job_type.



Example of overlapping total Specialization

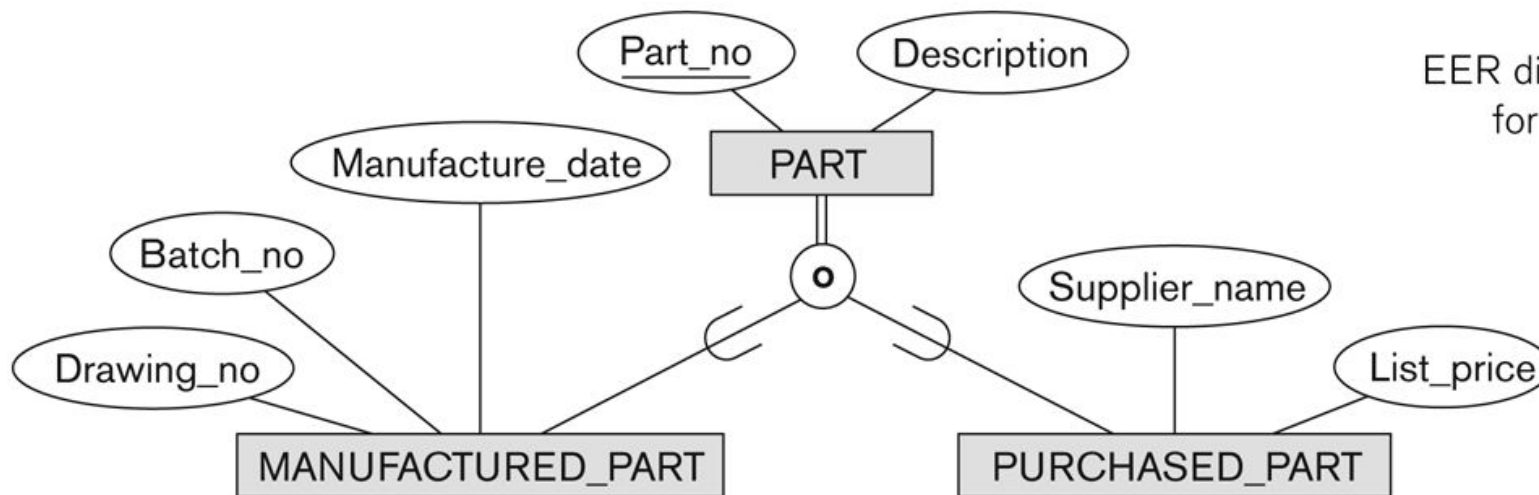


Figure 4.5

EER diagram notation
for an overlapping
(nondisjoint)
specialization.

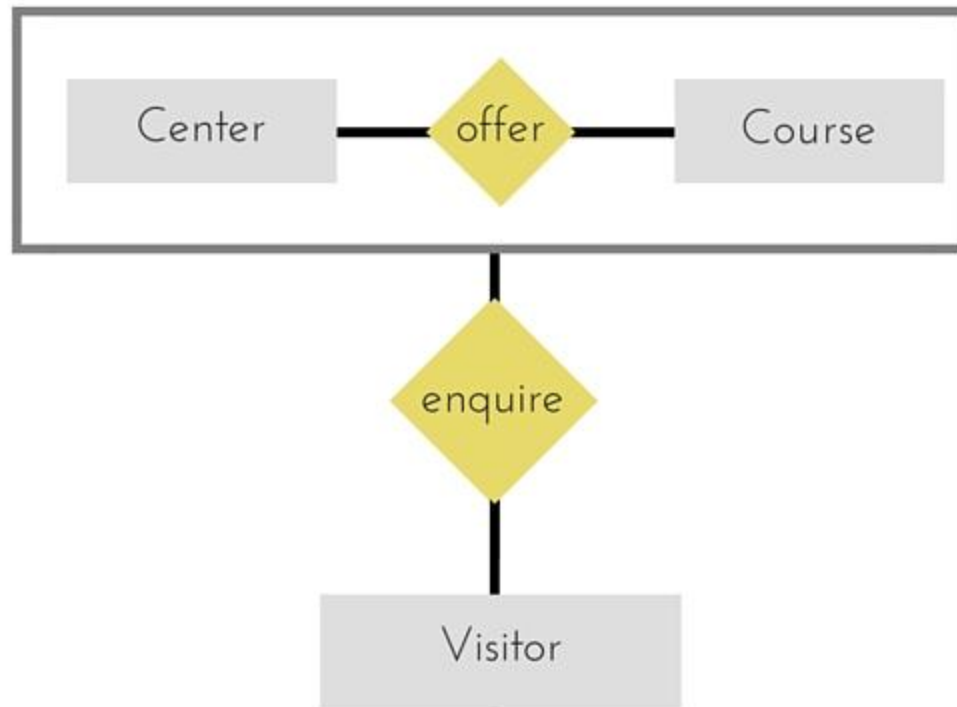
AGGREGATION

- A relationship set is an association between entity sets.
- Sometimes we have to model a relationship between a collection of entities and *relationships*.
- Suppose that we have an entity set called Projects and that each Projects entity is sponsored by one or more departments.
- The Sponsors relationship set captures this information.

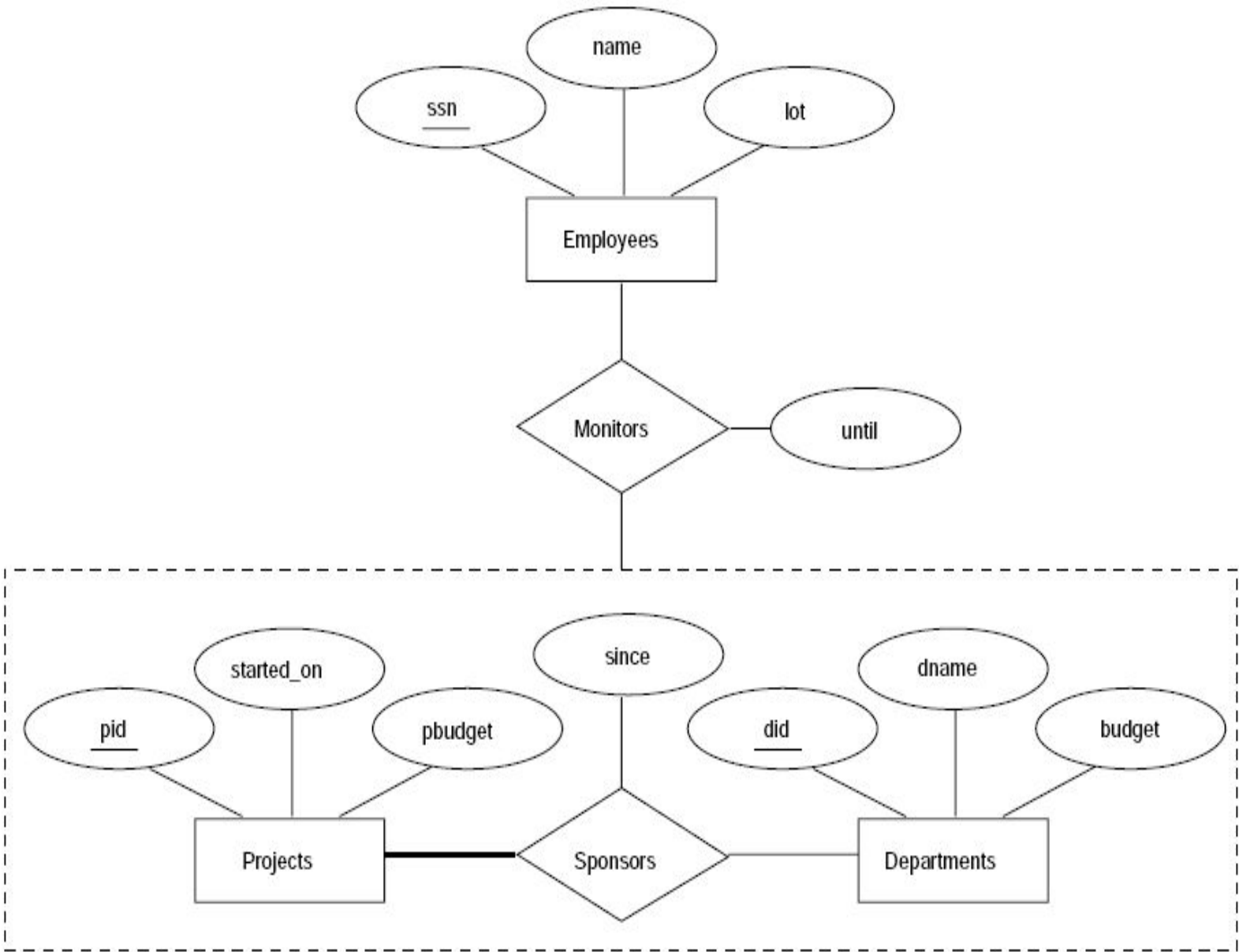


Aggregation

Aggregation is a process when relation between two entities is treated as a **single entity**.



In the diagram above, the relationship between **Center** and **Course** together, is acting as an Entity, which is in relationship with another entity **Visitor**. Now in real world, if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

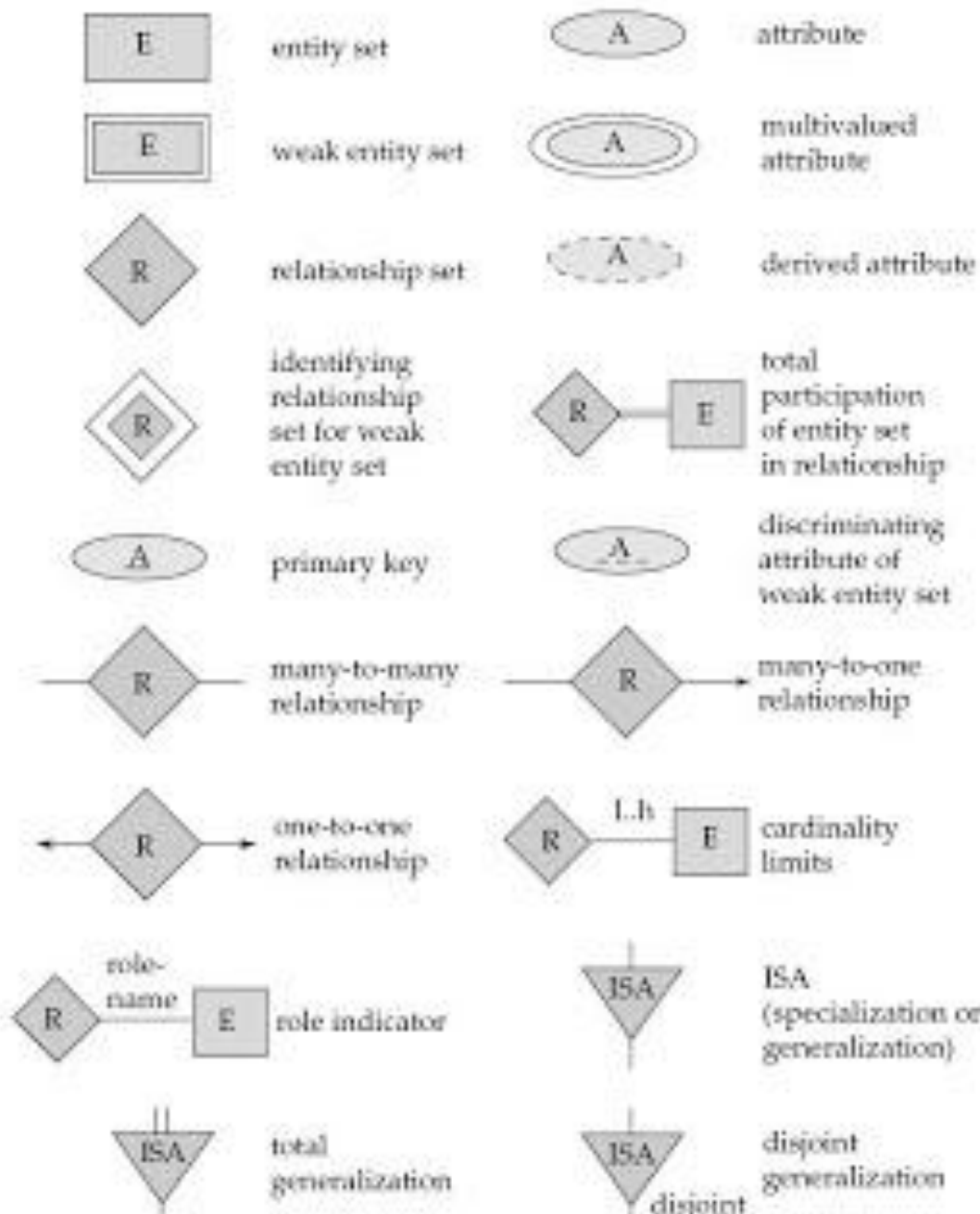


- A department that sponsors a project might assign employees to monitor the sponsorship.
- ie, Monitors should be a relationship set that associates a Sponsors relationship (rather than a Projects or Departments entity) with an Employees entity.
- However, we have defined relationships to associate two or more *entities*.

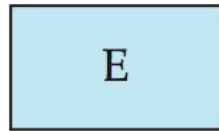


- In order to define a relationship set such as Monitors, we introduce a new feature of the ER model, called *aggregation*.
- A dashed box is used to illustrate aggregation.
- *Aggregation* allows us to indicate that a *relationshipset* (identified through a dashed box) participates in another relationship set.

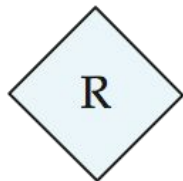




Summary of Symbols Used in E-R Notation



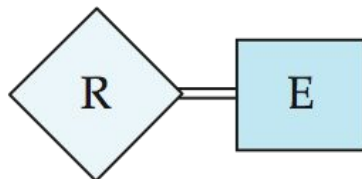
entity set



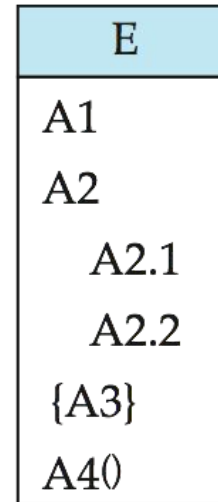
relationship set



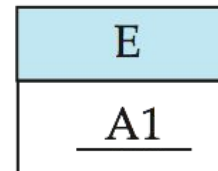
identifying
relationship set
for weak entity set



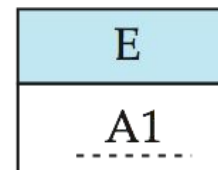
total participation
of entity set in
relationship



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)



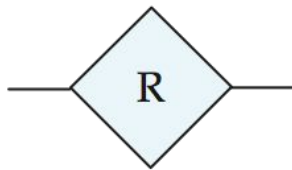
primary key



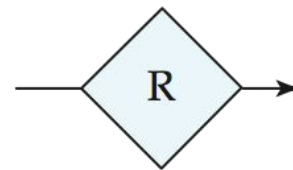
discriminating
attribute of
weak entity set



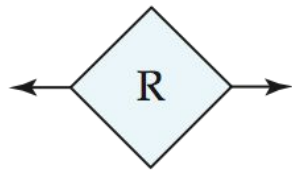
Symbols Used in E-R Notation (Cont.)



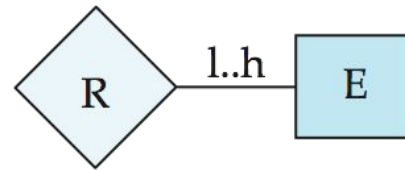
many-to-many
relationship



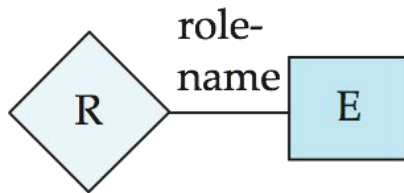
many-to-one
relationship



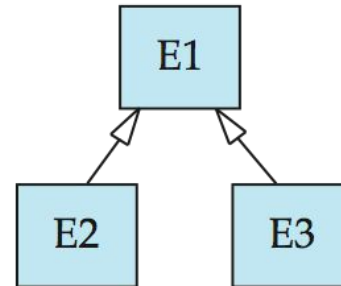
one-to-one
relationship



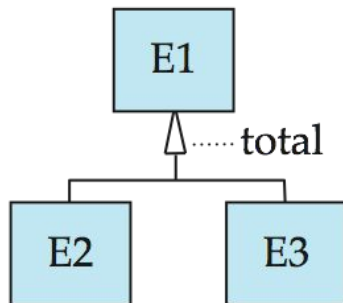
cardinality
limits



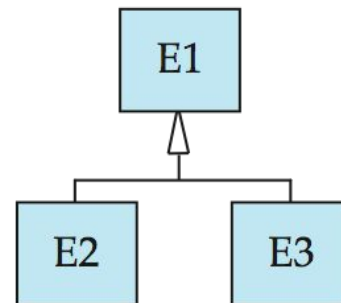
role indicator



ISA: generalization
or specialization



total (disjoint)
generalization



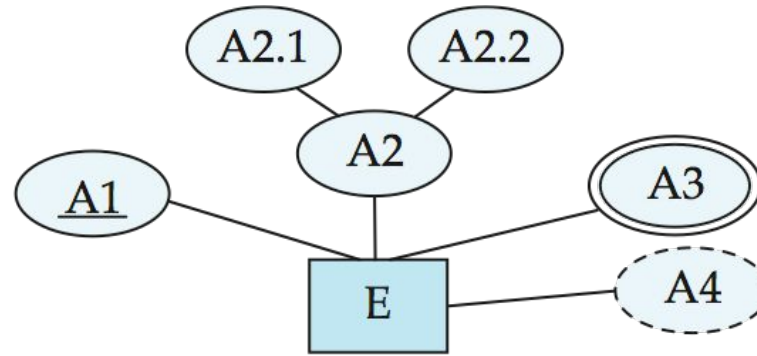
disjoint
generalization



Alternative ER Notations

- Chen, IDE1FX, ...

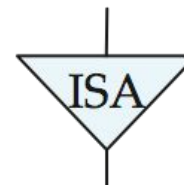
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



weak entity set



generalization



total
generalization

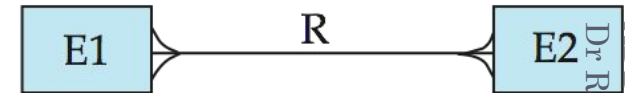
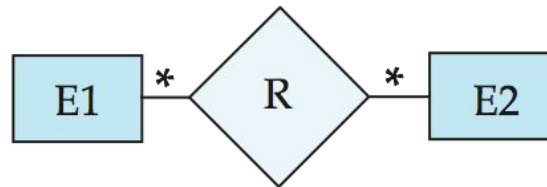


Alternative ER Notations

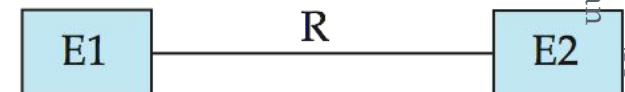
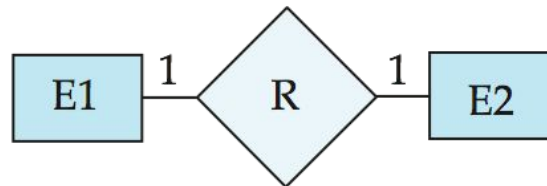
Chen

IDE1FX (Crows foot notation)

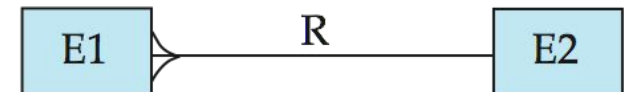
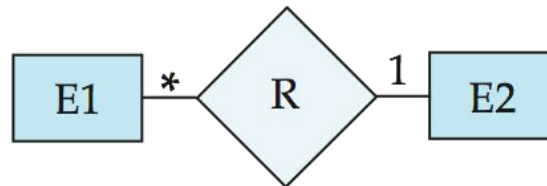
many-to-many
relationship



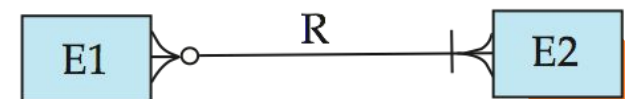
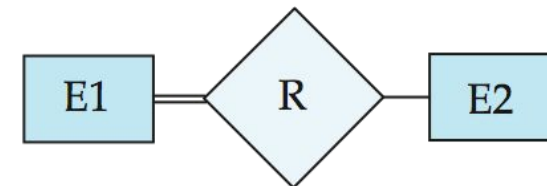
one-to-one
relationship

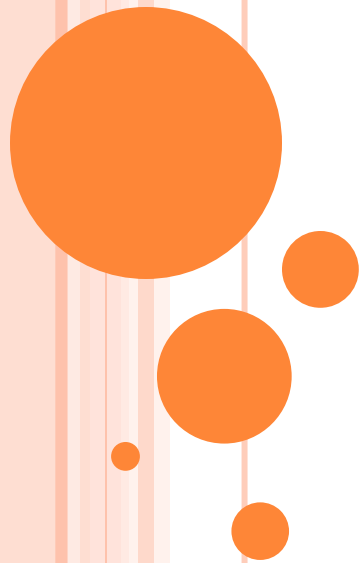


many-to-one
relationship



participation
in R: total (E1)
and partial (E2)



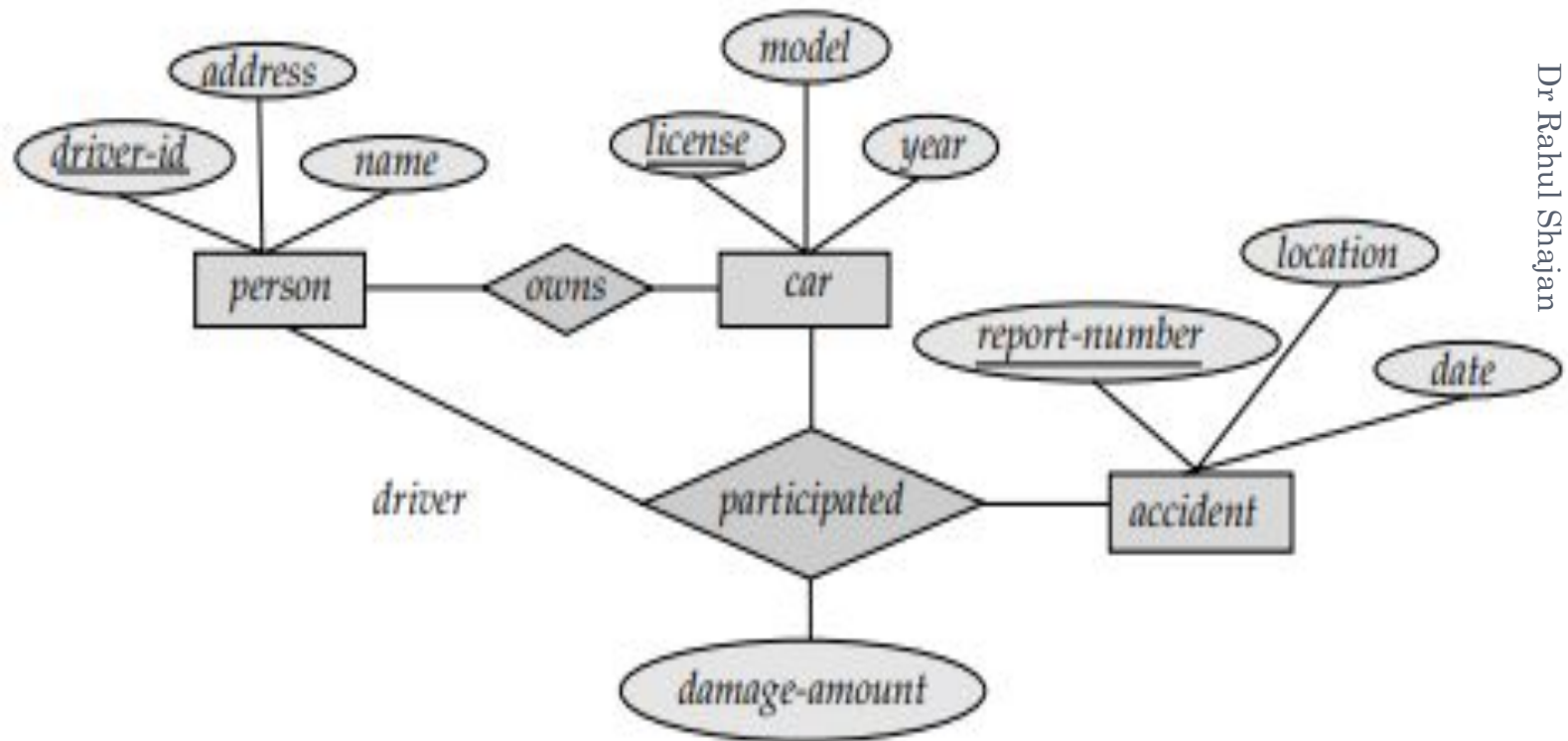


ER diagram

CONSTRUCT AN ER 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.



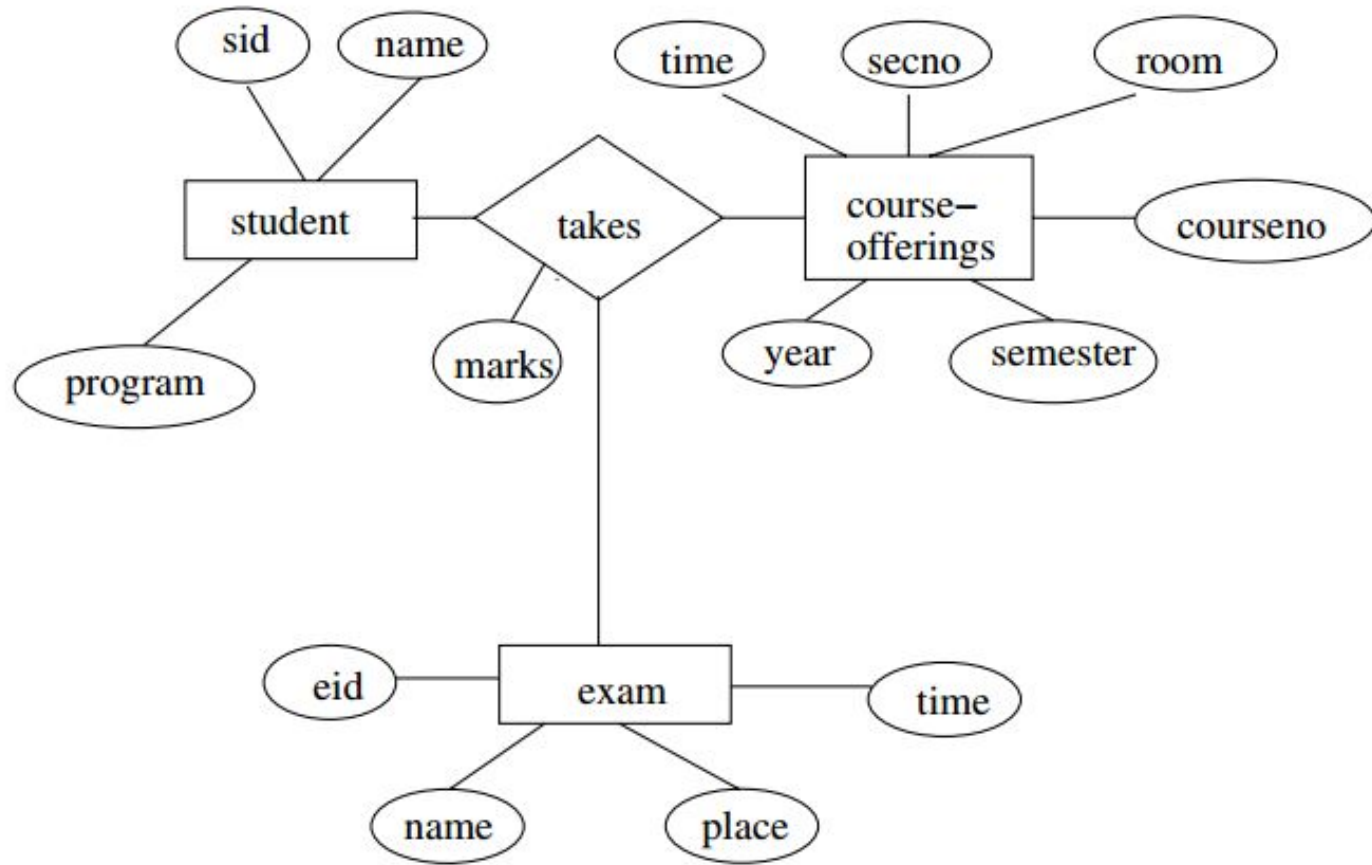
Construct an ER diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

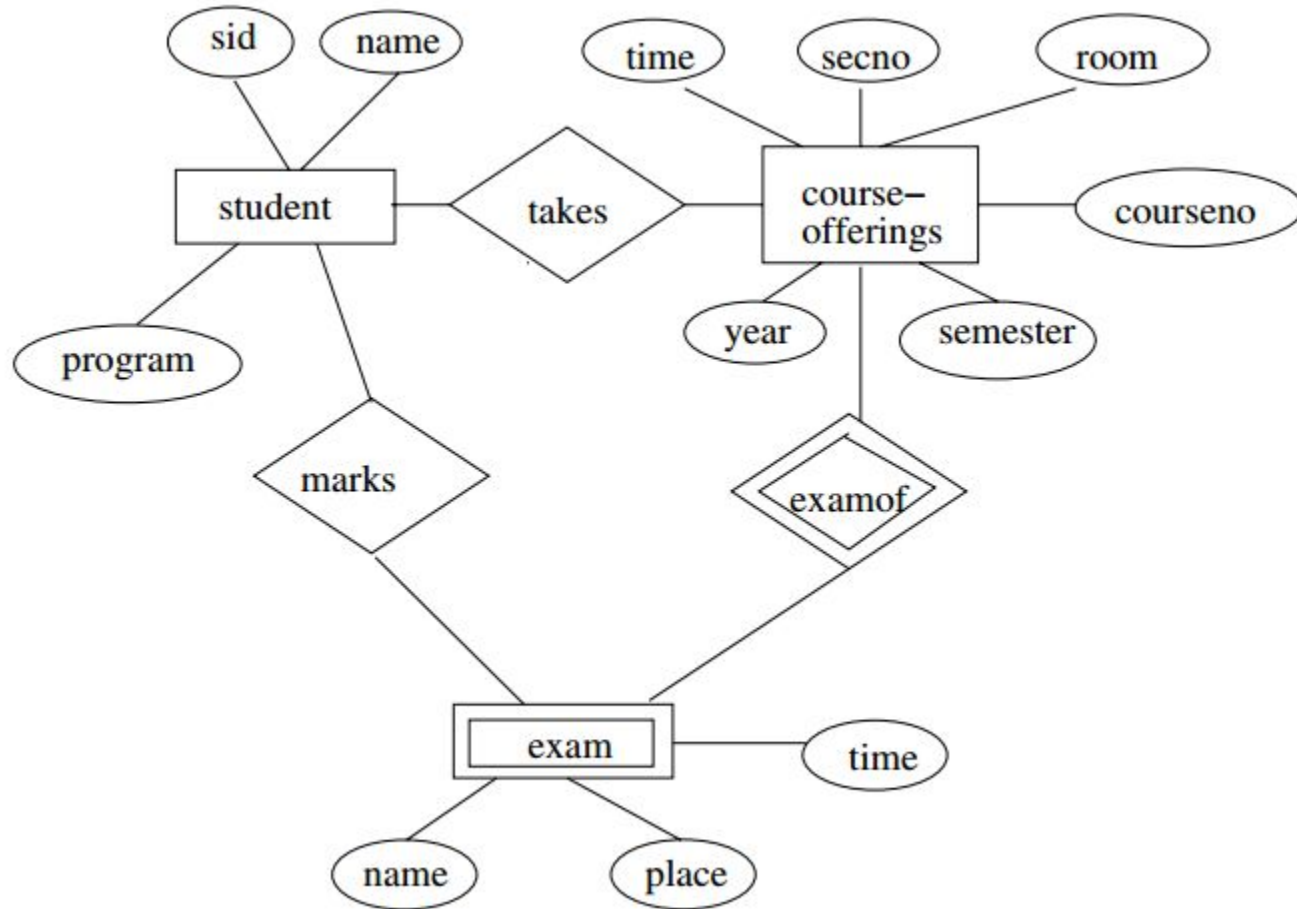


Consider a database used to record the marks that students get in different exams of different course offerings.

- a. Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.
- b. Construct an alternative E-R diagram that uses only a binary relationship between *students* and *course-offerings*. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.

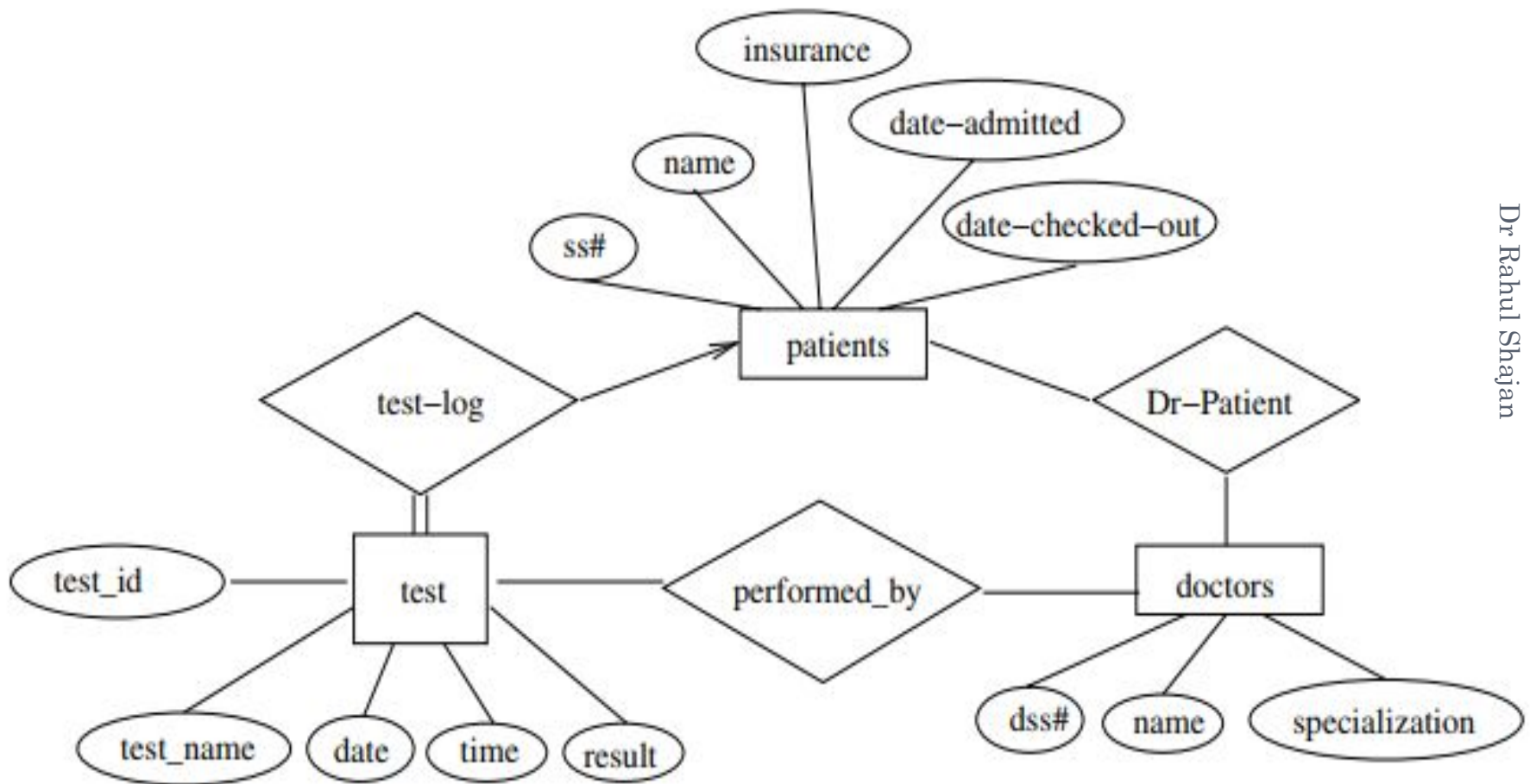






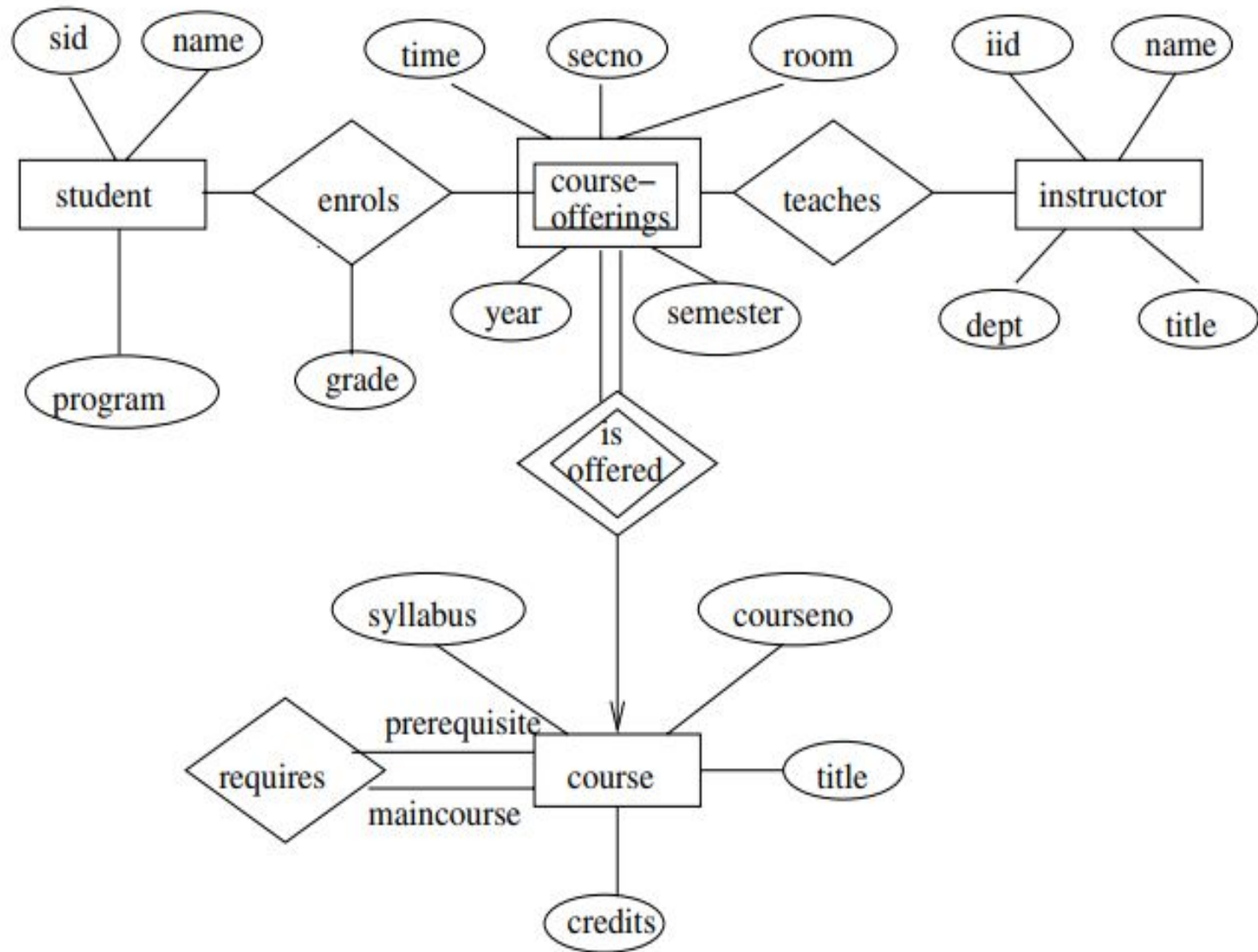
CONSTRUCT AN E-R DIAGRAM FOR A HOSPITAL WITH A SET OF PATIENTS AND A SET OF MEDICAL DOCTORS. ASSOCIATE WITH EACH PATIENT A LOG OF THE VARIOUS TESTS AND EXAMINATIONS CONDUCTED.





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 for the registrar's office. Document all assumptions that you make about the mapping constraints.





- A company database needs to store information about employees(identified by ssn with salary and phone attributes),departments(identified with dno with dname and budget as attributes) and children of employees(name,age).employee works in department.each department is *managed by* an employee.a child must identified uniquely by *name* when parent(who is an employee) assume that only one parent works for the company) is known



