ADBMS

UNIT-II : RELATIONAL DATA STRUCTURE: RELATION – DOMAINS & ATTRIBUTES – KEYS – SQL DATA TYPES – **E-R MODEL:** ENTITIES, ENTITY SETS – RELATIONSHIPS AND MAPPING CARDINALITIES, RELATIONSHIP SETS – ER DIAGRAM NOTATIONS – PARTICIPATION CONSTRAINTS - EXTENDED E-R FEATURES – RULES FOR TRANSFORMING ER DIAGRAM INTO TABLES – **DATA DEFINITION LANGUAGE** : CREATE, ALTER AND DROP TABLES –**NORMALIZATION** : NEED, NORMALIZATION PROCESS - **NORMAL FORMS :** 1NF, 2NF AND 3 NF – DENORMALIZATION. **TRANSACTION CONTROL LANGUAGE (TCL) COMMANDS:** COMMIT, SAVEPOINT, ROLLBACK - **DATABASE ADMINISTRATION:** DBA TASKS – USER PRIVILEGES

RELATIONAL DATA STRUCTURE – RELATION

- A relation is a two-dimensional table in which the rows of the table are individual records and the table columns are attributes of the records.
- In the relational model , relations are used to hold information about the objects (attributes) to be represented in the database.
- **Relation schema** is a named relation defined by a set of attribute and domain name pairs.

Properties of Relations

1. The relation has a distinct name in the relational schema.
2. Each cell of the relation contains exactly one value.
3. Each attribute has a distinct name.
4. The values of an attribute are all from the same domain.
5. Each tuple is distinct; there are no duplicate tuples.
6. The order of attributes has no significance.
7. The order of tuples has no significance.

DOMAIN

- A domain is the set of allowable values for one or more attributes.
- Every attribute in a relation is defined on a domain.
- Domains may be distinct for each attribute, or two or more attributes may be defined on the same domain.
- At any given time, typically there will be values in a domain that do not currently appear as values in the corresponding attribute.
- The domain concept is important because it allows the user to define in a central place the meaning and source of values that attributes can hold.
- Domain definition gives more information to the system when it undertakes the execution of a relational operation, and operations that are semantically incorrect can be avoided.

ATTRIBUTE

- An attribute is a named column of a relation, used to hold information about the objects to be represented in the database.
- The order of attributes has no significance.
- A column contains values of a single attribute.
- There are several types of Attributes
- Attributes are **Descriptive Properties possessed by each member of an entity set.**
- **For each attribute, there is a set of permitted values, called the Domain.**
- Formally, **an attribute of an entity set is a function that maps from the entity set into a domain**.
- **The attribute values** describing an entity **will constitute a significant portion of the data stored in the database.**
- **AN ATTRIBUTE TAKES A NULL VALUE WHEN AN ENTITY DOES NOT HAVE A VALUE FOR IT.**
  - Example: one may have no middle name.
- ***NULL CAN ALSO ASSIGNED WHEN AN ATTRIBUTE VALUE IS UNKNOWN. AN UNKNOWN VALUE MAY BE EITHER MISSING OR NOT KNOWN.***

ATTRIBUTE TYPES

- **SIMPLE AND COMPOSITE ATTRIBUTES**
  - ***SIMPLE ATTRIBUTES*** **cannot be divided into subparts**.).
  - ***COMPOSITE ATTRIBUTES*** help us **to group together related attributes**, making the modeling cleaner. Composite attribute may appear as a hierarchy.

    Example: In Composite attribute **address, its component attribute street can be further divided into streetnumber, streetname, and apartmentnumber**

- **DERIVED ATTRIBUTE**. The **value for this type of attribute can be derived from the values of other related attributes or entities**.

  Example : If the customer entity set has an attribute ***date...of..birth***, **we can calculate *age* from date...of..birth and the current date**. Thus, **age is a Derived Attribute**.

- **SINGLE-VALUED AND MULTIVALUED ATTRIBUTES** :
  - Attributes with single value are ***Single-valued attributes***.
  - **Attributes that has multiple values** are called ***Multi-valued attributes*** **.**

    Example : Phone Number

OTHER IMPORTANT TERMS TO REMEMBER

Tuple

- A tuple is a row of a relation.
- Each tuple is a record of details about an entity or object.
- The elements of a relation are the rows or tuples in the table.
- The order of tuples has no significance

Degree

- The ***degree* of a relation is the number of attributes it contains.**
- A relation with ***only one attribute*** would have *degree one* and be called a **UNARY RELATION**.
- A relation with **two attributes** is called **binary**, one with **three attributes** is called **ternary**, and after that the term **n-ary** is usually used.

Cardinality

- The **cardinality of a relation is the number of tuples it contains**.
- The *cardinality changes, as tuples are added or deleted.*

OTHER IMPORTANT TERMS TO REMEMBER

INTENSION

- **The structure of a relation, together with a specification of the domains and any other restrictions on possible values, is called its INTENSION.**
- **Intension is usually fixed unless the meaning of a relation is changed to include additional attributes.**
- The **Degree of A Relation** is a property of the intension of the relation
    - ***Degree of a relation is the number of attributes it contains.***

EXTENSION

- **The tuples are called the EXTENSION (or state) of a relation, which changes over time.**
- The **Cardinality** is a property of the extension of the relation and is determined from the particular instance of the relation at any given moment.

KEYS

- The **values of the Key attributes of an entity** must be such that they **can *uniquely identify* the entity**
- Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

PRIMARY KEY

- A primary is **a column or set of columns in a table that uniquely identifies tuples** (rows) in that table.
- The **value of primary key should be unique for each row** of the table.
- The column(s) that makes the **key cannot contain duplicate values**.
- The attribute(s) that is marked **as primary key is not allowed to have null values**.

Example :

Consider the table **STUDENT(Stu_ID, Stu_Name, Stu_Age)**

Attribute **Stu_Id alone is a primary key** as each student has a unique id that can identify the student record in the table.

PRIMARY KEY

- **Primary keys are not necessarily to be a single attribute (column). It can be a set of more than one attributes (columns).** We should choose more than one columns as primary key **only when there is no single column that can uniquely identify the tuple in table.**

Example :

Consider the table **ORDER(Customer_ID, Product_ID , Order_Quantity)**

Customer_ID alone cannot be a primary key as a single customer can place more than one order

Product_ID alone cannot be a primary key as more than one customers can place a order for the same product .

Order_Quantity alone cannot be a primary key as more than one customers can place the order for the same quantity.

Since none of the attributes alone were able to become a primary key, makes us try to make a set of attributes that plays the role of it.

{Customer_ID, Product_ID} together can identify the rows uniquely in the table so this set is the primary key for this table.

SUPER KEY

- A super key **is a set of one of more columns (attributes) to uniquely identify rows in a table.**
- ***A superkey may contain unrelated attributes***.

  Example: **the customer_id attribute of the entity set customer is sufficient to distinguish one customer entity from another. Thus, customer_id is a superkey. Similarly, the combination of customer_name and customer_id is a superkey for the entity set customer.**

CANDIDATE KEY

- A primary key is being selected from the group of candidate keys.
- A candidate key does not allow null values and have unique values.
- **A super key with no redundant attribute is known as candidate key**
- **Candidate keys are selected from the set of super keys,** the only thing we take care while selecting **candidate key is: It should not have any redundant attribute. Hence, they are also called as *Minimal Super Key*.**

SECONDARY KEY – Out of all candidate keys, only one gets selected as primary key, remaining keys are known as secondary or alternate keys.

COMPOSITE KEY – **A key that consists of more than one attribute to uniquely identify rows** (also known as records & tuples) in a table is called composite key.

Example: **Sales(cust_Id , order_Id , product_code, product_count)**

Key in above table: {cust_id, order_id}

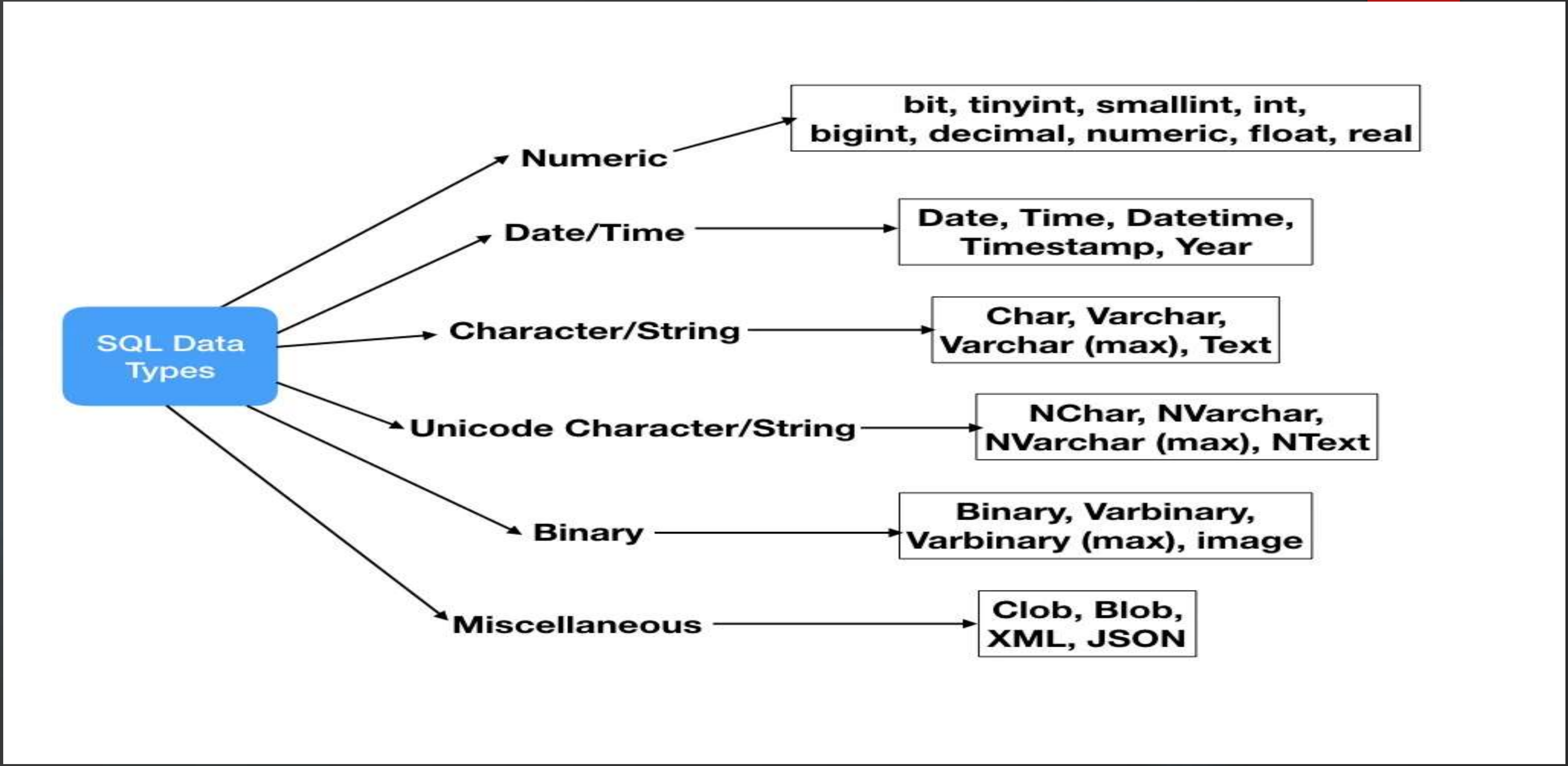
FOREIGN KEY – Foreign keys are the **columns of a table that points to the primary key of another table. They act as a cross-reference between tables.**

- **Foreign key may also point to a unique column (*NOT NECESSARILY A PRIMARY KEY*) of another table.**

BASIC DOMAIN TYPES

1. **char(n)**. Fixed length character string, with user-specified length *n.*
2. **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
3. **int.** Integer (a finite subset of the integers that is machine-dependent).
4. **smallint.** Small integer (a machine-dependent subset of the integer domain type).
5. **numeric(p, d) .** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point.
6. **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
7. **float(n).** Floating point number, with user-specified precision of at least *n* digits.
8. **date**: A calendar date containing a (four-digit) year month, and day of the month.
9. **time**: The time of day, in hours, minutes, and seconds.
10. **timestamp**: A combination of date and time.

SQL DATA TYPES


SQL Data Types
Numeric
bit, tinyint, smallint, int, bigint, decimal, numeric, float, real
Date/Time
Date, Time, Datetime, Timestamp, Year
Character/String
Char, Varchar, Varchar (max), Text
Unicode Character/String
NChar, NVarchar, NVarchar (max), NText
Binary
Binary, Varbinary, Varbinary (max), image
Miscellaneous
Clob, Blob, XML, JSON

ENTITY–RELATIONSHIP MODEL (ER model)

- An ER model is a **DESIGN OR BLUEPRINT OF A DATABASE** that can later be implemented as a database.
- The ER model **DEFINES THE CONCEPTUAL *(abstract / theoretical)* VIEW OF A DATABASE**. It works around real-world entities and the associations among them. At view level, the ER model is considered a **GOOD OPTION FOR DESIGNING DATABASES**.
- The entity-relationship (E-R) data model is **developed to facilitate database design by allowing specification of an *enterprise schema (plan)* that represents the overall logical structure of a database.**
- An Entity–relationship model (ER model) **describes the structure of a database with the help of Entity Relationship Diagram (ER Diagram).**

ENTITY RELATIONSHIP DIAGRAM (ER Diagram)

- An ER diagram **shows the relationship among entity sets.**
- We use ER diagrams where **we plan the database pictorially**.
- ER diagram basically **breaks requirement into entities, attributes and relationship.**

ENTITY

- An entity is **a "thing" or "object" in the real world that is distinguishable** from all other objects.
- An entity is **represented by a set of attributes (properties) , and the values for some set of attributes may uniquely identify an entity.**
- An entity may be **concrete, such as a person** or a book, or **abstract, such as a loan**, a holiday, or a concept
- The **function that an entity plays in a relationship is called that *ENTITY'S ROLE***. Since **entity sets are** participating in a relationship set that **are generally distinct**, **roles are implicit and are not usually specified.**

ENTITY SET

- An entity set is a **set of entities of the same type that share the same properties or attributes**. However, each entity may **have its own value for each attribute**.
- Entity sets **need not be disjoint**. ***(two sets are said to be Disjoint sets if they have no element in common)***

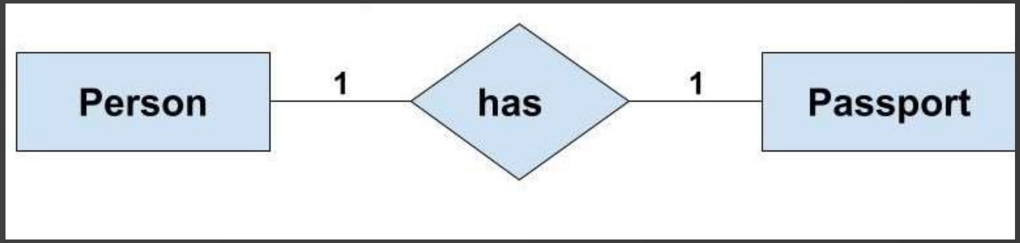
RELATIONSHIPS:

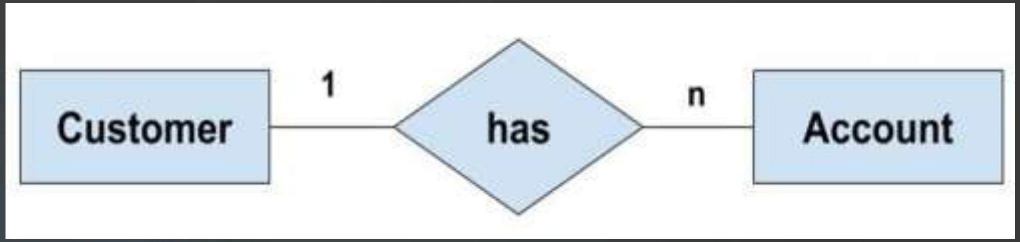
- The **association among entities is called a relationship**.
- The **association between entity sets is referred to as PARTICIPATION**

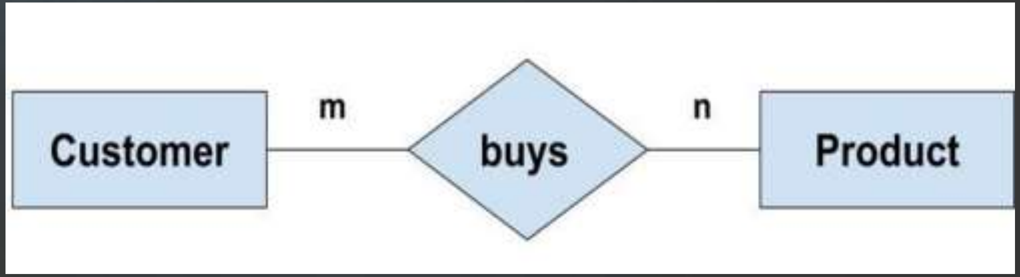
  For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

- The **function that an entity plays in a relationship is called that ENTITY'S ROLE.**
  - **Since entity sets participating in a relationship set are generally distinct, roles are implicit and are not usually specified.**

RELATIONSHIP TYPES:







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 (zero or more) 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. An entity in B, however, can be associated with any number (zero or more) of entities in A.

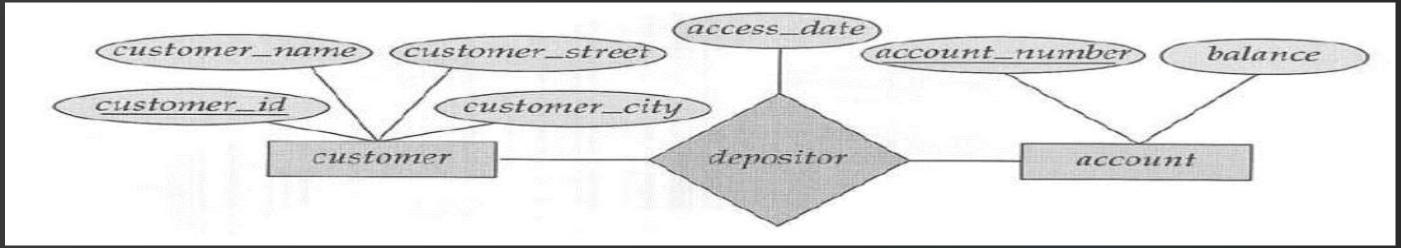
MANY-TO-MANY. An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

DEGREE OF RELATIONSHIP: The number of participating entities in a relationship defines the degree of the relationship.

- For example, Binary = degree 2 ; Ternary = degree 3

RELATIONSHIP SET : A set of relationships of similar type is called a relationship set.

- **A relationship too can have attributes. These attributes are called descriptive attributes.**
- *Example* : Consider a relationship set depositor with entity sets customer and account. We could associate the attribute access_date to that relationship to specify the most recent date on which a customer accessed an account.



MAPPING CARDINALITIES

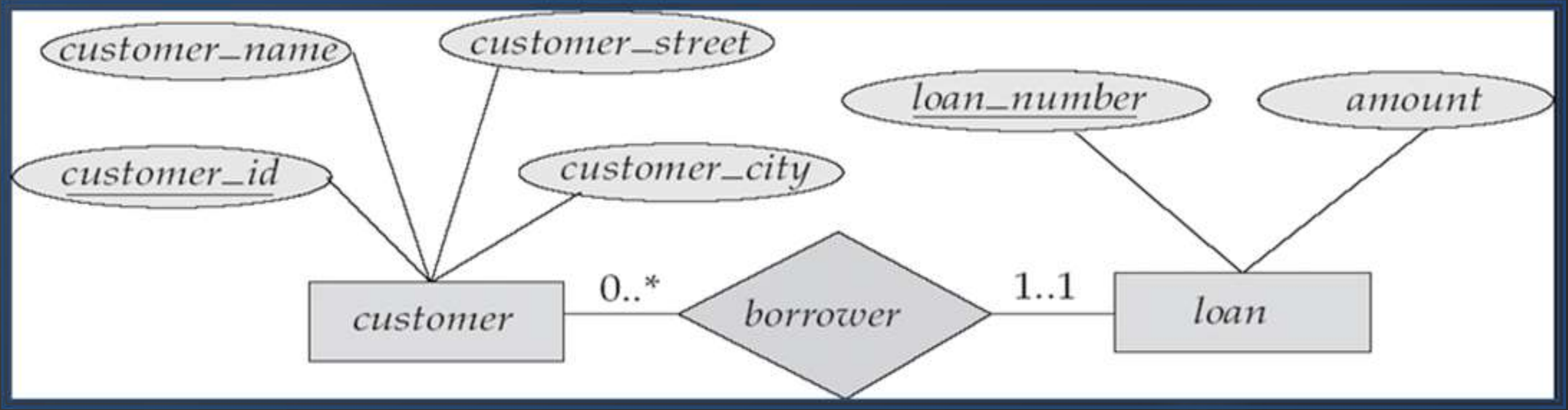
- Mapping cardinalities, or cardinality ratios expresses the number of entities to which another entity can be associated via a relationship set.
- Mapping cardinalities are most useful in describing binary relationship sets.

The appropriate mapping cardinality for a particular relationship set depends on the real-world situation that the relationship set is modeling .

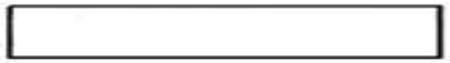
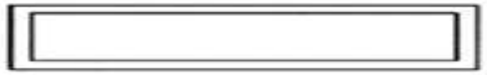
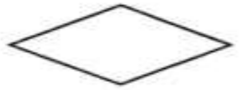
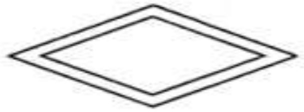
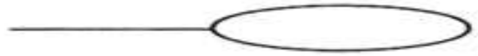
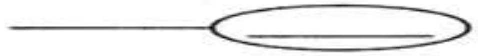
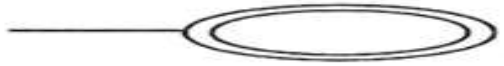
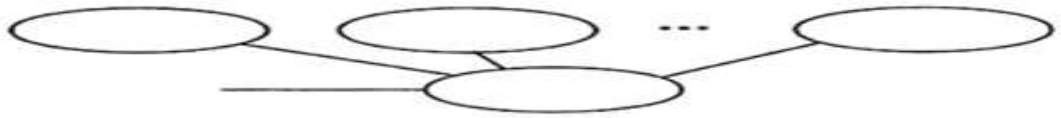
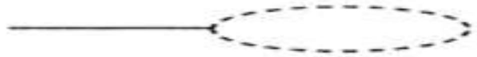
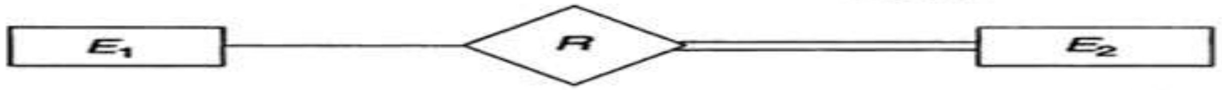
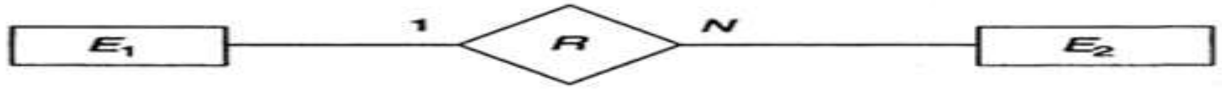
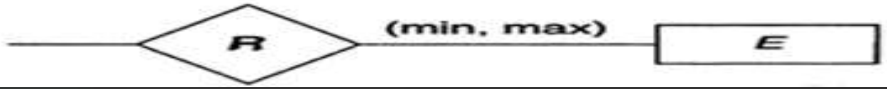
Representation of Mapping Cardinalities in E- R Diagram

E-R diagrams also provide a way to indicate more complex constraints on the number of times each entity participates in relationships in a relationship set.

- ***An edge between an entity set and a binary relationship set can have an associated minimum and maximum cardinality, shown in the form l..h, where l is the minimum and h the maximum cardinality.***
- ***A minimum value of 1 indicates total participation of the entity set in the relationship set.***
- ***A maximum value of 1 indicates that the entity participates in at most one relationship, while***
- ***A maximum value * indicates no limit.***

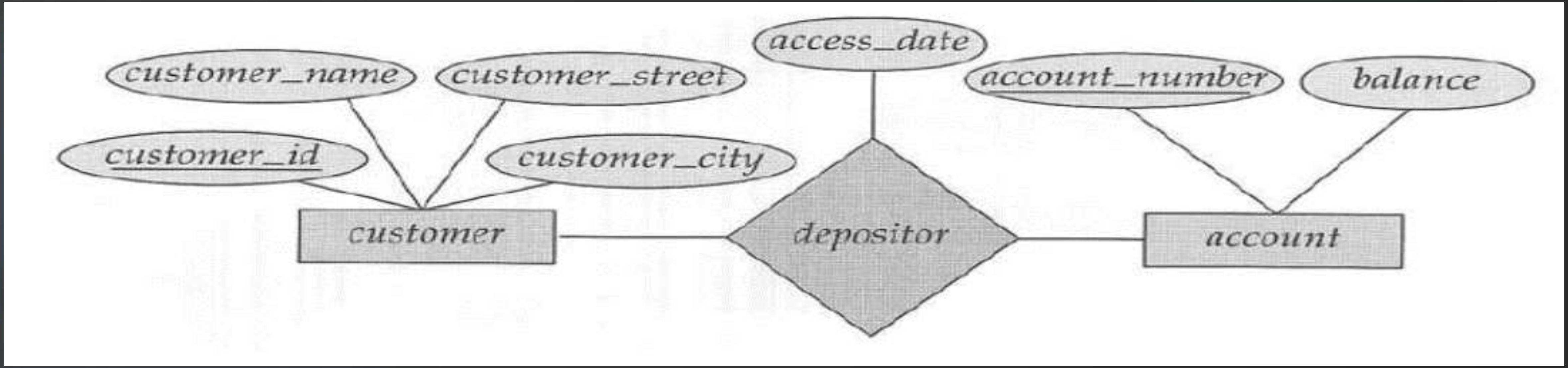


ER DIAGRAM NOTATIONS

Symbol	Meaning
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E_2 IN R
	CARDINALITY RATIO 1: N FOR E_1:E_2 IN R
	STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

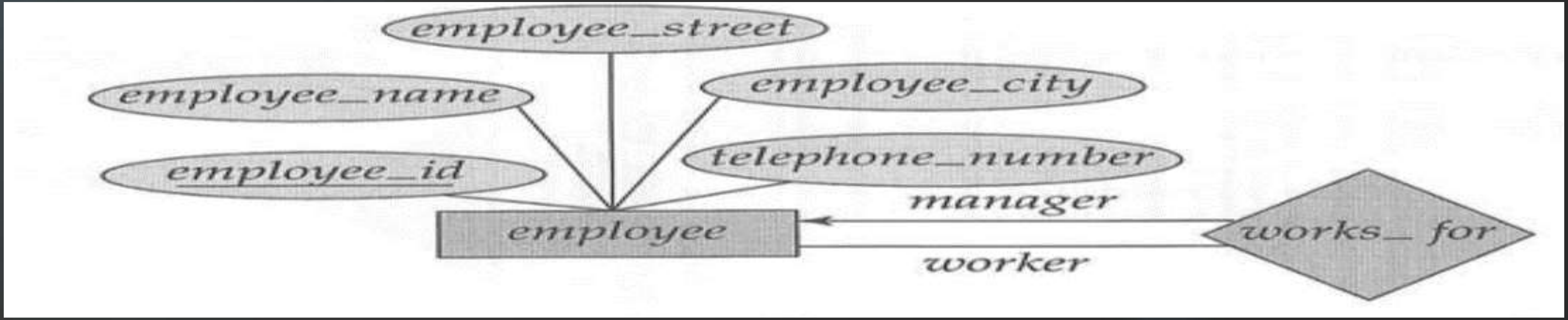
Relationship Instance

- **Relationship Instance in a given relationship set must be uniquely identifiable from its participating entities, without using the descriptive attributes.**
  - **A relationship instance in an E-R schema represents an association between the named entities in the real-world that is being modelled.**



Recursive Relationship Set

- **When the same entity set participates in a relationship set more than once, in different roles, then it is called as a RECURSIVE RELATIONSHIP SET, *explicit role names are necessary* to specify how an entity participates in a relationship instance.**
  - **Example : Consider an entity set employee that records information about all the employees of the bank. We may have a relationship set works_for that is modelled by ordered pairs of employee entities. The first employee of a pair takes the role of worker, whereas the second takes the role of manager. In this way, all relationships of works-for are characterized by (worker, manager) pairs; (manager, worker) pairs are excluded**



Participation Constraints

- The participation of an entity set E in a relationship set R is said to be **TOTAL PARTICIPATION *if every entity in E participates in at least one relationship in R.***
- ***If only some entities in E participate in relationships in R*, the participation of entity set E in relationship R is said to be Partial Participation.**

Example

We expect every loan entity to be related to at least one customer through the borrower relationship. Therefore **the participation of loan in the relationship set borrower is total.**

In contrast, **an individual can be a bank customer whether or not she has a loan with the bank. Hence, it is possible that only some of the customer entities are related to the loan entity set through the borrower relationship, and the participation of customer in the borrower relationship set is therefore partial**

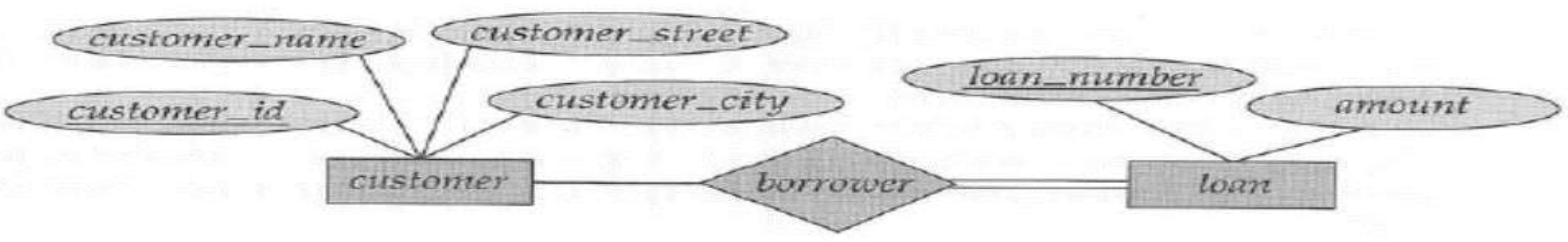


Figure 6.13 Total participation of an entity set in a relationship set.

RELATIONSHIP SET – PRIMARY KEY

THE PRIMARY KEY OF A RELATIONSHIP SET DEPENDS ON THE MAPPING CARDINALITY AND THE MEANING OF ANY ATTRIBUTES OF THE RELATIONSHIP SET.

Case 1: THE RELATIONSHIP SET HAS NO ATTRIBUTES

1. **one-to-one mapping cardinality: the primary key of any of the entity sets involved is a candidate key for the relationship set, because any entity (from any entity set) can be involved in at most one relationship.**
2. **one-to-many mapping cardinality: the primary key of the relationship set is the primary key of the second entity set (the to-many set), because an entity in the second entity set can be involved in at most one relationship.**
3. **many-to-one mapping cardinality: the primary key of the relationship set is the primary key of the first entity set (the to-many set), because an entity in the first entity set can be involved in at most one relationship.**
4. **many-to-many mapping cardinality: the primary key of the relationship set includes the PRIMARY KEYS OF ALL OF THE ENTITY SETS INVOLVED.**

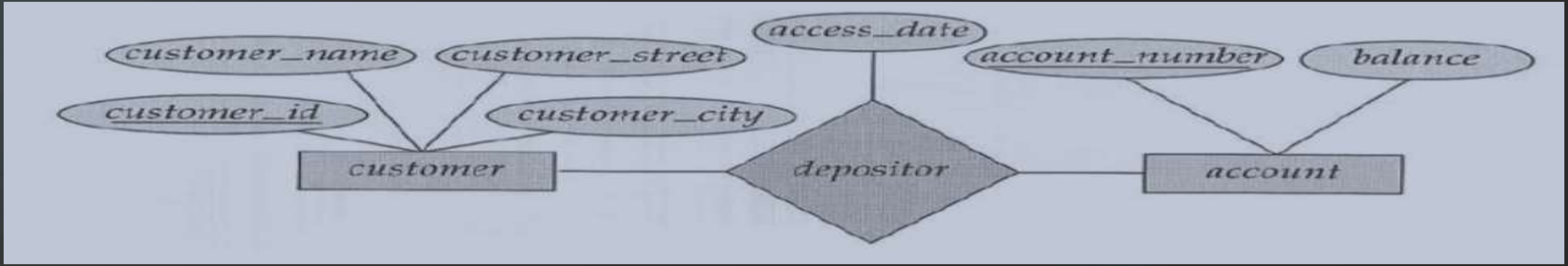
RELATIONSHIP SET – PRIMARY KEY

Case 2: THE RELATIONSHIP SET HAS ATTRIBUTES

- **primary keys of entity sets are included in the primary key of the relationship set exactly as in Case 1.**
- **ATTRIBUTES OF THE RELATIONSHIP SET ARE PART OF THE PRIMARY KEY IF THEY ARE NEEDED TO DISTINGUISH BETWEEN RELATIONSHIPS.**

Example : Consider a relationship set depositor with entity sets customer and account. We could associate the attribute access_date to that relationship to specify the most recent date on which a customer accessed an account.

Note that this attribute access_date cannot instead be placed in either entity set as it relates to both a customer and an account, and the relationship is many-to-many.



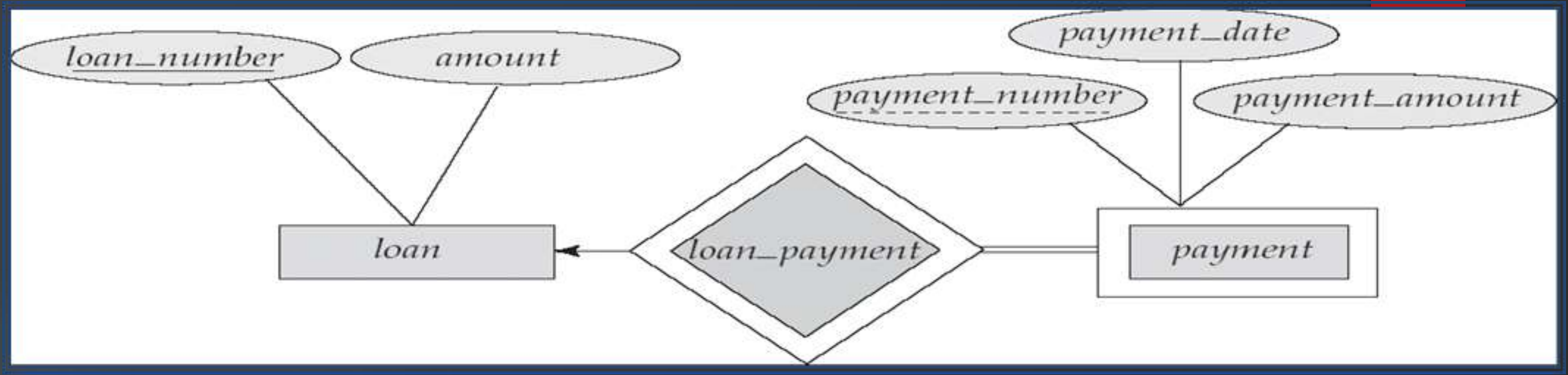
WEAK ENTITY SETS

- ***AN ENTITY SET THAT DOES NOT HAVE ATTRIBUTES TO FORM PRIMARY KEY IS A WEAK ENTITY SET.***
- ***The existence of a weak entity set depends on the existence of an identifying entity set***
- ***The Identifying Entity Set is said to OWN the Weak Entity Set that it identifies.***
- **The relationship associating the weak entity set with the identifying entity set is called the *IDENTIFYING RELATIONSHIP.***
  - ***The Identifying Relationship is MANY-TO-ONE FROM THE WEAK ENTITY SET TO THE IDENTIFYING ENTITY SET, AND THE PARTICIPATION OF THE WEAK ENTITY SET IN THE RELATIONSHIP IS TOTAL.***
- ***We depict a Weak Entity Set by DOUBLE RECTANGLES* . Identifying *relationship depicted using a DOUBLE DIAMONDS*.**
- **The identifying relationship set should have no descriptive attributes, since any required attributes can be associated with the weak entity set.**
- **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.***

WEAK ENTITY SETS

- **THE PRIMARY KEY OF A WEAK ENTITY SET IS FORMED BY THE PRIMARY KEY OF THE STRONG ENTITY SET ON WHICH THE WEAK ENTITY SET IS EXISTENCE DEPENDENT, PLUS THE WEAK ENTITY SET'S DISCRIMINATOR.**
- ***We may represent a weak entity set as a multivalued composite attribute of the owner entity set.***
- **A weak entity set may participate as owner in an identifying relationship with another weak entity set.**
- **A weak entity set can have more than one identifying entity set. A particular weak entity would then be identified by a combination of entities, one from each identifying entity set.**
- ***THE PRIMARY KEY OF THE WEAK ENTITY SET WOULD CONSIST OF THE UNION OF THE PRIMARY KEYS OF THE IDENTIFYING ENTITY SETS, PLUS THE DISCRIMINATOR OF THE WEAK ENTITY SET.***
- We underline the discriminator of a weak entity set with a D*ASHED LINE.*
  - ***payment_number* – *<u>Discriminator</u>* of the *<u>payment entity set</u>***
  - ***(loan_number, payment_number)* - Primary key for *<u>payment entity set</u>***

WEAK ENTITY SETS



Note:
The primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

EXTENDED E-R FEATURES (*Specialization & Generalization*)

SPECIALIZATION

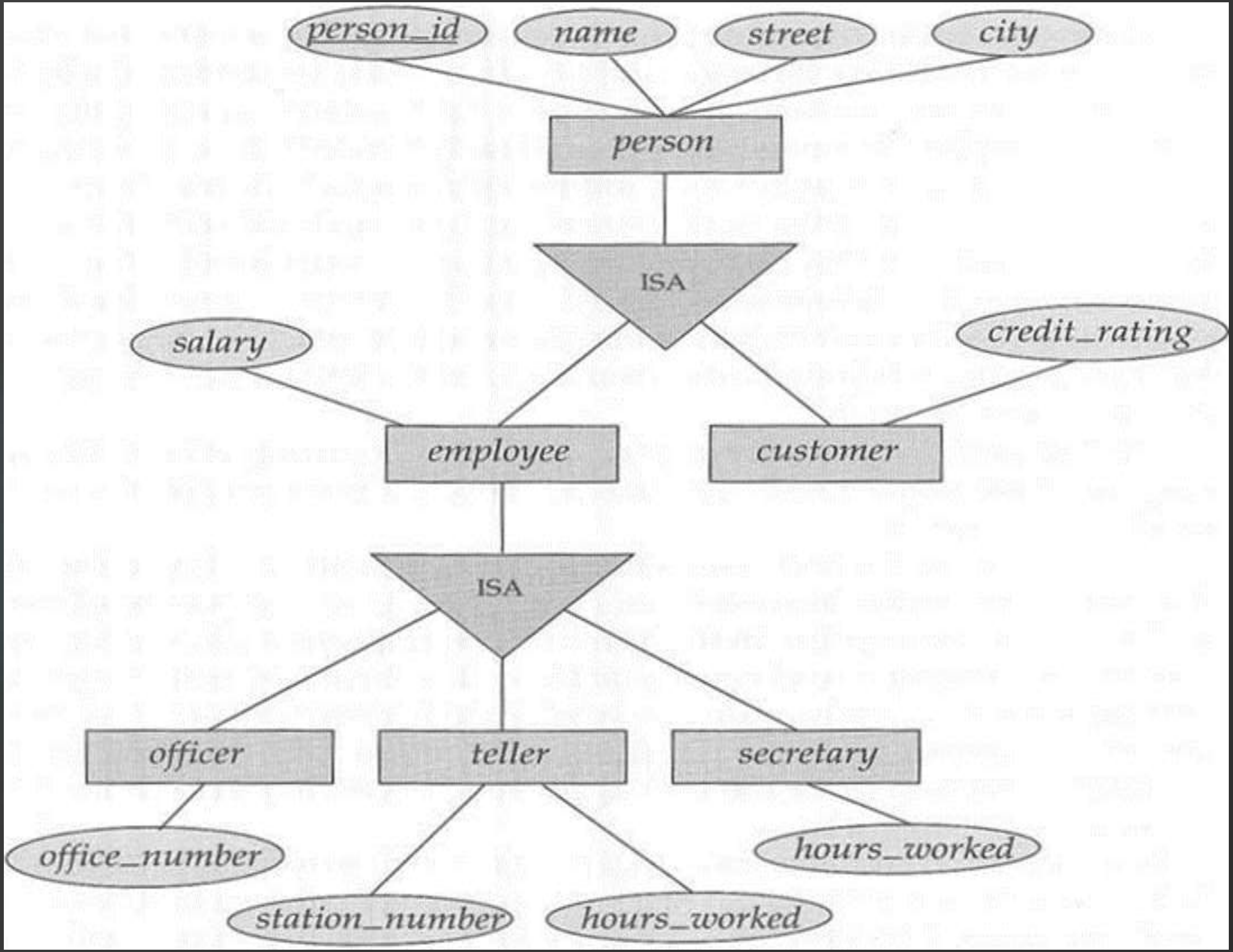
- The process of designating subgroupings within an entity set is called *Specialization*.
- A subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
- Specialization is the refinement from an initial entity set into successive levels of entity subsets. It represents a top-down design process in which distinctions are made explicit.
- Specialization starts from a single entity set, it emphasizes differences among entities within the set by creating distinct lower-level entity sets. These lower-level entity sets may have attributes, or may participate in relationships, that do not apply to all the entities in the higher-level entity set.
  - A designer applies specialization to represent distinctive features.

EXTENDED E-R FEATURES (*Specialization & Generalization*)

GENERALIZATION

- The design process of entity sets may proceed in bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.
- Higher- and lower-level entity sets also may be designated by the terms superclass and subclass, respectively. For all practical purposes, GENERALIZATION IS A SIMPLE INVERSION OF SPECIALIZATION
- Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences; it also permits an economy of representation in that shared attributes are not repeated.
- Generalization proceeds from the recognition that a number of entity sets share some common features (namely, they are described by the same attributes and participate in the same relationship sets). On the basis of their commonalities, generalization synthesizes these entity sets into a single, higher-level entity set.

EXTENDED E-R FEATURES (*Specialization & Generalization*)



In terms of an E-R diagram,

- *Specialization is depicted by a triangle component labeled ISA* .
  - **The label ISA stands for "is a" and represents a superclass-subclass relationship**.
  - Higher- and lower-level entity sets are depicted as regular entity sets-that is, as rectangles containing the name of the entity set.
- **Specialization is moving from Higher level Entity set to Lower Level Entity set** and
- **Generalization is moving from Lower level Entity set to Higher Level Entity set**.

ATTRIBUTE INHERITANCE

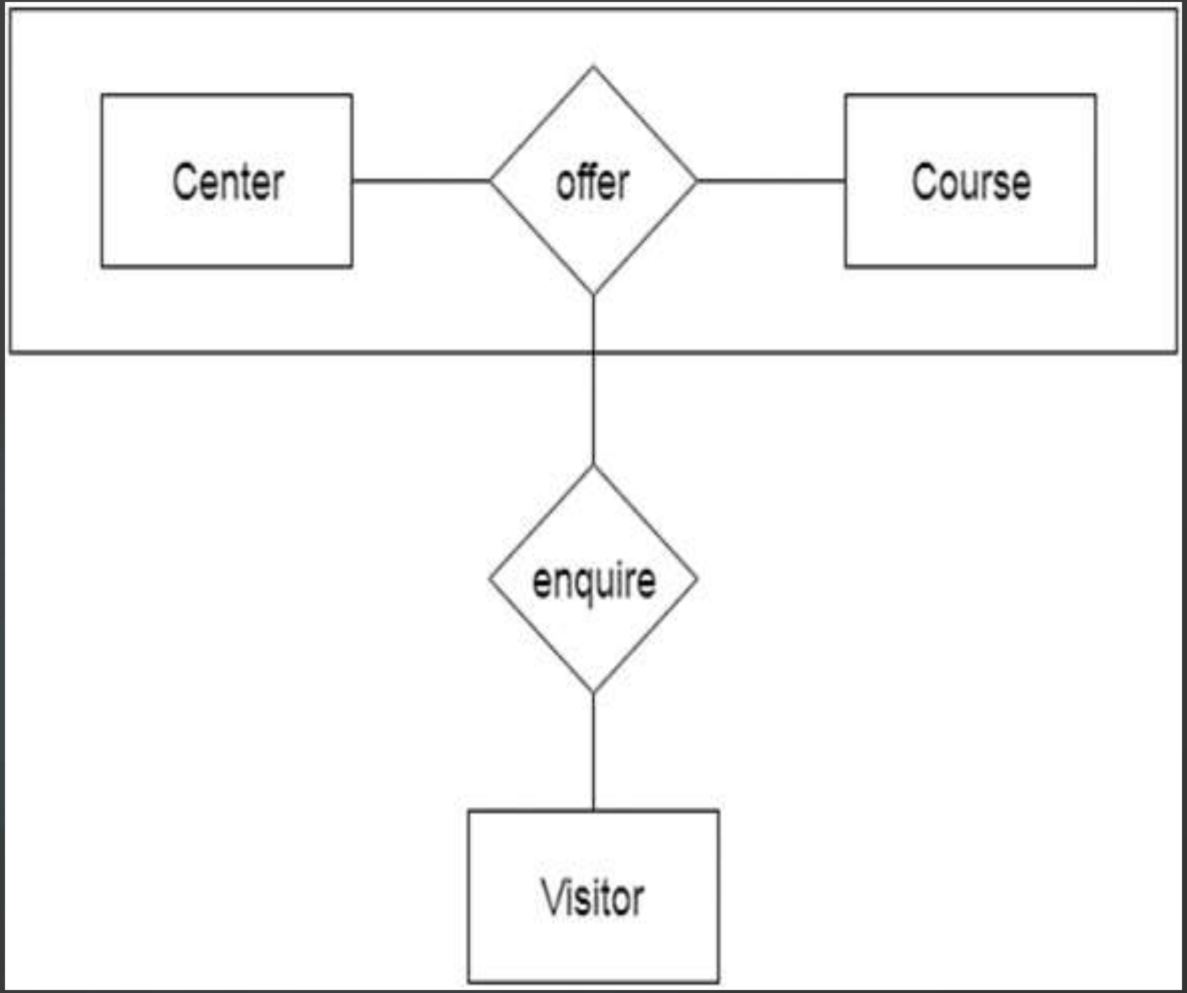
- **The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets. This is called as Attribute Inheritance.**

  Example: *customer* and *employee* inherit the attributes of *person.* Thus, *customer* is described by its *name, street,* and *city* attributes, and additionally a *customer_id* attribute; *employee* is described by its *name, street,* and *city* attributes, and additionally *employee_id* and *salary* attributes.

- **A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.**

  Example The *officer, teller,* and *secretary* entity sets can participate in the *works_for* relationship set, since the super class *employee* participates in the *works.for* relationship.

AGGREGATION



In AGGREGATION, the relation between two entities is treated as a single entity.

In aggregation, relationship of an entity with its corresponding entities is aggregated into a higher level entity.

Example:

In the real world, if a visitor visits a coaching CENTER then he will never enquire about the COURSE only or just about the CENTER instead he will enquire about both.

CENTER (AN ENTITY) OFFERS THE COURSE (ANOTHER ENTITY) ACT AS A SINGLE ENTITY , WHEN THEY ARE IN A RELATIONSHIP WITH THE ENTITY VISITOR.

HOW TO DRAW E-R DIAGRAM FOR A SPECIFIED DATABASE ?

Steps:

1. **Identify the requirement of the database**

   'Student attends class. Each class is divided into one or more sections. Each class will have its own specified subjects. Students have to attend all the subjects of the class that he attends'.

2. **Identify the entities**

   STUDENT, CLASS, SECTION, SUBJECT are the entities.

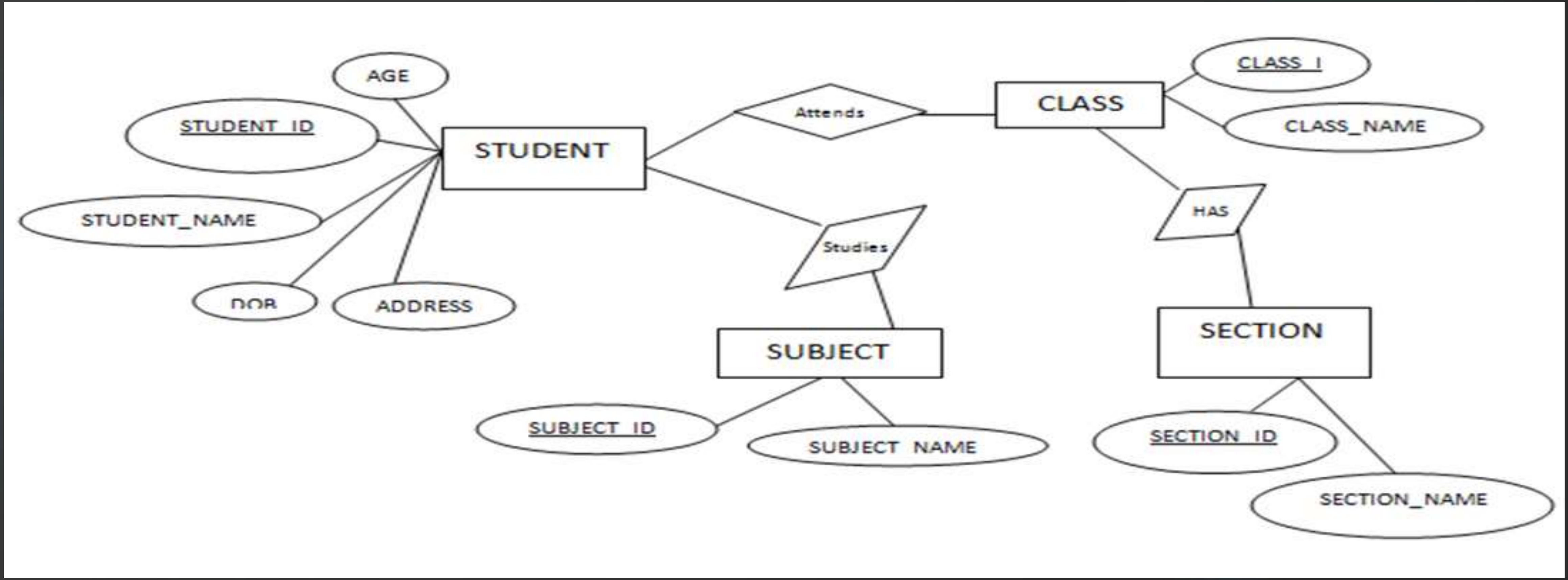
3. **Assume the Attributes of the entities**

STUDENT	CLASS	SECTION	SUBJECT
STUDENT_ID	CLASS_ID	SECTION_ID	SUBJECT_ID
STUDENT_NAME	CLASS_NAME	SECTION_NAME	SUBJECT_NAME
ADDRESS			
DOB			
AGE			
CLASS_ID			
SECTION_ID			

4. **Identify the relationships we have**

'Attends', 'has section', 'have subjects' and 'studies subjects' are the relations here.

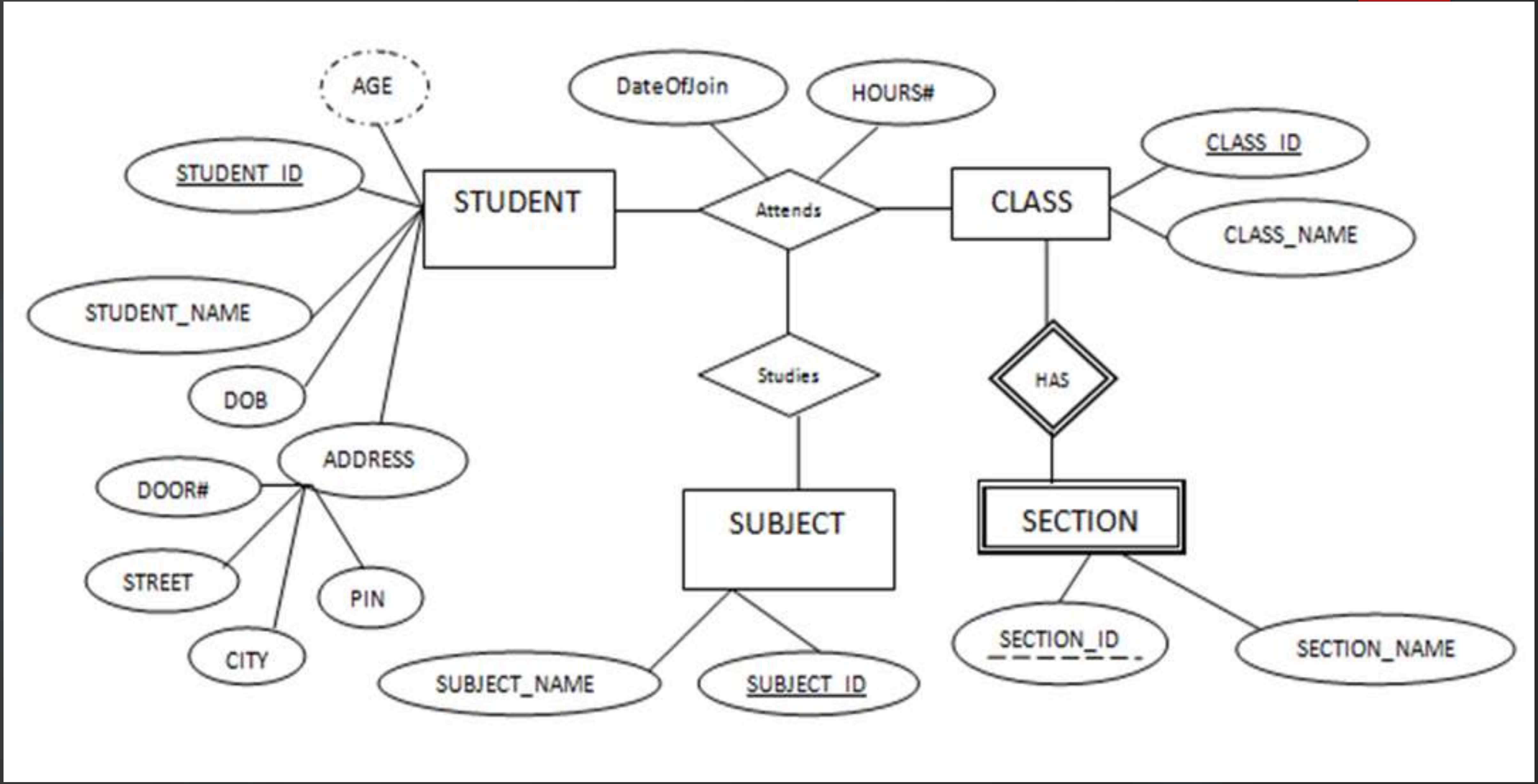
With this knowledge of requirement, we can draw the ER diagram as below.



Here,

- ***Age attribute can be derived*** from DOB. Hence we have to ***draw dashed oval.***
- ***Address is a composite attribute***. We have to ***draw its sub attributes too***. So that we will be very clear about his address details.
- ***There is no relation mentioned between Student and Section. Section is mapped only with Class. Existence of Section depends on Class. Each Class entity in Class entity set may be related with all the Section entities mentioned in the Section entity set.*** Hence, **SECTION IS A WEAK ENTITY**. Hence we have to ***represent it using Double Rectangle.***
- If we look at '**attends**' **relationship** between **STUDENT** and **CLASS**, we can have '**Joining Date**' and '**Total Number of Hours**' attributes. But it is an attribute of relation. We have to show them in the diagram.
- **Since each class is having different subjects and Students attends those subjects, we can modify the relation 'studies' to 'has' relation on the relation 'attends'.**

Now the diagram will change to reflect all above points.



RULES FOR TRANSFORMING ER DIAGRAM INTO TABLES

- ***Convert all the Entities in the diagram to tables.***

  All the entities represented in the rectangular box in the ER diagram become independent tables in the database.

- ***All single valued attributes of an entity is converted to a column of the table***
- ***All the attributes, whose value at any instance of time is unique, are considered as columns of that table.***
- ***Key attribute in the ER diagram becomes the Primary key of the table.***
- ***Declare the foreign key column, if applicable.***
- ***Any multi-valued attributes are converted into new table.***
- ***Any composite attributes are merged into same table as different columns.***
- ***One can ignore derived attribute, since it can be calculated at any time.***

RULES FOR TRANSFORMING ER DIAGRAM INTO TABLES

Convert Weak Entity Sets

- Weak entity is also represented as table. All the attributes of the weak entity forms the column of the table.
- The partial key attribute (discriminator) represented in the diagram cannot form the Primary key of this table. We have to add a foreign key column, which would be the Primary key column of its strong entity. This foreign key column along with the Partial key attribute column forms the Primary key of the table.

Representing 1:1 relationship

- Foreign key column can be added in either of the table, depending on the developer’s choice.

Representing 1:N relationship

- The primary key at 1 cardinality entity is added as foreign key to N cardinality entity

RULES FOR TRANSFORMING ER DIAGRAM INTO TABLES

Representing M:N relationship

- Both the participating entities are converted into tables, and a new table is created for the relation between them.
- Primary keys of entity tables are added into new table to form the composite primary key.
- We can add any additional columns, if present as attribute of the relation in ER diagram.

Self Referencing 1:N relation

- Primary key column acts as a foreign key in the same table.

Self Referencing M:N relation

- Both the primary keys from both tables act as a composite primary key in the new table. This reduces the storing of redundant data and consistency in the database.